User Interface - Bug #5118

TREELIST widget issues

01/28/2021 06:35 PM - Greg Shah

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

Description

Related issues:

Related to User Interface - Feature #5151: Add complete support for tree-list...	**WIP**
Related to User Interface - Bug #5280: TREEVIEW: Vertical rollover feature: a...	**Test**
Related to User Interface - Bug #5622: TREEVIEW widget issues	**WIP**

History

#1 - 01/28/2021 06:36 PM - Greg Shah

- Subject changed from TREE-LIST widget issues to TREELIST widget issues

As found in #5084, by Vladimir:

TREELIST issues:

1. Synchronization of other tree, columns and node attributes was not checked.
2. Setting sorted columns problem (see #5084-35)
3. Fixed columns problem (setting the attribute does nothing)
4. Column titles are not "dotted" when trimmed.
5. The contents of the last column cells is not clipped at right column border.
6. Column text right clipping boundary does not match that in column headers.
7. Incorrect focused row rendering in columns: next unfocused row is partly painted over the focused row grey background.
8. In OCX vertical scrolling is by rows. In FWD scrolling is by pixels (and **very** slow).
9. In OCX setCellXXXValue(index, YYY) with invalid index throws an error, in FWD it does nothing. At the moment I decided to log this situation as warning.
10. In OCX setCellIconValue(cellNo, iconId) set the string value to iconId. It is an error in a customer application. FWD mimic this erroneous behavior. So formally this is not an error. Probably, we should notify the customer application developers about this problem.

#3 - 01/29/2021 05:30 AM - Adrian Lungu

Regarding the changes to TreeWidgetBase.removeNode(), note that the focus change can also happen when the selected node was in the sub-tree of the removed node. Therefore _getSelecteNodeId() == nodeId.intValue() is not a sufficient condition. That is why the use of the TreeWidgetBase.validSelection() in the first place, which checks if the selected node is still valid (after remove). Another proper way to do this is to check for the ancestors of the selected node to identify if one of them is the removed node.

#4 - 01/29/2021 08:12 AM - Vladimir Tsichevski

Adrian Lungu wrote:

> Regarding the changes to TreeWidgetBase.removeNode(), note that the focus change can also happen when the selected node was in the sub-tree of the removed node. Therefore _getSelecteNodeId() == nodeId.intValue() is not a sufficient condition. That is why the use of the TreeWidgetBase.validSelection() in the first place, which checks if the selected node is still valid (after remove). Another proper way to do this is to check for the ancestors of the selected node to identify if one of them is the removed node.

Agreed, will fix this today.

#5 - 01/29/2021 08:14 AM - Vladimir Tsichevski

FIXED in rev. 12517. Another thing still unfixed (or a regression?): when the node is collapsed, and some node descendant is focused, the focus must change to the collapsed node itself.

#6 - 01/29/2021 09:08 AM - Vladimir Tsichevski

FIXED. The IdToPointer is implemented in FWD, but under different name nodeKeyToId. So the call to IdToPointer is not converted correctly.

Either the Java method name need to be changed, or conversion somehow fixed.

#7 - 01/29/2021 09:27 AM - Vladimir Tsichevski

Vladimir Tsichevski wrote:

> Adrian Lungu wrote:
>
> > Regarding the changes to TreeWidgetBase.removeNode(), note that the focus change can also happen when the selected node was in the sub-tree of the removed node. Therefore _getSelecteNodeId() == nodeId.intValue() is not a sufficient condition. That is why the use of the TreeWidgetBase.validSelection() in the first place, which checks if the selected node is still valid (after remove). Another proper way to do this is to check for the ancestors of the selected node to identify if one of them is the removed node.
>
> Agreed, will fix this today.

Fixed in 11967.

#8 - 01/29/2021 10:26 AM - Adrian Lungu

Vladimir Tsichevski wrote:

> Another thing still unfixed (or a regression?): when the node is collapsed, and some node descendant is focused, the focus must change to the collapsed node itself.

I fixed this for server-side node collapse: TreeFace.collapseNode(). However, the client-side collapse (handling mouse click) is not fixed.

#10 - 02/18/2021 04:38 AM - Adrian Lungu

- Related to Feature #5151: Add complete support for tree-list OCX triggers added

#11 - 02/18/2021 06:13 AM - Vladimir Tsichevski

FIXED in rev 12518.
Another difference:

In FWD a node is collapsed/expanded on mouse up event for **any mouse button** in **any part of tree node area**.
In OE only **left mouse up** in the expand/collapse node **button area** does this.

The difference causes at least one annoying effect: then you opens the node popup menu with right mouse click, the node always does expand/collapse either.

#15 - 03/15/2021 04:40 PM - Vladimir Tsichevski

An issue from #4908-4:

```
By pressing and releasing left and right mouse buttons while dragging a tree column header, I can cause a cras
h caused by IllegalStateException in MenuItemGuiImpl at AbstractWidget.screenPhyscalLocation, l. 1365:
```

```
         throw new IllegalStateException("widget is not in the container");
```

Note: this problem is probably not related to the tree list, but it can be reproduced with a tree list.

#16 - 03/16/2021 03:32 AM - Hynek Cihlar

Vladimir Tsichevski wrote:

An issue from #4908-4:

By pressing and releasing left and right mouse buttons while dragging a tree column header, I can cause a crash caused by IllegalStateException in MenuItemGuiImpl at AbstractWidget.screenPhyscalLocation, l. 1365:

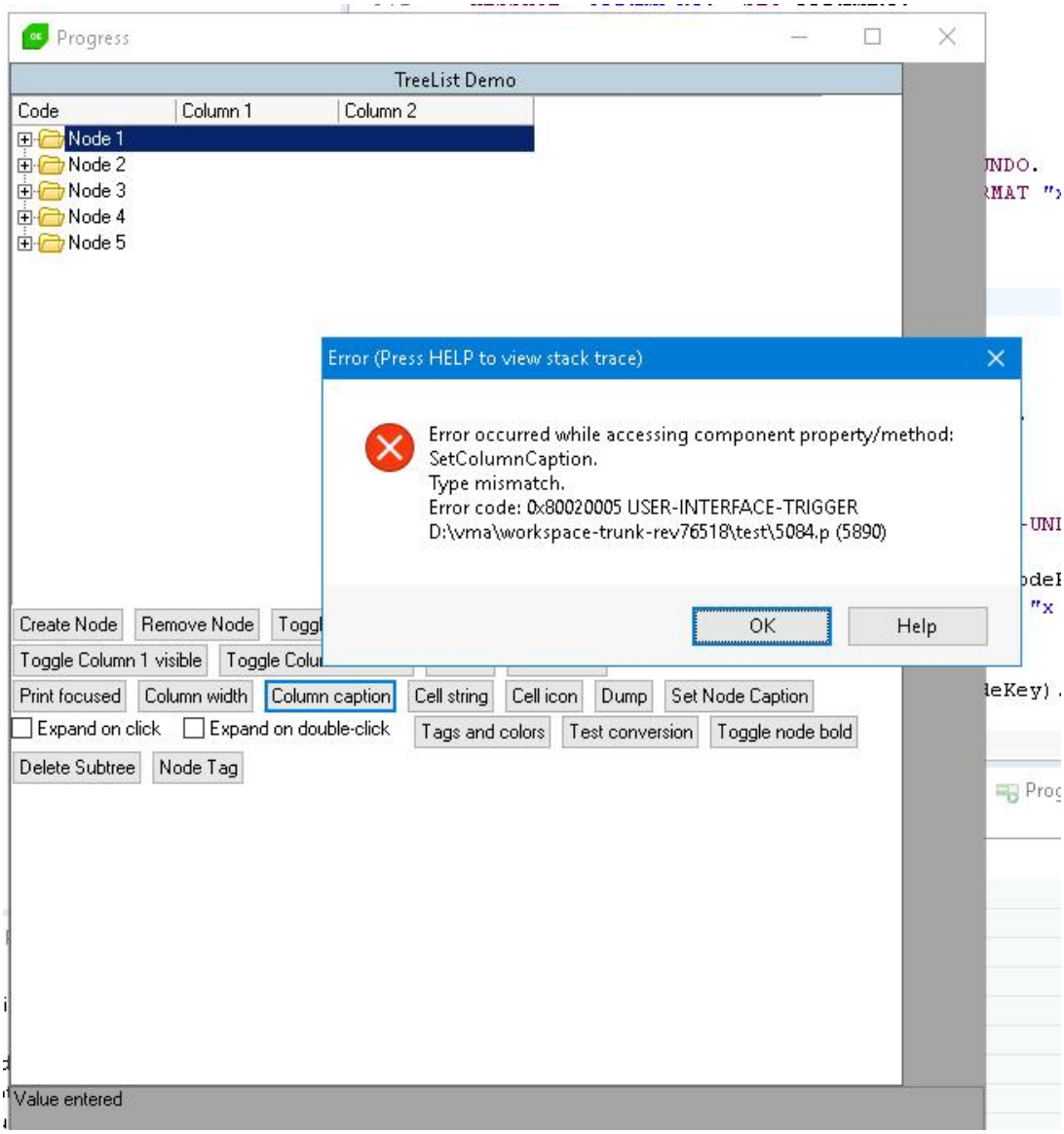
My guess is that the user action also triggers context menu processing. Try to add a break point in the MenuItemGuiImpl constructor or at the point where the exception is thrown and inspect the call stack.

#19 - 03/23/2021 09:10 AM - Vladimir Tsichevski

- File 5118-unknown-caption-value.png added

Another difference:

In 4gl it is illegal to pass an unknown character value as the column caption:



In FWD a question mark is set as the caption, no error is generated.

I think, the same difference does exist for all API calls to the customer-specific version of the tree OCX.
We have to decide what to do in these situations:

1. Silently do nothing
2. Raise some home-brewed application error.

I think, silently using the '?' as the value is not correct in situations like this.

#20 - 04/07/2021 11:05 AM - Vladimir Tsichevski

3821c, rev. 12253. The width and caption of the first default column changed to 101 and "code" to make it compatible with the vmacmtree1.

#21 - 04/07/2021 11:12 AM - Greg Shah

Is this customer specific? Shouldn't such values be handled by 4GL source file changes? I don't see why a generic widget should have these values.

#22 - 04/07/2021 11:16 AM - Vladimir Tsichevski

Greg Shah wrote:

> Is this customer specific? Shouldn't such values be handled by 4GL source file changes? I don't see why a generic widget should have these values.

Yes, it is. Probably, we may use the default width of the first column 101 (instead of 200), which is reasonable.

#23 - 04/07/2021 11:24 AM - Greg Shah

OK, leave the 101 and remove the "code". Please propose a 4GL code edit for setting the "code" value. We can send it to the customer for inclusion in their application, once we've confirmed it works here.

#24 - 04/07/2021 11:46 AM - Vladimir Tsichevski

Greg Shah wrote:

> OK, leave the 101 and remove the "code".

Done in rev. 12253.

> Please propose a 4GL code edit for setting the "code" value. We can send it to the customer for inclusion in their application, once we've confirmed it works here.

At the moment, the tree caption setting (the SetColumnCaption call) is used in the **redacted_customer_program_name** only. Other 4gl usages exist, but they are not converted. In **redacted_customer_program_name**, the width and caption of the first (already existing) column is **always** explicitly set from the DB values (lines 2153-2158).

Also, there is no way the caption can be read back from the tree.

As the result, we may safely assume, the initial tree column caption value is irrelevant in **redacted_customer_name**. So no patching is required.

#25 - 05/13/2021 05:24 PM - Vladimir Tsichevski

FIXED. Another issue regarding SORTED-COLUMNS.

Calling TreeListWidget.setSortedColumns(NumberType value) causes IndexOutOfBoundsException here:

```
    Daemon Thread [Conversation [00000008:bogus]] (Suspended (exception IndexOutOfBoundsException))
    ArrayList<E>.rangeCheck(int) line: 659
    ArrayList<E>.remove(int) line: 498
    TreeListWidget<N>.setSortedColumns(NumberType) line: 200
    Five084.lambda$6() line: 309
```

The reason is that wrong methods remove(int) and add(int) are used to update the array instead of remove(Object) and add(Object):

```
   public void setSortedColumns(NumberType value)
   {
      if (value == null || value.isUnknown() || value.intValue() < 0)
      {
         getAttr("sort", () -> config.sort).clear();
      }
      else
      {
         int index = value.intValue();
         if (index >= 0 && index < columns.size())
         {
            getAttr("sort", () -> config.sort).remove(index);
                                               ^^^^^^
            getAttr("sort", () -> config.sort).add(index);
         }
      }

   }
```

#27 - 05/18/2021 12:08 PM - Vladimir Tsichevski

FIXED

Another issue described first in #5282: the node value can be of a type other than String.

As the result, a ClassCastException is thrown in the following line of ClassicTheme.drawTreeViewNode():

```
     String text = (String) node.value.getValue(valueIndex);
```

#28 - 05/18/2021 12:08 PM - Vladimir Tsichevski

- Status changed from New to WIP

#29 - 05/18/2021 12:31 PM - Vladimir Tsichevski

Another issue: scrolling by mouse wheel does not work: with Swing the tree can usually be scrolled by at most one row. With WEB no scrolling happens at all. See #4868.

