



GTK Development Blog

All things GTK

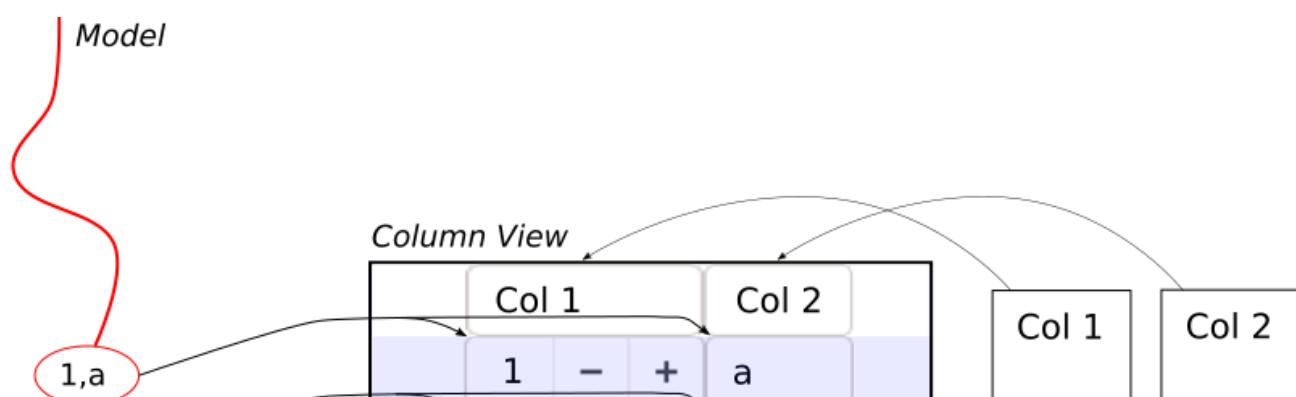
GtkColumnView