#30 - 05/18/2021 12:43 PM - Vladimir Tsichevski

The problem in #5118-19 is true for other calls, for example, for at least the last two arguments of SetCellIconValue(nodePtr, columnNo, iconNo). FWD client **crashes** if an unknown value is passed.

#31 - 05/18/2021 01:42 PM - Vladimir Tsichevski

Vladimir Tsichevski wrote:

> Another issue described first in #5282: the node value can be of a type other than String.
> As the result, a ClassCastException is thrown in the following line of ClassicTheme.drawTreeViewNode():
>
> [...]

Fixed in 3821c rev. 12434.

#32 - 05/18/2021 03:13 PM - Vladimir Tsichevski

FIXED

There is also an unimplemented "icon column datatype" feature described in #5282-5.

#33 - 05/18/2021 06:47 PM - Vladimir Tsichevski

FIXED in rev. 12481

Text cell value are left-trimmed in FWD. They should not.

#34 - 05/19/2021 01:47 PM - Vladimir Tsichevski

- File 5118-34.mp4 added

FIXED in rev. 12481

The first tree column is rendered and active outside its bounds!

See the 5118-34.mp4 video demonstration:

If I drag the second column left, the following things are wrong:

1. There is no limit on the gap between the column left side and the tree left side. There must be a limit equal to the minimum column width.
2. All first column visible components are rendered left to the second column right side.
3. Not only they are rendered, but stay active
4. The second column becomes inactive in this position.

I suspect, dragging the boundary between the first and second columns does not affect the first column width.

#35 - 05/19/2021 02:26 PM - Vladimir Tsichevski

FIXED

Vladimir Tsichevski wrote:

> Another issue regarding SORTED-COLUMNS.
>
> Calling TreeListWidget.setSortedColumns(NumberType value) causes IndexOutOfBoundsException here:

The #5364 issue is dedicated to this problem.

#37 - 05/19/2021 03:12 PM - Vladimir Tsichevski

- Related to Bug #5280: TREEVIEW: Vertical rollover feature: assure TREEVIEW is compatible with MS Treeview control added

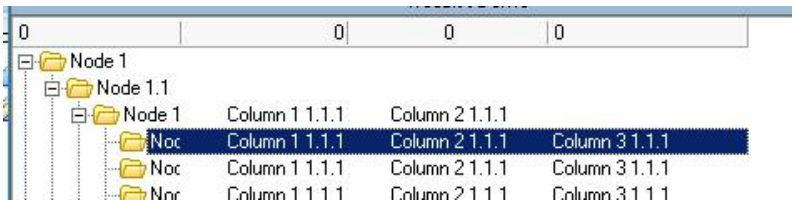
#39 - 05/21/2021 03:51 PM - Vladimir Tsichevski

- File treelist-fwd.png added

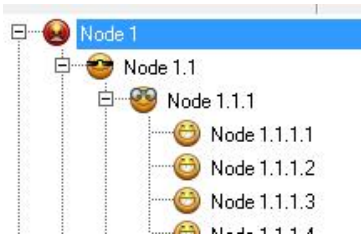
- File treelist-oe.png added

Tree element positions differs from the original:

In original tree the horizontal distance between the columns (vertical dotted lines) is 16 pixels, the row height is also 16 pixel:



In FWD this distances are more: 26 and 20 pixels correspondingly:



Also, icons are not centered relative to vertical dotted lines.

#40 - 05/21/2021 04:46 PM - Vladimir Tsichevski

- File WrongHorizontalScrollbar.mp4 added

- File WrongHorizontalScrollbar2.mp4 added

- File WrongVerticalScrollbar.mp4 added

Scrollbar visibility and (sometimes) displayed values are wrong:

WrongHorizontalScrollbar.mp4

In this video a node is expanded, which causes the **vertical** scrollbar appearance (correct) and also the **horizontal** scrollbar appearance (incorrect).

WrongVerticalScrollbar.mp4

In this video a column is resized, which causes the **horizontal** scrollbar appearance (correct) and also the **vertical** scrollbar appearance (incorrect). The value displayed by the vertical scrollbar is also incorrect.

WrongHorizontalScrollbar2.mp4

In this video a column width is increased first, which causes the **horizontal** scrollbar appearance (correct), and then is decreased back, which should cause the horizontal scrollbar disappear, but the scrollbar does not disappear as expected.

#41 - 05/21/2021 05:06 PM - Vladimir Tsichevski

Hynek,

there is the following field in TreeConfig

```
/** Flag to toggle sorting */
public boolean sorted;
```

It seems, this field is orphaned. It is set to true in the constructor and is never changed afterwards.
Is this field still needed? Otherwise it is better be removed for good.

#42 - 05/21/2021 05:15 PM - Vladimir Tsichevski

FIXED

In the original TREELIST it is possible to reset column sorting by clicking on the column header with the Ctrl key. In FWD this is currently not supported.

#43 - 05/21/2021 05:20 PM - Hynek Cihlar

Vladimir Tsichevski wrote:

> Hynek,
>
> there is the following field in TreeConfig
> [...]
>
> It seems, this field is orphaned. It is set to true in the constructor and is never changed afterwards.
> Is this field still needed? Otherwise it is better be removed for good.

If there is an API method in the original treeview and treelist native controls to toggle sorting this field will be needed to implement this feature.

#44 - 05/22/2021 01:54 PM - Vladimir Tsichevski

Hynek Cihlar wrote:

> If there is an API method in the original treeview and treelist native controls to toggle sorting this field will be needed to implement this feature.

There is no such method in vmacmtree1.

#45 - 05/23/2021 01:39 PM - Hynek Cihlar

Vladimir Tsichevski wrote:

> Hynek Cihlar wrote:
>
> > If there is an API method in the original treeview and treelist native controls to toggle sorting this field will be needed to implement this feature.
>
> There is no such method in vmacmtree1.

There is Sorted property in MS Treeview OCX.

#46 - 05/24/2021 08:03 AM - Vladimir Tsichevski

Hynek Cihlar wrote:

> Vladimir Tsichevski wrote:
>
> > Hynek Cihlar wrote:
> >
> > > If there is an API method in the original treeview and treelist native controls to toggle sorting this field will be needed to implement this feature.
> >
> > There is no such method in vmacmtree1.
>
> There is Sorted property in MS Treeview OCX.

Is it used currently in any project?

#47 - 05/24/2021 10:38 AM - Hynek Cihlar

Vladimir Tsichevski wrote:

> Hynek Cihlar wrote:
>
> > There is no such method in vmacmtree1.
>
> There is Sorted property in MS Treeview OCX.
>
> Is it used currently in any project?

No, but there is good chance it will be on a future one.

#48 - 05/24/2021 10:41 AM - Greg Shah

Our intention is to have a completed and compatible OCX replacement. We don't want to have to add features with every new project.

#49 - 05/24/2021 11:37 AM - Vladimir Tsichevski

Greg Shah wrote:

> Our intention is to have a completed and compatible OCX replacement. We don't want to have to add features with every new project.

Then we need to define, implement and document this feature either?

#50 - 05/24/2021 11:43 AM - Greg Shah

No, but I don't want to remove stuff that will need to be added back later.

#51 - 05/24/2021 02:56 PM - Vladimir Tsichevski

FIXED in rev. 12481

Rendering trelist icon columns: in FWD icons are always centered regardless of the column alignment.
In original OCX column alignment is always honored.

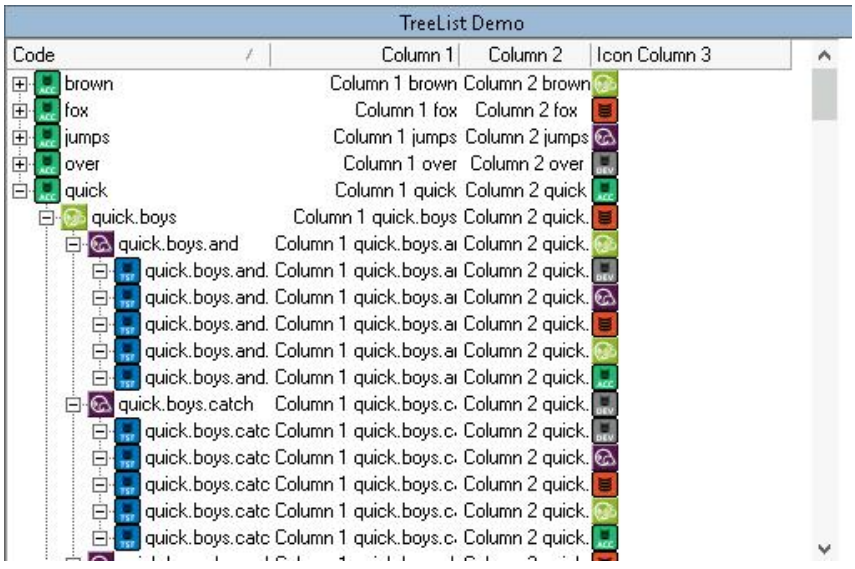
#52 - 05/24/2021 07:16 PM - Vladimir Tsichevski

- File 5115-narrow-column-FWD.png added

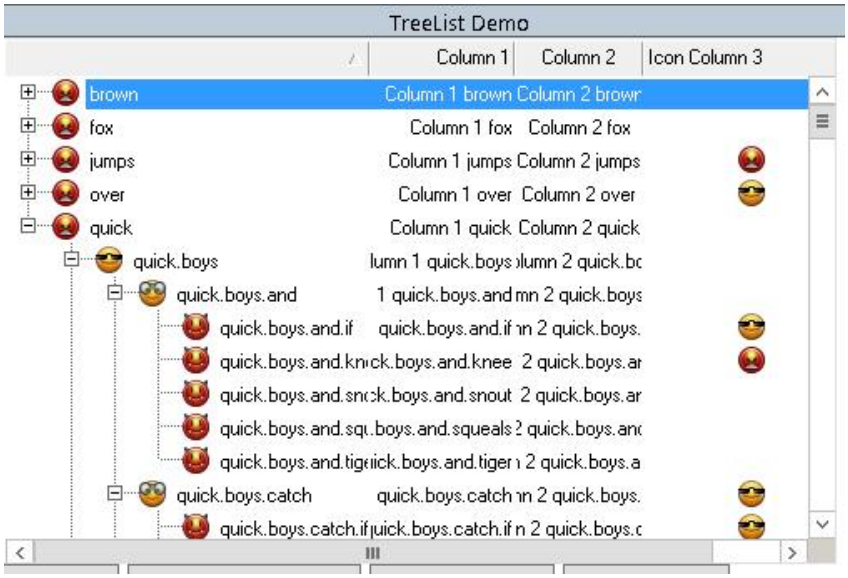
- File 5115-narrow-column-OE.png added

FIXED in rev. 12481

Narrow column alignment is incorrect in FWD: in original OCX, it the column text does not fit in the column width, it is always right-aligned:



In FWD the column alignment does not depend in the text length:



#53 - 05/24/2021 07:19 PM - Vladimir Tsichevski

Unimplemented OCX call: GetNodeVisibleInViewPort. This call return value of the logical type is used in customer code. In FWD this function is implemented as a stub and always returns the undefined value.

#54 - 05/24/2021 07:23 PM - Vladimir Tsichevski

In FWD, if a child node is added, the parent node always expands. In the original OCX this does not happen.

#55 - 05/24/2021 07:28 PM - Vladimir Tsichevski

In FWD, if a child node is added, and a sorting by some column is set, nodes are automatically sorted. In the original OCX this does not happen, and the new node is always added to the end of parent's child list.

To sort nodes, a special procedure ReSort does exist in the original OCX, which is implemented in FWD as a stub only, and does nothing.

#56 - 05/25/2021 11:33 AM - Vladimir Tsichevski

Vladimir Tsichevski wrote:

> In the original TREELIST it is possible to reset column sorting by clicking on the column header with the Ctrl key. In FWD this is currently not supported.

Fixed in 3821c rev. 12452.

#58 - 05/26/2021 08:03 AM - Vladimir Tsichevski

FIXED in rev. 12481

Incorrect handling values passed to setCellString for icon type cells.

In original OCX, all cell values are effectively stored "as-is" in the node data, so a string value is stored as a string with no conversion.
When a cell is rendered, the value is coerced to the desired data type.
For icon type columns, a string value is trimmed and parsed into an integer, the result is used as index into the cell icon image list.
If the string cannot be parsed into integer, no error is reported, and no icon rendered.

In FWD the string value is displayed instead of an icon.

#59 - 05/26/2021 11:46 AM - Vladimir Tsichevski

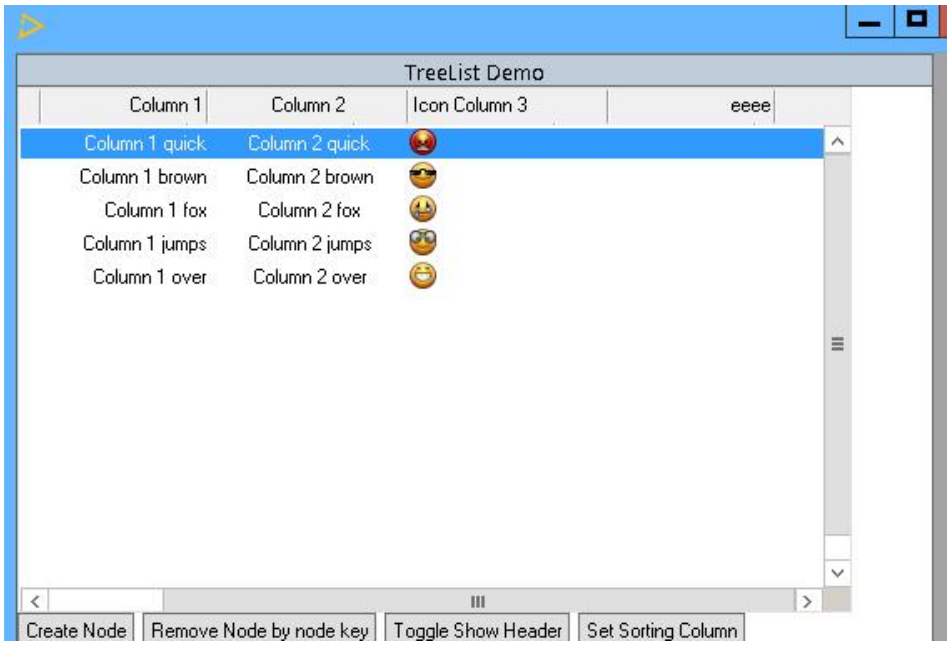
- File 5118-layout-before-cell-value-changed.png added

- File 5118-layout-broken-after-cell-value-changed.png added

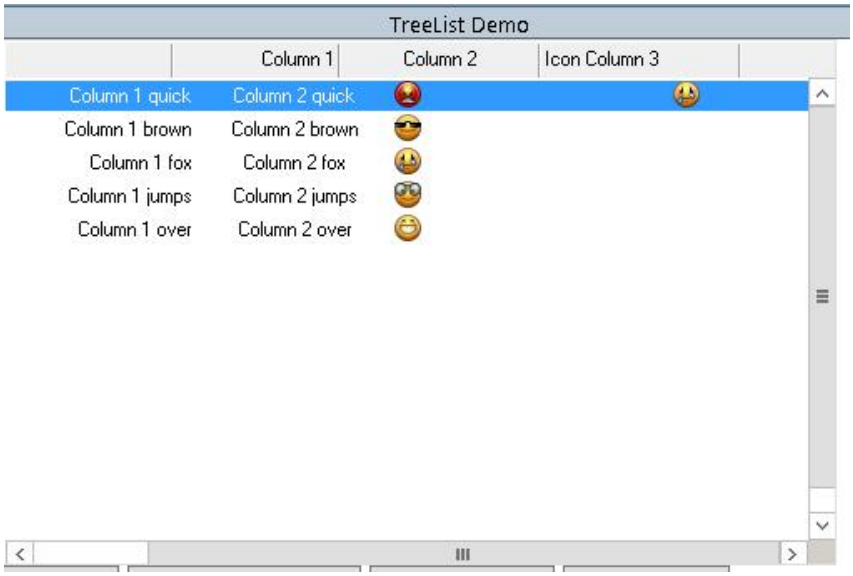
Tree headers layout is broken after a cell value changed in a horizontally scrolled tree list:

The test application window before a cell value was set in the column 4.

Before the change:



after the change:



The tree list header is rendered un-scrolled.

#60 - 05/26/2021 02:02 PM - Vladimir Tsichevski

- File 5118-oe-column-colors.png added

FIXED in rev. 12481

Background colors passed as an argument to tree list column create functions, do not reach the tree cell rendering procedure, so the tree background is always white in FWD.

In OE the same tree with non-white column background looks like this:

#61 - 05/26/2021 02:20 PM - Vladimir Tsichevski

PARTLY FIXED in rev. 12481
FIXED in rev. 13480.

The following TreeList procedures:

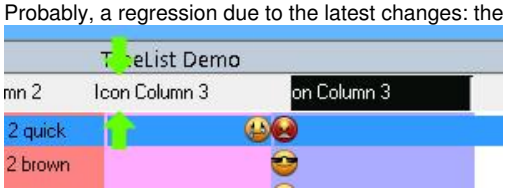
```
   void setCellFgColor(NumberType nodeId, NumberType colIndex, NumberType fgrValue);
   void setCellBgColor(NumberType nodeId, NumberType colIndex, NumberType bgrValue);
```

are used in customer code, but are currently implemented in FWD as stubs only.

#62 - 05/27/2021 08:39 AM - Vladimir Tsichevski

- File 5118-62.png added

Probably, a regression due to the latest changes: the horizontal position of arrows while a column header is dragged is incorrect (shifted right).



#64 - 05/27/2021 10:15 AM - Vladimir Tsichevski

FIXED in rev. 12481

In FWD, there is no line dividing the headers and cells. In the original OCX there is.

#65 - 05/27/2021 10:46 AM - Vladimir Tsichevski

FIXED
When the user starts column resizing by dragging a column dividing line, the whole tree header is re-painted shifted one pixel down. After resizing is complete, the header is not repainted and stays shifted. Changing node selection restores the original view.

#66 - 05/27/2021 06:12 PM - Vladimir Tsichevski

FIXED in rev. 12481

Also, on the image in #5118-62, it is seen that drawing the focused node is incorrect: the blue strip is painted outside of the cells area from the right.

The problem origin is that a wrong column width is passed to the procedure, which draws the first column: the whole tree body width is passed instead of the column width.

This also probably explains the #5118-34 issue.

#67 - 05/27/2021 06:49 PM - Vladimir Tsichevski

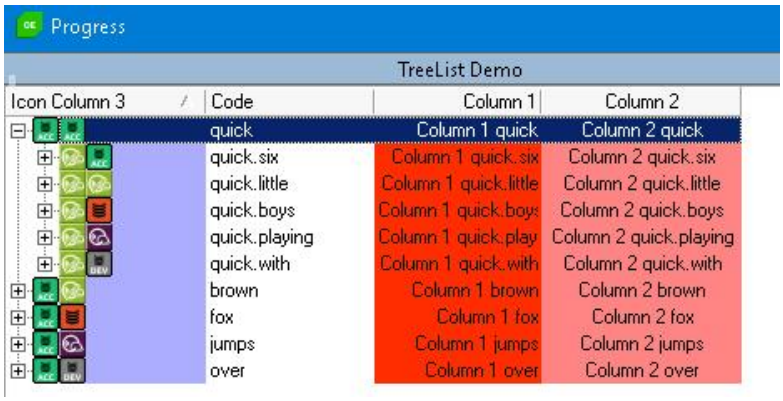
- File 5118-67-FWD.png added

- File 5118-67-OE.png added

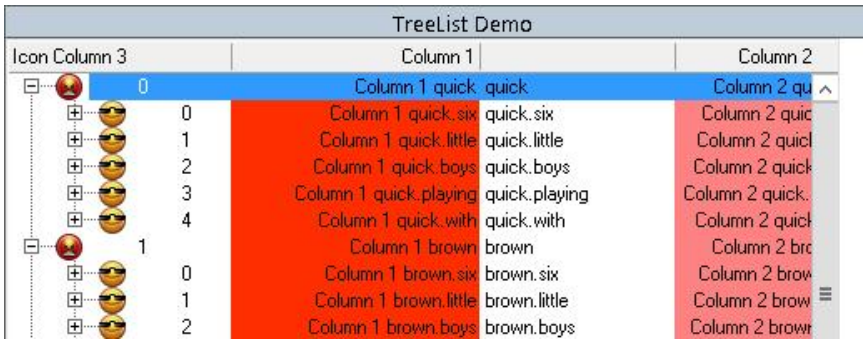
FIXED

In original OCX when dragged to the leftmost position, cells of an "icon-type" column are rendered exactly the same as in other positions. In FWD, the cell values are rendered as text.

In OE:



In FWD:



#68 - 05/28/2021 10:50 AM - Vladimir Tsichevski

Tree node background and foreground colors seem to be supported, but setting these does not affect the tree view. May be, is is a regression.

If set, these colors must override the corresponding column colors.
Cell colors must override node colors.

#69 - 05/28/2021 01:20 PM - Vladimir Tsichevski

FIXED in rev. 12481

Caption rendering: if the text does not fit the caption width, is must be left-aligned.

#70 - 05/31/2021 11:15 AM - Vladimir Tsichevski

FIXED in rev. 12481

TreeWidgetBase.clearNodesImageList() destroys **all** images related to the widget, including the cell icon images. It should affect node images only.

#71 - 05/31/2021 04:50 PM - Vladimir Tsichevski

The MultiSelect feature (both the getter and the setter) is not currently implemented, but it is used in customer code.

#72 - 05/31/2021 07:31 PM - Vladimir Tsichevski

FIXED in rev. 12481
In the original TREELIST OCX, at the moment a TREELIST is created, there are 3 icons already in the tree node image list.
These images are supposedly come from the associated .wrx file or some DLL resource.
Note: these icons are cleared with the ClearNodesImageList call.

So, in FWD we need to adjust the logic to the following:

1. Add these 3 stock images to the tree image list at the moment of tree (or tree config) creation.
2. From that moment on, never bother about stock images.

#73 - 06/01/2021 02:51 AM - Sergey Ivanovskiy

Please take into account that Image List OCX can be set for Tree View OCX. These testcases ./treeview/test-image-list-1.w and ./treeview/test-image-list-2.w test this functionality. They can be found in testcases/uast/treeview/.

#74 - 06/01/2021 05:02 AM - Vladimir Tsichevski

Sergey Ivanovskiy wrote:

> Please take into account that Image List OCX can be set for Tree View OCX. These testcases ./treeview/test-image-list-1.w and ./treeview/test-image-list-2.w test this functionality. They can be found in testcases/uast/treeview/.

Thank you. Sergey. I've seen this ImageList feature, and I am trying not to break things here. Will use these test either.

#75 - 06/01/2021 07:24 AM - Hynek Cihlar

Vladimir Tsichevski wrote:

> In the original TREELIST OCX, at the moment a TREELIST is created, there are 3 icons already in the tree node image list.
> These images are supposedly come from the associated .wrx file or some DLL resource.
> Note: these icons are cleared with the ClearNodesImageList call.
>
> So, in FWD we need to adjust the logic to the following:
>
> 1. Add these 3 stock images to the tree image list at the moment of tree (or tree config) creation.
> 2. From that moment on, never bother about stock images.

Sounds good. But please make sure TREEVIEW still works as expected.

#76 - 06/01/2021 07:25 AM - Hynek Cihlar

Hynek Cihlar wrote:

> Vladimir Tsichevski wrote:
>
> > In the original TREELIST OCX, at the moment a TREELIST is created, there are 3 icons already in the tree node image list.
> > These images are supposedly come from the associated .wrx file or some DLL resource.
> > Note: these icons are cleared with the ClearNodesImageList call.
> >
> > So, in FWD we need to adjust the logic to the following:
> >
> > 1. Add these 3 stock images to the tree image list at the moment of tree (or tree config) creation.
> > 2. From that moment on, never bother about stock images.
>
> Sounds good. But please make sure TREEVIEW still works as expected.

Never mind, you already discussed this above.

#77 - 06/01/2021 08:01 AM - Vladimir Tsichevski

Sergey Ivanovskiy wrote:

> Please take into account that Image List OCX can be set for Tree View OCX. These testcases ./treeview/test-image-list-1.w and ./treeview/test-image-list-2.w test this functionality. They can be found in testcases/uast/treeview/.

Unfortunately, these examples do not convert correctly, the 4gl line:

```
chCtrlFrame-2:TreeView:ImageList=chCtrlFrame:ImageList.
```

converts to invalid Java:

```
new handle(hTreeView.unwrapTreeView().setImageList(hImageList.unwrapImageList()));
```

#78 - 06/01/2021 11:58 AM - Vladimir Tsichevski

Vladimir Tsichevski wrote:

> Unfortunately, these examples do not convert correctly, the 4gl line:

Rev. 12481: after the converted Java code were fixed manually, both test examples work as expected.

#79 - 06/01/2021 12:27 PM - Vladimir Tsichevski

Fixed by rev. 12481: #5118-32, #5118-69, #5118-72.

#80 - 06/01/2021 06:27 PM - Eugenie Lyzenko

Vladimir,

The customer application fix to be in sync with 12481 - 12487 caused another extension jar failure:

```
...
censored
...
```

Why do you need to include import a.stange.class.to.be loaded.*;?

Could you please fix this ASAP.

#81 - 06/01/2021 07:12 PM - Vladimir Tsichevski

Eugenie Lyzenko wrote:

> Vladimir,
>
> Could you please fix this ASAP.

this was a glitch. Fixed.

Note: this customer-related problem is better to discuss in the appropriate project.

#82 - 06/01/2021 07:21 PM - Eugenie Lyzenko

Vladimir Tsichevski wrote:

> Eugenie Lyzenko wrote:
>
> > Vladimir,
> >
> > Could you please fix this ASAP.
>
> this was a glitch. Fixed.
>
> Note: this customer-related problem is better to discuss in the appropriate project.

Yes, you are right. I have removed dangerous info.

#83 - 06/03/2021 03:50 PM - Vladimir Tsichevski

FIXED. New findings:

Each cell in the original OCX has two distinct slots to keep the "cell value":

1. value (string, double date, or integer), always assigned by any SetXXXValue call.
2. display string

OCX calls treat these fields as follows:

1. SetCellStringValue assigns the **string** argument to the value
2. SetCellDateValue assigns the **date** argument to the value
3. SetCellDecimalValue assigns the **double-precision** argument to the value, convert it to a string (with decimal and thousand separators?) and assigns the result to string field
4. SetCellIconValue assigns the integer argument to the value
5. SetCellIntegerValue converts the integer argument to a string (with decimal and thousand separators?) and assigns the result to string field
6. GetCellStringValue returns the value converted to a string
7. CreateSubnode assigns the **caption** argument to the first cell string
8. NodeCaption getter and setters get and set the first cell string (not the value)

The string values are also used to **sort nodes**.

#85 - 06/08/2021 02:59 PM - Vladimir Tsichevski

Adrian Lungu wrote:

> Vladimir Tsichevski wrote:
>
> > Another thing still unfixed (or a regression?): when the node is collapsed, and some node descendant is focused, the focus must change to the collapsed node itself.
>
> I fixed this for server-side node collapse: TreeFace.collapseNode(). However, the client-side collapse (handling mouse click) is not fixed.

Now fixed in rev. 12517.

#86 - 06/08/2021 03:13 PM - Vladimir Tsichevski

Vladimir Tsichevski wrote:

> In FWD a node is collapsed/expanded on mouse up event for **any mouse button** in **any part of tree node area**.
> In OE only **left mouse up** in the expand/collapse node **button area** does this.

FIXED in rev 12518.

#87 - 06/08/2021 03:28 PM - Vladimir Tsichevski

- File 5118-bad-header-text-rendering.mp4 added

FIXED in rev. 12521. Greg Shah wrote:

> 4. Column titles are not "dotted" when trimmed.

They are indeed dotted, but at the wrong moments, and they are not "undotted" back, then they should.

See the 5118-bad-header-text-rendering.mp4 for an illustration.

Also: when there is not enough horizontal space for a header text, this text (after dotting) should be always left-aligned.

#88 - 06/09/2021 11:37 AM - Vladimir Tsichevski

Vladimir Tsichevski wrote:

> Greg Shah wrote:
>
> > 4. Column titles are not "dotted" when trimmed.
>
> They are indeed dotted, but at the wrong moments, and they are not "undotted" back, then they should.
>
> See the 5118-bad-header-text-rendering.mp4 for an illustration.
>
> Also: when there is not enough horizontal space for a header text, this text (after dotting) should be always left-aligned.

FIXED in rev. 12521.

#89 - 06/17/2021 05:41 PM - Vladimir Tsichevski

Refactoring the TREELIST/TREEVIEW implementation

1. Make TREEVIEW implemented as a layer over the TREELIST. TREEVIEW is a one-column TREELIST with caption display switched off.
2. Internal refactoring: remove usage of Java generics in TREELIST/TREEVIEW implementation classes.

Make TREEVIEW implemented as a layer over the TREELIST

Rationale

1. It is not hard to achieve: TREEVIEW **is** de-facto just a one-column TREELIST with caption display switched off.
2. This change automatically solves a number of known TREELIST issues related to first tree list column rendering.
3. It makes implementation much simpler and code more readable.

The tree list **column rendering issue**: to render the first tree list column, the procedure ClassicTheme.drawTreeViewBody is currently used, which is also used for drawing TREEVIEW-s. This procedure knows nothing about the TREELIST columns and more complex data connected with the columns, like **custom colors** and column **data type** support. As the result, the first TREELIST column is rendered incorrectly in many situations described earlier in this issue comments.

Implementation details

1. The necessary TREELIST-related data and methods are pulled up from TREELIST-implementation classes to tree base classes.
2. Optionally: **tree view nodes** and **tree list nodes** remain implemented in different subclasses of base tree node interfaces or classes. This may allow tree view node implementation to take a bit less memory.

Change state

Work to do.

Remove usage of Java generics in TREELIST/TREEVIEW implementation classes

Rationale

1. This change is **required** to make the first change possible.
2. TREEVIEW and TREELIST and all parts correspondingly have common ancestors. TreeWidgetBase, TreeNodeResource, TreeNodeEntry, TreeBodyGuiImp, TreeGuiImpl, TreeConfig etc. - these are the common ancestors for two different class sets.
3. The conventional class ancestry is mixed with generics, and this prevents from using the base class instance references instead of concrete classes where possible.
4. Conventional class inheritance is all we need here, and Java generics need not be used where simple inheritance is enough.
5. Generics are useful if we are writing library for using with any unknown in advance types provided later by user. Our situation is completely different: we are dealing with a closed set of known and closely related classes.

Implementation details

1. Java generics parameters are removed from all related class definitions. Conventional Java type casts from base types to derived types used when concrete type is expected.
2. Optional refactoring: additional API methods are added to base interfaces, which would allow to decrease the number of casts.

Change state

Already **implemented** in my local FWD installation, smoke-tested, unpolished, not-committed.

#90 - 06/18/2021 04:46 AM - Sergey Ivanovskiy

Vladimir, TreeViewGuiImpl supports the editing of node labels but TreeListGuiImpl doesn't implement this functionality now. Did you take this functionality into account? Does tree list support manual and automatic drag and drop functionalities? I know that one screen of the customer's project uses manual drag and drop functionality. It means that drag and drop listeners are implemented by 4GL code. Now for the web client the dragged node under the mouse pointer isn't detected correctly by the converted 4GL code but the swing client doesn't have this issue #5172-50.

#91 - 06/18/2021 07:33 AM - Greg Shah

> Make TREEVIEW implemented as a layer over the TREELIST. TREEVIEW is a one-column TREELIST with caption display switched off.

My primary concern here is that this will add extra/unnecessary complexity to TREEVIEW. The TREEVIEW is the common use case and it is used in many applications. TREELIST is used much less often (most applications do NOT use it). The addition of complexity in TREEVIEW can have long term costs in the number of bugs and the fragility of the control.

A secondary concern is that this will break or cause problems in custom widgets which extend from the TREEVIEW or use the TREEVIEW as an embedded widget. There is at least one customer-specific widget which does this.

I admit that I have not analyzed the impacts, it is just my intuition that these are areas of concern.

> Internal refactoring: remove usage of Java generics in TREELIST/TREEVIEW implementation classes.
> ...
> Java generics parameters are removed from all related class definitions.

I assume this is limited to just the tree classes. If so, then I am OK with the idea. You will need to put some javadoc in (to all related classes) to explain that these must be avoided.

#92 - 06/18/2021 07:47 AM - Hynek Cihlar

Vladimir Tsichevski wrote:

> 1. The conventional class ancestry is mixed with generics, and this prevents from using the base class instance references instead of concrete classes where possible.

Vladimir, please provide an example which prevents the use of base class.

1. Conventional class inheritance is all we need here, and Java generics need not be used where simple inheritance is enough.

The generics provide extra type safety.

1. Generics are useful if we are writing library for using with any unknown in advance types provided later by user. Our situation is completely different: we are dealing with a closed set of known and closely related classes.

There are many other cases where we do use generics even though we deal with closed set of known classes. For an example the widget tree itself. Again, this adds extra type safety.

#93 - 06/18/2021 07:56 AM - Eugenie Lyzenko

Vladimir,

Please keep in mind in your efforts there is another customer specific external widget that is highly dependent from TREE-LIST. To not to completely stop it's functionality as result of this rework.

#94 - 06/18/2021 08:05 AM - Hynek Cihlar

Vladimir Tsichevski wrote:

> ### Make TREEVIEW implemented as a layer over the TREELIST
>
> **Rationale**
>
> 1. This change automatically solves a number of known TREELIST issues related to first tree list column rendering.

Yes, this is a valid concern. The question is whether such a huge change of design is needed to resolve this simple issue. One solution would be to decouple treelist drawing from treeview altogether. I.e. move all treeview drawing logic in the treeview specific classes and treelist drawing logic in the treelist specific classes.

1. It makes implementation much simpler and code more readable.

When you say current implementation is not simple, do you have any arguments for this or is this your subjective perception? I don't myself find the implementation hard to understand for example. But again this is my subjective perception. I'm not trying to say the code is simple or difficult, I'm only pointing out we should get some evidence to support such a claim - code metrics, developers poll, etc - and do a qualified decision based on such evidence.

I agree with Greg, that merging both implementations in one will introduce its own layer of complexity. Both the original OCX controls have very unique set of features, different look and feel, some differences in event handling. The master implementation would have to contain added layer of parametrization logic. Also both the controls have different set of APIs, while treeview API is more object oriented, treelist is rather flat.

#95 - 06/18/2021 09:18 AM - Vladimir Tsichevski

Sergey Ivanovskiy wrote:

> Vladimir, TreeViewGuiImpl supports the editing of node labels but TreeListGuiImpl doesn't implement this functionality now. Did you take this functionality into account? Does tree list support manual and automatic drag and drop functionalities? I know that one screen of the customer's project uses manual drag and drop functionality. It means that drag and drop listeners are implemented by 4GL code. Now for the web client the dragged node under the mouse pointer isn't detected correctly by the converted 4GL code but the swing client doesn't have this issue #5172-50.

I am not going to remove the tree view classes completely, I propose to change the internal data structures and implementation details, I do not see, how can this prevent the drag-n-drop feature implementation.

Besides, the TREELIST initial prototype supports the node drag-n-drop feature, so pulling this feature up and thus making it available in TREELIST will not hurt.

BTW, I cannot convert TREEVIEW examples currently: conversion produces code that wont compile. I think, this should be fixed.

#96 - 06/18/2021 09:36 AM - Vladimir Tsichevski

Greg Shah wrote:

> > Make TREEVIEW implemented as a layer over the TREELIST. TREEVIEW is a one-column TREELIST with caption display switched off.
>
> My primary concern here is that this will add extra/unnecessary complexity to TREEVIEW. The TREEVIEW is the common use case and it is used in many applications. TREELIST is used much less often (most applications do NOT use it). The addition of complexity in TREEVIEW can have long term costs in the number of bugs and the fragility of the control.

Agreed. But this TREELIST/TREEVIEW module is finite and closed. All bugs should and will be eventually fixed.
And, keeping two independent implementations means we have two bug sources instead of one.

> A secondary concern is that this will break or cause problems in custom widgets which extend from the TREEVIEW or use the TREEVIEW as an embedded widget. There is at least one customer-specific widget which does this.

I think, we will need to check this in any case.

> I admit that I have not analyzed the impacts, it is just my intuition that these are areas of concern.
>
> > Internal refactoring: remove usage of Java generics in TREELIST/TREEVIEW implementation classes.
> > ...
> > Java generics parameters are removed from all related class definitions.
>
> I assume this is limited to just the tree classes. If so, then I am OK with the idea. You will need to put some javadoc in (to all related classes) to explain that these must be avoided.

As to mentioning in Javadoc. I think, this can be noted in history entries only. Not that using generics here should be avoided, they are just are not needed here, so nobody will ever want generics after the change is complete.

#97 - 06/18/2021 10:39 AM - Vladimir Tsichevski

Hynek Cihlar wrote:

> Vladimir Tsichevski wrote:
>
> 1. The conventional class ancestry is mixed with generics, and this prevents from using the base class instance references instead of concrete classes where possible.
>
> Vladimir, please provide an example which prevents the use of base class.

Currently with generics (templates) we effectively have two **different** class trees. They are not interchangeable from the compiler stand point, and that is the whole idea of generics.

Our real situation here we **can** use a single class tree, and the factory pattern to create appropriate implementations. We use the common interfaces only and never bother about the concrete types.

1. Conventional class inheritance is all we need here, and Java generics need not be used where simple inheritance is enough.

> The generics provide extra type safety.

The only places we should bother about the type safety are the factory methods (and since we have only one such method, I think, we could afford the effort :-)). After that done, the type safety is provided by inheritance while we rely on the common API only. Most of the TREELIST/TREEVIEW code already match this criteria, so passing concrete implementation types to them is just a bad programming style.

1. Generics are useful if we are writing library for using with any unknown in advance types provided later by user. Our situation is completely different: we are dealing with a closed set of known and closely related classes.

> There are many other cases where we do use generics even though we deal with closed set of known classes.

And I do not think it is always a good thing. For example, the FontDetails class declaration has generic parameter, which I as a class user can no affect and am never bothering about. But in the code using the class I should always write FontDetails<?> instead of just FontDetails just to make compiler happy. Besides, this breaks the whole type safety idea, I think.

> For an example the widget tree itself. Again, this adds extra type safety.

In case a common API is used the type safety is guaranteed by inheritance mechanism. We neither have to nor not want to bring implementation details to such code.

#98 - 06/18/2021 10:41 AM - Vladimir Tsichevski

Eugenie Lyzenko wrote:

> Vladimir,
>
> Please keep in mind in your efforts there is another customer specific external widget that is highly dependent from TREE-LIST. To not to completely stop it's functionality as result of this rework.

Yes, I am aware of this problem :-(.

#99 - 06/18/2021 11:06 AM - Vladimir Tsichevski

Hynek Cihlar wrote:

> Vladimir Tsichevski wrote:
>
> ## Make TREEVIEW implemented as a layer over the TREELIST
>
> **Rationale**
>
> 1. This change automatically solves a number of known TREELIST issues related to first tree list column rendering.
>
> Yes, this is a valid concern. The question is whether such a huge change of design is needed to resolve this simple issue. One solution would be to decouple treelist drawing from treeview altogether. I.e. move all treeview drawing logic in the treeview specific classes and treelist drawing logic in the treelist specific classes.

Yes, this can be solution. The downside is after we copy/paste, we will have two peaces of code instead one, and from this moment on, we will need to support both.

> 1. It makes implementation much simpler and code more readable.
>
> When you say current implementation is not simple, do you have any arguments for this or is this your subjective perception? I don't myself find the implementation hard to understand for example. But again this is my subjective perception. I'm not trying to say the code is simple or difficult, I'm only pointing out we should get some evidence to support such a claim - code metrics, developers poll, etc - and do a qualified decision based on such evidence.

For me, the following construct is not simple to comprehend:

```
TreeBodyGuiImpl<
      C extends TreeConfig<N>,
      T extends TreeGuiImpl<C, ? extends TreeBodyGuiImpl<C,T,N>, N>,
      N extends TreeNodeEntry>
```

and all these efforts just to discover that just TreeBodyGuiImpl is enough here :-(.

> I agree with Greg, that merging both implementations in one will introduce its own layer of complexity. Both the original OCX controls have very unique set of features, different look and feel, some differences in event handling. The master implementation would have to contain added layer of parametrization logic. Also both the controls have different set of APIs, while treeview API is more object oriented, treelist is rather flat.

I am going mostly to change the way the first column data is handled and the first column is rendered. At the moment, the rendering is done with **the same** method call, so there no difference does exist here so far. I am not going to touch any public APIs.

And, this second refactoring depends on the first one, but not otherwise. We can discuss them independently.

And, I am pretty sure the first generic-related refactoring is possible, because I have done it already, a lot of unnecessary code removed, and everything still works as before. I do not see any possible related problem in the future here.

As to the second refactoring, I am not so sure, so I can try the copy/paste approach you proposed first.

#100 - 06/18/2021 02:50 PM - Hynek Cihlar

Vladimir Tsichevski wrote:

> Hynek Cihlar wrote:
>
> > Vladimir Tsichevski wrote:
> >
> > ## Make TREEVIEW implemented as a layer over the TREELIST
> >
> > **Rationale**
> >
> > 1. This change automatically solves a number of known TREELIST issues related to first tree list column rendering.
>
> Yes, this is a valid concern. The question is whether such a huge change of design is needed to resolve this simple issue. One solution would be to decouple treelist drawing from treeview altogether. I.e. move all treeview drawing logic in the treeview specific classes and treelist drawing logic in the treelist specific classes.

Yes, this can be solution. The downside is after we copy/paste, we will have two peaces of code instead one, and from this moment on, we will need to support both.

> 1. It makes implementation much simpler and code more readable.

> When you say current implementation is not simple, do you have any arguments for this or is this your subjective perception? I don't myself find the implementation hard to understand for example. But again this is my subjective perception. I'm not trying to say the code is simple or difficult, I'm only pointing out we should get some evidence to support such a claim - code metrics, developers poll, etc - and do a qualified decision based on such evidence.

For me, the following construct is not simple to comprehend:

It looks a bit scary, but I think it makes good sense. You allow to parameterize the base class with concrete config, body and node classes. But I'm not against simplifications that don't introduce issues elsewhere.

> I agree with Greg, that merging both implementations in one will introduce its own layer of complexity. Both the original OCX controls have very unique set of features, different look and feel, some differences in event handling. The master implementation would have to contain added layer of parametrization logic. Also both the controls have different set of APIs, while treeview API is more object oriented, treelist is rather flat.

> I am going mostly to change the way the first column data is handled and the first column is rendered. At the moment, the rendering is done with **the same** method call, so there no difference does exist here so far. I am not going to touch any public APIs.

Sounds good.

> As to the second refactoring, I am not so sure, so I can try the copy/paste approach you proposed first.

I didn't mean to do a copy and paste. Any common code can be placed in base or utility classes.

#101 - 06/18/2021 02:52 PM - Hynek Cihlar

Vladimir Tsichevski wrote:

> And, I am pretty sure the first generic-related refactoring is possible, because I have done it already, a lot of unnecessary code removed, and everything still works as before. I do not see any possible related problem in the future here.

Can you share your changes?

#102 - 06/18/2021 03:14 PM - Vladimir Tsichevski

Hynek Cihlar wrote:

> As to the second refactoring, I am not so sure, so I can try the copy/paste approach you proposed first.

I didn't mean to do a copy and paste. Any common code can be placed in base or utility classes.

In this particular case there is too much of context, which is prepared at all levels (the tree level, node level, node cell level), and used everywhere. It is very hard to extract meaningful methods in this situation, too much of parameters should be passed to these methods, it is very hard to provide meaningful explanations to these parameters and the methods themselves.

So flat code looks like a much lesser evil here.

#103 - 06/18/2021 03:16 PM - Vladimir Tsichevski

So, this is the result of the discussion and the current development state:

1. Making TREELIST/TREEVIEW implementation generic-free: **accepted**, done (it works, but need some additional refactoring to introduce inheritance where currently Java cast from base to derived types are used, this will assure the type safety).
2. Implementing TREEVIEW as one-column TREELIST: **declined** (may be temporary, until additional reasons to make this change arise). Solving the first TREELIST column rendering issues by separating the rendering code. I've just done this too.

#104 - 06/18/2021 03:22 PM - Vladimir Tsichevski

Hynek Cihlar wrote:

> Vladimir Tsichevski wrote:
>
> > And, I am pretty sure the first generic-related refactoring is possible, because I have done it already, a lot of unnecessary code removed, and everything still works as before. I do not see any possible related problem in the future here.
>
> Can you share your changes?

There is a problem: I've started this change **after** I've done the part of the work related to other tree issues, so the changes are intermixed.

And this "scrolling" work is not finished yet, so it is not safe to commit anything.

I can create a diff file and post it here though.

#105 - 06/18/2021 03:28 PM - Hynek Cihlar

Vladimir Tsichevski wrote:

> I can create a diff file and post it here though.

Yes, this is what I meant.

#106 - 06/18/2021 03:33 PM - Vladimir Tsichevski

- File 17.06.2021-tree-body-rendering-refactored-and-fixed-missed-files-added.diff added

Hynek Cihlar wrote:

> Vladimir Tsichevski wrote:
>
> > I can create a diff file and post it here though.
>
> Yes, this is what I meant.

Done:
17.06.2021-tree-body-rendering-refactored-and-fixed-missed-files-added.diff

#107 - 06/21/2021 02:13 AM - Hynek Cihlar

Vladimir Tsichevski wrote:

> Hynek Cihlar wrote:
>
> > Vladimir Tsichevski wrote:
> >
> > > I can create a diff file and post it here though.
> >
> > Yes, this is what I meant.
>
> Done:

17.06.2021-tree-body-rendering-refactored-and-fixed-missed-files-added.diff

The generics are gone for the price of having to down cast on multiple places. This was the type safety benefit I was mentioning earlier, dealing with concrete and well known types. Frankly I can't quantify what version is better but my guts incline more towards generics.

But if Greg is fine with the changes, I'm too :-).

#108 - 06/21/2021 07:43 AM - Greg Shah

> Frankly I can't quantify what version is better but my guts incline more towards generics.

Since it is not absolutely clear, let's avoid the generics. To the degree it is needed to help refactor the code, then we are "ahead".

#109 - 06/21/2021 10:20 AM - Vladimir Tsichevski

Hynek Cihlar wrote:

> The generics are gone for the price of having to down cast on multiple places. This was the type safety benefit I was mentioning earlier, dealing with concrete and well known types.

I have mentioned earlier, that this is work-in-progress, and most if not all downcasts must disappear as some additional overloaded functions are introduced.

> Frankly I can't quantify what version is better but my guts incline more towards generics.

As my change shows generics can be just removed in this very case, and this helps solving real issues.

#110 - 06/22/2021 09:57 AM - Vladimir Tsichevski

- File 5118-110.mp4 added

FIXED in rev. 12602.

Node navigation by keyboard works incorrectly. See the attached 5118-110.mp4 video.
In this video I navigate to 9-th node with the DOWN key, then press the UP key once.
The focused changes to the first tree node instead of the previous 8-th node.

The problem origin: the TreeGuiImpl.NodesIterator works incorrectly in some situations when iterating in backward direction.

#111 - 06/23/2021 10:10 AM - Vladimir Tsichevski

- File 5118-111.png added

In OCX the scrollable container wraps the whole tree widget, including the tree caption. In FWD it wraps the tree body only:

This current design also makes it problematic the fixed columns implementation.

#112 - 06/23/2021 10:11 AM - Hynek Cihlar

Vladimir Tsichevski wrote:

> In OCX the scrollable container wraps the whole tree widget, including the tree caption. In FWD it wraps the tree body only:

This came from MS Treeview.

#113 - 06/23/2021 10:20 AM - Vladimir Tsichevski

Hynek Cihlar wrote:

> Vladimir Tsichevski wrote:
>
> > In OCX the scrollable container wraps the whole tree widget, including the tree caption. In FWD it wraps the tree body only:
>
> This came from MS Treeview.

I gather, in TREEVIEW there is no caption, so this problem cannot exist in TREEVIEW.

#114 - 06/23/2021 10:25 AM - Hynek Cihlar

Vladimir Tsichevski wrote:

> Hynek Cihlar wrote:
>
> > Vladimir Tsichevski wrote:
> >
> > > In OCX the scrollable container wraps the whole tree widget, including the tree caption. In FWD it wraps the tree body only:
> >
> > This came from MS Treeview.
>
> I gather, in TREEVIEW there is no caption, so this problem cannot exist in TREEVIEW.

I would keep this as is. From the UX perspective, scrolling only the records makes better sense.

#115 - 06/23/2021 10:55 AM - Vladimir Tsichevski

Hynek Cihlar wrote:

> I would keep this as is. From the UX perspective, scrolling only the records makes better sense.

I am not talking about scrolling the caption vertically along with the tree body. The original OCX does not behaves so.

I mean the caption must be in the same viewport with the body and scrolled horizontally with the body.

I would prefer implement the caption and caption items as part of tree body widget. The rationale:

1. The caption is **not** a plain sum of caption items: the first and the last caption items should be painted a bit differently.
2. The tree body implemented as **one widget**: columns are **not** implemented as widgets. So why should column headers be widgets?
3. The original OCX library is implemented in this way.

The alternative: add a container widget holding the caption and the body. This would make the widget tree a bit more complex.

As to vertical scrolling, the vertical scrolling is done by rows only, and is implemented by drawing only the nodes visible in viewport, so this is not a problem.

#116 - 06/23/2021 11:01 AM - Hynek Cihlar

Vladimir Tsichevski wrote:

> 1. The tree body implemented as **one widget**: columns are **not** implemented as widgets. So why should column headers be widgets?

Because they respond to user input individually. They can be pressed/depressed, moved, resized, etc. There is a lot of user input optimizations implemented at the web gui driver, having the caption items split in multiple widgets allows to utilize these optimizations.

#117 - 06/23/2021 11:48 AM - Vladimir Tsichevski

Hynek Cihlar wrote:

> Vladimir Tsichevski wrote:
>
> > 1. The tree body implemented as **one widget**: columns are **not** implemented as widgets. So why should column headers be widgets?
>
> Because they respond to user input individually. They can be pressed/depressed, moved, resized, etc. There is a lot of user input optimizations implemented at the web gui driver, having the caption items split in multiple widgets allows to utilize these optimizations.

When a caption is dragged, a ghost widget is created to visualize dragging. I do not propose to change that. Are there any other optimizations unrelated to ghost widget?

#118 - 06/23/2021 11:51 AM - Vladimir Tsichevski

Expand/collapse button hit test issue:

In the original OCX, clicking with mouse outside the expand/collapse button does not trigger the action.
In FWD, clicking below or above the button trigger the expand/collapse.

#119 - 06/23/2021 12:36 PM - Hynek Cihlar

Vladimir Tsichevski wrote:

> Hynek Cihlar wrote:
>
> > Vladimir Tsichevski wrote:
> >
> > 1. The tree body implemented as **one widget**: columns are **not** implemented as widgets. So why should column headers be widgets?
> >
> > Because they respond to user input individually. They can be pressed/depressed, moved, resized, etc. There is a lot of user input optimizations implemented at the web gui driver, having the caption items split in multiple widgets allows to utilize these optimizations.
>
> When a caption is dragged, a ghost widget is created to visualize dragging. I do not propose to change that. Are there any other optimizations unrelated to ghost widget?

With individual widgets you get these for free. Selective redraw (only invalidated widgets need to redraw), event handling, z-order, draw output caching, direct manipulation, some input handled in JS driver, visibility, enabled state, layout.

#120 - 06/23/2021 01:04 PM - Vladimir Tsichevski

Hynek Cihlar wrote:

> Selective redraw (only invalidated widgets need to redraw)

Caption items are very simple objects to redraw, the tree body is much more complex, and currently it is completely repainted on every occasion, which is a heavy operation for a large tree. Besides, FWD supports adding dirty areas which should be repainted, so we can use this feature to smart repaint individual items if we ever really need this.

> event handling

Not a big deal here comparing to the event handling in the tree body, which is more complex, and which is monolith.

> z-order

How caption benefit from this?

> draw output caching

Is this used in ordinary caption (not a ghost item) drawing?

> direct manipulation

Is it used here?

> some input handled in JS driver, visibility, enabled state, layout.

Are TREELISTs really benefit from all these?

#121 - 06/23/2021 03:02 PM - Hynek Cihlar

Vladimir Tsichevski wrote:

> Hynek Cihlar wrote:
>
> > Selective redraw (only invalidated widgets need to redraw)
>
> Caption items are very simple objects to redraw, the tree body is much more complex, and currently it is completely repainted on every occasion, which is a heavy operation for a large tree. Besides, FWD supports adding dirty areas which should be repainted, so we can use this feature to smart repaint individual items if we ever really need this.
>
> > event handling
>
> Not a big deal here comparing to the event handling in the tree body, which is more complex, and which is monolith.
>
> > z-order
>
> How caption benefit from this?
>
> > draw output caching
>
> Is this used in ordinary caption (not a ghost item) drawing?

Any widget benefits from this. If a complex widget is well structured it will add to its responsiveness.

> direct manipulation

> Is it used here?

> some input handled in JS driver, visibility, enabled state, layout.

> Are TREELISTs really benefit from all these?

Any input handled already in JS driver will improve UI responsiveness. For the rest of the widget tree features, these may become important as TREELIST receives new UI elements and becomes more complex.

Besides, non-monolith well structured widget allows you to reuse existing widgets. Look how the system dialogs are implemented for example (like the file chooser dialog - FileDialogGuiImpl). It could have been implemented as a monolith, but instead its clever structure of existing widgets (1) shortened its implementation time, (2) benefits from the JS driver optimizations, (3) allows for theming, (4) and benefits from the ongoing improvements of the individual widgets.

#122 - 06/23/2021 05:57 PM - Vladimir Tsichevski

Hynek Cihlar wrote:

> Any widget benefits from this. If a complex widget is well structured it will add to its responsiveness.

To some extent, yes. Note: our complex tree widget is **not** well structured: otherwise tree cells were implemented as widgets. And there is a reason to this: it would be hard to manage this many widgets by the system. In our case the column, row and cell structure is very regular, so it is quite simple to paint the whole widget, and to route user events.

> direct manipulation

> Is it used here?
>
> > some input handled in JS driver, visibility, enabled state, layout.
>
> Are TREELISTs really benefit from all these?
>
> Any input handled already in JS driver will improve UI responsiveness.

Exactly. But AFAIK, currently there is no TREELIST-specific support in JS driver :-(

> For the rest of the widget tree features, these may become important as TREELIST receives new UI elements and becomes more complex.

Which UI elements do we plan to add? Nothing but a cell editor comes to my mind, and yes, this cell editor should be implemented as a widget.

Besides I doubt it will ever happen with TREELIST. TREELIST was created as a replacement for a quite specific customer OCX with some features are quite unusual and hard to explain. It will be quite hard to use it as a general purpose widget in other projects without breaking things.

> Besides, non-monolith well structured widget allows you to reuse existing widgets. Look how the system dialogs are implemented for example (like the file chooser dialog - FileDialogGuiImpl).

This file dialog example is not very relevant. File dialog (and any user-created complex window) is created from universal ready-to-use independent blocks. TREELIST differs from file dialog greatly in that: 1) no ready-to-use blocks can be used to build it from; 2) not part of TREELIST can be used outside of TREELIST.

#123 - 06/24/2021 02:48 AM - Hynek Cihlar

Vladimir Tsichevski wrote:

> Hynek Cihlar wrote:
>
> > Any widget benefits from this. If a complex widget is well structured it will add to its responsiveness.
>
> To some extent, yes. Note: our complex tree widget is **not** well structured: otherwise tree cells were implemented as widgets. And there is a reason to this: it would be hard to manage this many widgets by the system. In our case the column, row and cell structure is very regular, so it is quite simple to paint the whole widget, and to route user events.

As long as the cells don't handle user input (no edit, no focus handling, no events fired) it doesn't make much sense to make them widgets. The title bar is a different case, here widgets do make sense. By well-structured I didn't mean to make everything into widgets uncoditionally.

> > direct manipulation
>
> Is it used here?
>
> > some input handled in JS driver, visibility, enabled state, layout.
>
> Are TREELISTs really benefit from all these?
>
> Any input handled already in JS driver will improve UI responsiveness.
>
> Exactly. But AFAIK, currently there is no TREELIST-specific support in JS driver :-(

Since the title bar is made from individual widgets, JS driver can already provide some optimizations in drawing and user input. If you make the title bar a single monolith widget, then yes, there will be no optimizations from JS at all.

> > For the rest of the widget tree features, these may become important as TREELIST receives new UI elements and becomes more complex.
>
> Which UI elements do we plan to add? Nothing but a cell editor comes to my mind, and yes, this cell editor should be implemented as a widget.
>
> Besides I doubt it will ever happen with TREELIST. TREELIST was created as a replacement for a quite specific customer OCX with some features are quite unusual and hard to explain. It will be quite hard to use it as a general purpose widget in other projects without breaking things.

Actually TREELIST was created to be a generic widget usable to any project. It is already part of our public wiki documentation as one of the extension points to 4GL. But this is irrelevant anyway. Even if it were meant for a single customer, it wouldn't mean we should do bad design choices :-).

> > Besides, non-monolith well structured widget allows you to reuse existing widgets. Look how the system dialogs are implemented for example (like the file chooser dialog - FileDialogGuiImpl).
>
> This file dialog example is not very relevant. File dialog (and any user-created complex window) is created from universal ready-to-use

independent blocks. TREELIST differs from file dialog greatly in that: 1) no ready-to-use blocks can be used to build it from; 2) not part of TREELIST can be used outside of TREELIST.

I disagree here. The title bar item is one such reusable element. It is a panel with a background and border, both subjects to a current theme. With additional features new such elements will be added.

#124 - 06/24/2021 11:23 AM - Vladimir Tsichevski

FIXED in rev. 12602.

Another issue: in OCX, when a tree is sorted or tree sort direction changes, the tree is scrolled vertically, so the node, which was at top remains visible and at the top.
In FWD the tree is not scrolled, so the current view position is lost for the user.

#125 - 06/24/2021 01:00 PM - Vladimir Tsichevski

- File 5118-125.mp4 added

Yet another issue: arrows are not displayed when a column header is dragged: it is possible to change the column order by dragging column header without the arrows ever visible.

This presumably happens when the columns widths differ much.

See the 5118-125.mp4 video.

#126 - 06/30/2021 12:45 PM - Vladimir Tsichevski

FIXED in rev. 12602:

1. #5118-1 Bullet 4.
2. #5118-1 Bullet 8.
3. #5118-29
4. #5118-39
5. #5118-40
6. #5118-65
7. #5118-67
8. #5118-110
9. #5118-124

#127 - 06/30/2021 01:19 PM - Greg Shah

What is the list of remaining issues for this task?

#128 - 06/30/2021 02:17 PM - Vladimir Tsichevski

Greg Shah wrote:

> What is the list of remaining issues for this task?

1. #5118-53 Unimplemented OCX call: GetNodeVisibleInViewPort.
2. #5118-54 In FWD, if a child node is added, the parent node always expands.
3. #5118-55 In FWD, if a child node is added, and a sorting by some column is set, nodes are automatically sorted.
4. #5118-59 headers layout is broken after a cell value changed in a horizontally scrolled tree list
5. #5118-61 Not fully implemented setCellFgColor and setCellBgColor
6. #5118-62 horizontal position of arrows while a column header is dragged is incorrect (shifted right)
7. #5118-68 Tree node background and foreground colors seem to be supported, but setting these does not affect the tree view.
8. #5118-71 The MultiSelect feature (both the getter and the setter) is not currently implemented, but it is used in customer code.
9. #5118-77 Setting image list statement converts incorrectly.
10. #5118-151 Column dragging problems

And also two related issues: #5212 and #5280

#129 - 07/01/2021 04:40 PM - Vladimir Tsichevski

Fixed in 3821c rev. 12624.

The API call EnsureVisible(node) works incorrectly:

1. In FWD the node is forced to be the **top node**, even if there is not enough nodes after this node to fill the view, and the view can be scrolled up to fill the view. In OCX the node is **not** forced to be the top node, and the view described (with empty space below, which can be filled by scrolling up) is illegal in OCX.
2. The **scroll model** is not updated in FWD as expected, vertical scrollbar is not visible, and vertical scrolling is impossible after this point.
3. The node should be **selected** after it is made visible. In FWD this does not happen.

#131 - 07/01/2021 05:05 PM - Vladimir Tsichevski

Fixed in 3821c rev. 12624.

In OCX, in a **single-column tree**, the width of the only column is set to the widget width regardless of the value set by application. Also, the column width cannot be set by dragging the column header boundary (dragging is possible, but no column width change happens in the end).

#132 - 07/02/2021 02:06 PM - Vladimir Tsichevski

#5118-129, #5118-131 fixed in 3821c rev. 12624.

#133 - 07/02/2021 04:39 PM - Hynek Cihlar

Code review 3821c revisions 12599, 12600, 12602, 12624.

Overall this is a good set of changes. The implementation of toString with ToStringHelper is very elegant.

There are many non-functional changes that make review difficult.

TreeList.java is missing history entry.

TreeBodyGuiImpl.config is missing javadoc.

Some of the changes introduce line breaks unnecessarily, the GCD code standards allows 110 chars.

Why are addLast and addFirst removed? They are making the code more readable:

```
-                    nodes.addLast(node);
+                    nodes.add(nodes.size(), node);
```

Instead of introducing BaseConfig in TreeBodyGuiImpl to allow size fields it woud be better to introduce just the size fields in the class. Also dimension should honor width and height, but it does not. So it may happen that dimenion().width != width().

While I like the separation of model in the implemented tree scrolling I don't like that we now have two separate scrolling implementations. Obviously this makes the code more complicated and more expensive to maintain. I don't see a reason why scrolling needed to be reinvented. If it was for the caption, the original mechanism could handle that, too.

TreeBodyGuiImpl.modelChanged is called by the model when it changes, the notification handler however calls set on the same model. Beside its hard to grasp logic, can't this cause an unwanted recursion in some cases?

Instead of

```
+       for (int y = top; y < bottom; y += 2)
+       {
+          gd.drawLine(x, y, x, y);
+       }
```

use GuiDriver.setLineStroke.

This note is just for the record. With the new more complex widgets, the themed drawing could benefit from a litte bit of a structure, instead of putting everything in a single class.

#134 - 07/02/2021 06:09 PM - Vladimir Tsichevski

Hynek Cihlar wrote:

> Code review 3821c revisions 12599, 12600, 12602, 12624.
>
> Overall this is a good set of changes. The implementation of toString with ToStringHelper is very elegant.

I wonder if it is possible to add a meaningful and uniform toString() implementation to most of other FWD classes. The default Object.toString() does not make sense in most cases. It would make the logging and debugging a lot easier.

> There are many non-functional changes that make review difficult.

Sorry for that. I took my chance to make the code cleaner and better as well.

> TreeList.java is missing history entry.

Will fix this.

> TreeBodyGuiImpl.config is missing javadoc.

Hmm... It is not:

```
/** The widget config. The application sets the {@link BaseConfig#widthChars}
 * and {@link BaseConfig#heightChars} while laying out. */
protected BaseConfig config = new BaseConfig();
```

> Some of the changes introduce line breaks unnecessarily, the GCD code standards allows 110 chars.

The 110 characters is absolute maximum, but lines so long is hard to read.

> Why are addLast and addFirst removed? They are making the code more readable:
> [...]

It was an attempt to replace the LinkedList type reference to more generic List type. Now I see it was not a very good idea. So I will restore the original implementation of TreeNodeCollectionResource.node. Thank you for spotting this.

> Instead of introducing BaseConfig in TreeBodyGuiImpl to allow size fields it would be better to introduce just the size fields in the class.

The ScrollableContainer needs to set its widget size. How can I do this in FWD?

> Also dimension should honor width and height, but it does not. So it may happen that dimenion().width != width().

It is a bug. Will change the implementation to the default one.

> While I like the separation of model in the implemented tree scrolling I don't like that we now have two separate scrolling implementations. Obviously this makes the code more complicated and more expensive to maintain. I don't see a reason why scrolling needed to be reinvented.

Because the existing scrolling implementation does not do what is used here. It operates with widgets only, and specifically, with viewport widgets. The scrolled widget is always painted completely. For long and wide trees this is very costly.
Both horizontal and vertical scrolling are managed in a single call, which is IMO not a good thing, since they are in most cases completely independent by their nature. The implementation is very complicated.

This new scrolling is based on classic MVC concept, which works perfectly for this purpose. The model, is clearly and completely separated from views and controllers. All parts implementing views and controllers are completely separated from each other, they see the common model only.

Besides, this very scrolling implementation was already implemented and tested in grid, so this is not a single usage already.

> If it was for the caption, the original mechanism could handle that, too.

No, it was not for the caption.

> TreeBodyGuiImpl.modelChanged is called by the model when it changes, the notification handler however calls set on the same model. Beside its hard to grasp logic, can't this cause an unwanted recursion in some cases?

The logic is simple: as the result of changing the model, the layout may change too, which requires the new model recalculation. So, the recursion is expected and intended. An yes, the implementer is responsible for not to make the recursion infinite.

In our tree example: as the result of the change of the model, scrollbars can be shown and hidden, as the result, the managed widget dimensions will be changes, which will affect the model again.

> Instead of
> [...]
> use GuiDriver.setLineStroke.

I did not know about this option. I used the existing implementation as the base. So, I gather, the existing implementation author neither did know about this option :-).

Will try to re-implement this using the GuiDriver.setLineStroke.

> This note is just for the record. With the new more complex widgets, the themed drawing could benefit from a litte bit of a structure, instead of putting everything in a single class.

In this very case this would double or triple the number of code lines :-(.

#135 - 07/02/2021 10:45 PM - Eugenie Lyzenko

Hynek Cihlar wrote:

> Code review 3821c revisions 12599, 12600, 12602, 12624.
>
> Overall this is a good set of changes. The implementation of toString with ToStringHelper is very elegant.
>
> There are many non-functional changes that make review difficult.
>
> TreeList.java is missing history entry.
>
> TreeBodyGuiImpl.config is missing javadoc.
>
> Some of the changes introduce line breaks unnecessarily, the GCD code standards allows 110 chars.
>
> Why are addLast and addFirst removed? They are making the code more readable:
> [...]
>
> Instead of introducing BaseConfig in TreeBodyGuiImpl to allow size fields it woud be better to introduce just the size fields in the class. Also dimension should honor width and height, but it does not. So it may happen that dimenion().width != width().
>
> While I like the separation of model in the implemented tree scrolling I don't like that we now have two separate scrolling implementations. Obviously this makes the code more complicated and more expensive to maintain. I don't see a reason why scrolling needed to be reinvented. If it was for the caption, the original mechanism could handle that, too.
>
> TreeBodyGuiImpl.modelChanged is called by the model when it changes, the notification handler however calls set on the same model. Beside its hard to grasp logic, can't this cause an unwanted recursion in some cases?
>
> Instead of
> [...]
> use GuiDriver.setLineStroke.
>
> This note is just for the record. With the new more complex widgets, the themed drawing could benefit from a litte bit of a structure, instead of putting everything in a single class.

I have nothing to add.

It is a big and good work.

#136 - 07/05/2021 09:20 AM - Hynek Cihlar

Vladimir Tsichevski wrote:

> Hynek Cihlar wrote:
>
> > Code review 3821c revisions 12599, 12600, 12602, 12624.
> >
> > Some of the changes introduce line breaks unnecessarily, the GCD code standards allows 110 chars.
>
> The 110 characters is absolute maximum, but lines so long is hard to read.

I think we should pick a line length and stick to it. I don't myself mind 110 or 90 or any other meaningful length. It is a bigger issue when the line length varies across the files IMO.

> > Instead of introducing BaseConfig in TreeBodyGuiImpl to allow size fields it would be better to introduce just the size fields in the class.
>
> The ScrollableContainer needs to set its widget size. How can I do this in FWD?

AbstractContainer, which is a base class of ScrollableContainer already declares size field and implements the size methods (like width and @height). So you should not override them yourself.

> > While I like the separation of model in the implemented tree scrolling I don't like that we now have two separate scrolling implementations. Obviously this makes the code more complicated and more expensive to maintain. I don't see a reason why scrolling needed to be reinvented.
>
> Because the existing scrolling implementation does not do what is used here. It operates with widgets only, and specifically, with viewport widgets. The scrolled widget is always painted completely. For long and wide trees this is very costly.
> Both horizontal and vertical scrolling are managed in a single call, which is IMO not a good thing, since they are in most cases completely independent by their nature. The implementation is very complicated.
>
> This new scrolling is based on classic MVC concept, which works perfectly for this purpose. The model, is clearly and completely separated from views and controllers. All parts implementing views and controllers are completely separated from each other, they see the common model only.

I'm sure the original implementation could be improved to fulfill your needs. The other scrolling cases could have benefited from these improvements and we could have end up with a single implementation and less code to maintain. Anyway, we can eventually merge these two implementations in the future.

#137 - 07/05/2021 09:57 AM - Vladimir Tsichevski

Hynek Cihlar wrote:

> Vladimir Tsichevski wrote:
>
> > Hynek Cihlar wrote:
> >
> > > Code review 3821c revisions 12599, 12600, 12602, 12624.
> > >
> > > Some of the changes introduce line breaks unnecessarily, the GCD code standards allows 110 chars.
> >
> > The 110 characters is absolute maximum, but lines so long is hard to read.
>
> I think we should pick a line length and stick to it. I don't myself mind 110 or 90 or any other meaningful length. It is a bigger issue when the line length varies across the files IMO.

Splitting an expression into several lines makes code more readable and easier to debug in some situation. For example:

```
if (nodeValue != null
    && nodeValue.hasChildren
    && nodeValue.expanded != moveForwardOrExpand)
```

is more readable than:

```
if (nodeValue != null && nodeValue.hasChildren && nodeValue.expanded != moveForwardOrExpand)
```

And the first variant allows to execute every sub-expression as a step in debugger, so the user sees clearly which exactly condition was not met.

So, I do not think, we should make any line 110 character-long lines at any cost.

> > Instead of introducing BaseConfig in TreeBodyGuiImpl to allow size fields it would be better to introduce just the size fields in the class.
> >
> > The ScrollableContainer needs to set its widget size. How can I do this in FWD?
>
> AbstractContainer, which is a base class of ScrollableContainer already declares size field and implements the size methods (like width and @height). So you should not override them yourself.

I do not.
We are talking about the TreeBodyGuiImpl here, which is not a Container. I think, some kind of overloaded setSize() method should be declared in the Widget interface.

#138 - 07/05/2021 10:44 AM - Hynek Cihlar

Vladimir Tsichevski wrote:

> Hynek Cihlar wrote:
>
> > Vladimir Tsichevski wrote:
> >
> > > Hynek Cihlar wrote:
> > >
> > > > Code review 3821c revisions 12599, 12600, 12602, 12624.
> > > >
> > > > Some of the changes introduce line breaks unnecessarily, the GCD code standards allows 110 chars.
> > >
> > > The 110 characters is absolute maximum, but lines so long is hard to read.
> >
> > I think we should pick a line length and stick to it. I don't myself mind 110 or 90 or any other meaningful length. It is a bigger issue when the line length varies across the files IMO.
>
> Splitting an expression into several lines makes code more readable and easier to debug in some situation. For example:
> [...]
> is more readable than:
> [...]
>
> And the first variant allows to execute every sub-expression as a step in debugger, so the user sees clearly which exactly condition was not met.
>
> So, I do not think, we should make any line 110 character-long lines at any cost.

I have been referring to javadocs.

> > > > Instead of introducing BaseConfig in TreeBodyGuiImpl to allow size fields it would be better to introduce just the size fields in the class.
> > >
> > > The ScrollableContainer needs to set its widget size. How can I do this in FWD?
> >
> > AbstractContainer, which is a base class of ScrollableContainer already declares size field and implements the size methods (like width and @height). So you should not override them yourself.
>
> I do not.
> We are talking about the TreeBodyGuiImpl here, which is not a Container. I think, some kind of overloaded setSize() method should be declared in the Widget interface.

For the TreeBodyGuiImpl, introduce size field of type Dimension and override dimension, width and height.

#139 - 07/05/2021 11:29 AM - Vladimir Tsichevski

Hynek Cihlar wrote:

> We are talking about the TreeBodyGuiImpl here, which is not a Container. I think, some kind of overloaded setSize() method should be declared in the Widget interface.

For the TreeBodyGuiImpl, introduce size field of type Dimension and override dimension, width and height.

This will not help. The ScrollableContainer needs a way to set widget size of **any** kind widget. I used BaseConfig with its fields just because it is more or less standard way to hold widget dimensions for non-container widgets. IMO, setting fields directly in config is ugly, but is used everywhere across FWD. Providing an abstract setSize method (or, better two setWidth and setHeight methods) in Widget interface would solve this problem once and for all.

#140 - 07/05/2021 12:06 PM - Hynek Cihlar

Vladimir Tsichevski wrote:

> Hynek Cihlar wrote:
>
> > We are talking about the TreeBodyGuiImpl here, which is not a Container. I think, some kind of overloaded setSize() method should be declared in the Widget interface.
>
> For the TreeBodyGuiImpl, introduce size field of type Dimension and override dimension, width and height.
>
> This will not help. The ScrollableContainer needs a way to set widget size of **any** kind widget. I used BaseConfig with its fields just because it is more or less standard way to hold widget dimensions for non-container widgets. IMO, setting fields directly in config is ugly, but is used everywhere across FWD. Providing an abstract setSize method (or, better two setWidth and setHeight methods) in Widget interface would solve this problem once and for all.

The size of non-container widgets is not meant to be assigned externally, some widgets calculate their size based on the font size and layout settings. Either make the body a container (which I think makes more sense as the in-place editors should be placed in the body and not the tree widget itself) or make the body always inherit the parent's size (you will have to deal with the way your scrolling container is structured). Assigning config fields from other classes is not something we should do.

#141 - 07/06/2021 05:53 PM - Vladimir Tsichevski

Hynek Cihlar wrote:

> For the TreeBodyGuiImpl, introduce size field of type Dimension and override dimension, width and height.

I've implemented a more correct solution to the problem. What we are really doing here is setting the controlled widget viewport size.
So I've implemented the IViewport interface with methods for setting viewport width and height. The ScrollableContainer tests if the controlled widget implements this interface, and if yes used it.

Otherwise, the old (not correct really) implementation is used for containers and other widgets. I decided to keep this old implementation for a while until the PlanBoard and dxgrid are updated to use the new method too.

See the 12639 commit.

#142 - 07/07/2021 04:25 AM - Hynek Cihlar

Vladimir Tsichevski wrote:

> See the 12639 commit.

The changes are good. Just please remove the 'I' from interface names. We don't want the .NET convention in our sources.

#143 - 07/07/2021 04:34 AM - Vladimir Tsichevski

Hynek Cihlar wrote:

> Vladimir Tsichevski wrote:
>
> > See the 12639 commit.
>
> The changes are good. Just please remove the 'I' from interface names. We don't want the .NET convention in our sources.

The name would clash with an existing widget name.

#144 - 07/07/2021 06:00 AM - Hynek Cihlar

Vladimir Tsichevski wrote:

> Hynek Cihlar wrote:
>
> > Vladimir Tsichevski wrote:
> >
> > > See the 12639 commit.
> >
> > The changes are good. Just please remove the 'I' from interface names. We don't want the .NET convention in our sources.
>
> The name would clash with an existing widget name.

Pick a different name :-).

#145 - 07/07/2021 02:04 PM - Eugenie Lyzenko

Vladimir,

We have the customer specific code issues for recent changes. See another specific task.

#146 - 07/08/2021 12:58 PM - Vladimir Tsichevski

Hynek Cihlar wrote:

> Vladimir Tsichevski wrote:
>
> > See the 12639 commit.
>
> The changes are good. Just please remove the 'I' from interface names. We don't want the .NET convention in our sources.

Done in rev. 12647.

#147 - 07/08/2021 01:44 PM - Hynek Cihlar

Vladimir Tsichevski wrote:

> Hynek Cihlar wrote:
>
> > Vladimir Tsichevski wrote:
> >
> > See the 12639 commit.
>
> The changes are good. Just please remove the 'I' from interface names. We don't want the .NET convention in our sources.

Done in rev. 12647.

Also please rename I2DScrollable and any other I-interfaces in the related changes.

#148 - 07/08/2021 04:04 PM - Vladimir Tsichevski

Fixed menu showing issue in 3821c rev. 12650. Do nothing if already show popup menu showing attempted.

#149 - 07/08/2021 04:25 PM - Vladimir Tsichevski

Hynek Cihlar wrote:

> Also please rename I2DScrollable and any other I-interfaces in the related changes.

Done in 12651.

#150 - 07/09/2021 03:29 AM - Hynek Cihlar

Vladimir Tsichevski wrote:

> Hynek Cihlar wrote:

> Also please rename I2DScrollable and any other I-interfaces in the related changes.

Done in 12651.

The changes are good. I have no more objections to the treelist changes above.

#151 - 07/09/2021 08:46 AM - Vladimir Tsichevski

FIXED Another differences:

1. In FWD column dragging start right at the first mouse move event after the mouse button was pressed. In original OCX there is a 5 pixel margin: the drag mode starts only after mouse was dragged at least 5 pixel in any direction from the mouse press position. Otherwise the mouse click actions are triggered.
2. In FWD column reordering is always triggered after the drag operation is complete (event if the mouse was moved only one pixel). In original OCX the column is moved only if the ghost widget was dragged to the target column position. The destination position is displayed with the arrows, and arrows always point to the target column. In FWD the later is not always the case.

#153 - 07/13/2021 02:01 PM - Vladimir Tsichevski

Vladimir Tsichevski wrote:

> Another differences:
>
> 1. In FWD column dragging start right at the first mouse move event after the mouse button was pressed. In original OCX there is a 5 pixel margin: the drag mode starts only after mouse was dragged at least 5 pixel in any direction from the mouse press position. Otherwise the mouse click actions are triggered.
> 2. In FWD column reordering is always triggered after the drag operation is complete (event if the mouse was moved only one pixel). In original OCX the column is moved only if the ghost widget was dragged to the target column position. The destination position is displayed with the arrows, and arrows always points to the target column. In FWD the later is not always the case.

Fixed in 3821c rev. 12674.

#154 - 08/18/2021 05:12 PM - Vladimir Tsichevski

The API call tree:DeleteChildNodes(nodeId) is not currently implemented, but is is used in customer code.

#155 - 08/18/2021 07:09 PM - Vladimir Tsichevski

Vladimir Tsichevski wrote:

> The API call tree:DeleteChildNodes(nodeId) is not currently implemented, but is is used in customer code.

Fixed in 3821c rev. 12838. Please, review.

#156 - 08/19/2021 07:42 AM - Vladimir Tsichevski

DeleteOnCollapse mode issue:

In FWD, in DeleteOnCollapse mode the collapsed node subtree is not removed, if the node was collapsed when the user presses the collapse button on client.

#157 - 08/19/2021 07:49 AM - Vladimir Tsichevski

Mouse event (OnMouseLeftUp, OnMouseLeftDown, OnMouseRightUp) triggers issue: all 3 callback arguments passed to 4gl are always zero.

#158 - 08/19/2021 06:55 PM - Vladimir Tsichevski

The hasChildren (branch or leaf) node attribute issue:

1. When a child is added to a node, FWD sets the hasChildren flag to true (correct, matches the OCX behavior).
2. When the last node child is removed from a node, FWD sets the hasChildren flag to false (**incorrect**, does **not** match the OCX behavior).

This feature allows the expand button to be displayed even for nodes having no children. In such nodes children can be created on-demand then a tree node is being expanded.

#160 - 08/20/2021 10:40 AM - Vladimir Tsichevski

Vladimir Tsichevski wrote:

> The hasChildren (branch or leaf) node attribute issue:
>
> 1. When a child is added to a node, FWD sets the hasChildren flag to true (correct, matches the OCX behavior).
> 2. When the last node child is removed from a node, FWD sets the hasChildren flag to false (**incorrect**, does **not** match the OCX behavior).

Fixed in 3821c rev. 12846.

#161 - 08/26/2021 12:47 PM - Vladimir Tsichevski

- Related to Bug #5622: TREEVIEW widget issues added

#163 - 11/17/2021 01:49 PM - Vladimir Tsichevski

OnCollapsing trigger implementation: in FWD there is an incomplete support for this event: corresponding event ID constant is allocated, and the event is triggered.

But this event is not supported by conversion, so it never reaches the intended 4gl procedure.

Indeed, there is no such event in original OCX by obvious reason: unlike the OnExpanding event, there is no need to let the application to confirm node collapsing.

So, this wrongly implemented support must be completely removed from FWD.

#164 - 11/17/2021 02:30 PM - Hynek Cihlar

Vladimir Tsichevski wrote:

> Indeed, there is no such event in original OCX by obvious reason: unlike the OnExpanding event, there is no need to let the application to confirm node collapsing.

Why is this obvious? Why should not be the app allowed to prevent collapsing? If I remember right, the collapsing event was introduced for TREEVIEW widget. IMHO it will be useful for TREELIST, too.

#165 - 11/17/2021 02:44 PM - Vladimir Tsichevski

Hynek Cihlar wrote:

> Vladimir Tsichevski wrote:
>
> > Indeed, there is no such event in original OCX by obvious reason: unlike the OnExpanding event, there is no need to let the application to confirm node collapsing.
>
> Why is this obvious? Why should not be the app allowed to prevent collapsing? If I remember right, the collapsing event was introduced for TREEVIEW widget. IMHO it will be useful for TREELIST, too.

The OnExpanding event allows the application to see if there are child nodes in the model, and, if yes, build the subtree and return true ti allow expansion, otherwise, do nothing and return false to prevent the expansion an generation of the OnExpanded event.

The OnCollapsing event does not exists. Neither in the TREELIST OCX, nor in TREEVIEW.

Here is the full list of TreeView OCX events:

1. BeforeLabelEdit(short * Cancel)
2. AfterLabelEdit(short * Cancel, BSTR * NewString)
3. Collapse(LPDISPATCH Node)

4. Expand(LPDISPATCH Node)
5. NodeClick(LPDISPATCH Node)
6. KeyDown(short * KeyCode, short Shift)
7. KeyUp(short * KeyCode, short Shift)
8. KeyPress(short * KeyAscii)
9. MouseDown(short Button, short Shift, long x, long y)
10. MouseMove(short Button, short Shift, long x, long y)
11. MouseUp(short Button, short Shift, long x, long y)
12. Click()
13. DblClick()
14. NodeCheck(LPDISPATCH Node)
15. OLEStartDrag(LPDISPATCH * Data, long * AllowedEffects)
16. OLEGiveFeedback(long * Effect, BOOL * DefaultCursors)
17. OLESetData(LPDISPATCH * Data, short * DataFormat)
18. OLECompleteDrag(long * Effect)
19. OLEDragOver(LPDISPATCH * Data, long * Effect, short * Button, short * Shift, float * x, float * y, short * State)
20. OLEDragDrop(LPDISPATCH * Data, long * Effect, short * Button, short * Shift, float * x, float * y)

#166 - 11/17/2021 02:48 PM - Hynek Cihlar

Vladimir Tsichevski wrote:

> Hynek Cihlar wrote:
>
> > Vladimir Tsichevski wrote:
> >
> > > Indeed, there is no such event in original OCX by obvious reason: unlike the OnExpanding event, there is no need to let the application to confirm node collapsing.
> >
> > Why is this obvious? Why should not be the app allowed to prevent collapsing? If I remember right, the collapsing event was introduced for TREEVIEW widget. IMHO it will be useful for TREELIST, too.
>
> The OnExpanding event allows the application to see if there are child nodes in the model, and, if yes, build the subtree and return true ti allow expansion, otherwise, do nothing and return false to prevent the expansion an generation of the OnExpanded event.
>
> The OnCollapsing event does not exists. Neither in the TREELIST OCX, nor in TREEVIEW.
>
> Here is the full list of TreeView OCX events:

I see, in that case it does make sense to get rid of it from FWD.

#167 - 11/23/2021 05:06 PM - Vladimir Tsichevski

Hynek Cihlar wrote:

> Vladimir Tsichevski wrote:
>
> > Hynek Cihlar wrote:
> >
> > > Vladimir Tsichevski wrote:
> > >
> > > > Indeed, there is no such event in original OCX by obvious reason: unlike the OnExpanding event, there is no need to let the application to confirm node collapsing.
> > >
> > > Why is this obvious? Why should not be the app allowed to prevent collapsing? If I remember right, the collapsing event was introduced for TREEVIEW widget. IMHO it will be useful for TREELIST, too.
> >
> > The OnExpanding event allows the application to see if there are child nodes in the model, and, if yes, build the subtree and return true ti allow expansion, otherwise, do nothing and return false to prevent the expansion an generation of the OnExpanded event.
> >
> > The OnCollapsing event does not exists. Neither in the TREELIST OCX, nor in TREEVIEW.
> >
> > Here is the full list of TreeView OCX events:
>
> I see, in that case it does make sense to get rid of it from FWD.

Done in 3821c Revision 13186 as part of the fix for #5823.

#168 - 11/24/2021 05:10 AM - Hynek Cihlar

Vladimir Tsichevski wrote:

> Done in 3821c Revision 13186 as part of the fix for #5823.

Code review 3821c revision 13186. The changes are good.

#171 - 12/21/2021 07:40 AM - Vladimir Tsichevski

- File 5118-treelist-focused-OE.png added

- File 5118-treelist-unfocused-OE.png added

When widget is focused, the selected row must render differently

In original OCX, a dotted rectangle is drawn around the selected row when the widget is focused:

When unfocused:

When focused:

#174 - 02/04/2022 01:00 PM - Vladimir Tsichevski

New issue related to #5118-61.

Current management of **node colors** is incorrect in TREELIST. It must match that in original vmacmtree:

In original vmacmtree node *cell* background and foreground (text) colors are stored for each cell (this feature is correctly implemented in FWD currently).
Along with the color the *override* flag is stored to mark the color is overridden for this very cell (this also works OK in FWD, using null values as "not overridden".

In vmacmtree there is **no field for storing node background and foreground colors**, and the **first node cell colors** are used instead (this is **not so** in FWD currently).

The node colors related functions must work as follows:

GetNodeBGColor(node)

If the cell background color was overridden for the first node cell, then return the overridden color value, otherwise return white color system value.

SetNodeBGColor(node)

Override the background color for all cell values.

GetNodeFGColor(node)

If the cell foreground color was overridden for the first node cell, then return the overridden color value, otherwise return white color system value.

SetNodeFGColor(node)

Override the foreground color for all cell values.

Currently the node colors are managed by methods in TreeWidgetBase both for TREEVIEW and TREELIST. It is necessary to push down these methods along with the fields explicitly storing color values, and provide different implementations.

#175 - 02/04/2022 02:34 PM - Vladimir Tsichevski

Vladimir Tsichevski wrote:

> New issue related to #5118-61.

These and #5118-61 must be fixed in 3821c rev. 13480.

#176 - 07/07/2022 05:56 AM - Vladimir Tsichevski

In original OCX, an attempt to remove the hidden tree root (the node with index 0 and empty key) is silently ignored.
In FWD such attempt crashes the client with AssertionError.

The problematic situation can be reproduced in one of the customer application.

#177 - 07/07/2022 06:24 AM - Vladimir Tsichevski

Vladimir Tsichevski wrote:

> In original OCX, an attempt to remove the hidden tree root (the node with index 0 and empty key) is silently ignored.
> In FWD such attempt crashes the client with AssertionError.
>
> The problematic situation can be reproduced in one of the customer application.

Fixed in 3821c rev 14047.

#178 - 07/19/2022 10:44 AM - Hynek Cihlar

Vladimir Tsichevski wrote:

> Vladimir Tsichevski wrote:
>
> > In original OCX, an attempt to remove the hidden tree root (the node with index 0 and empty key) is silently ignored.
> > In FWD such attempt crashes the client with AssertionError.
> >
> > The problematic situation can be reproduced in one of the customer application.
>
> Fixed in 3821c rev 14047.

The change is good.

Files

5118-unknown-caption-value.png	16.2 KB	03/23/2021	Vladimir Tsichevski
5118-34.mp4	393 KB	05/19/2021	Vladimir Tsichevski
treelist-oe.png	3.31 KB	05/21/2021	Vladimir Tsichevski
treelist-fwd.png	4.51 KB	05/21/2021	Vladimir Tsichevski
WrongHorizontalScrollbar.mp4	61.1 KB	05/21/2021	Vladimir Tsichevski
WrongVerticalScrollbar.mp4	61.9 KB	05/21/2021	Vladimir Tsichevski
WrongHorizontalScrollbar2.mp4	213 KB	05/21/2021	Vladimir Tsichevski
5115-narrow-column-OE.png	23.2 KB	05/24/2021	Vladimir Tsichevski
5115-narrow-column-FWD.png	24.2 KB	05/24/2021	Vladimir Tsichevski
5118-layout-before-cell-value-changed.png	10.3 KB	05/26/2021	Vladimir Tsichevski
5118-layout-broken-after-cell-value-changed.png	9.13 KB	05/26/2021	Vladimir Tsichevski
5118-oe-column-colors.png	9.2 KB	05/26/2021	Vladimir Tsichevski
5118-62.png	4.36 KB	05/27/2021	Vladimir Tsichevski
5118-67-OE.png	10.2 KB	05/27/2021	Vladimir Tsichevski
5118-67-FWD.png	15.4 KB	05/27/2021	Vladimir Tsichevski
5118-bad-header-text-rendering.mp4	384 KB	06/08/2021	Vladimir Tsichevski
17.06.2021-tree-body-rendering-refactored-and-fixed-missed-files-added.diff	288 KB	06/18/2021	Vladimir Tsichevski
5118-110.mp4	284 KB	06/22/2021	Vladimir Tsichevski
5118-111.png	2.82 KB	06/23/2021	Vladimir Tsichevski
5118-125.mp4	106 KB	06/24/2021	Vladimir Tsichevski
5118-treelist-focused-OE.png	1.68 KB	12/21/2021	Vladimir Tsichevski
5118-treelist-unfocused-OE.png	1.84 KB	12/21/2021	Vladimir Tsichevski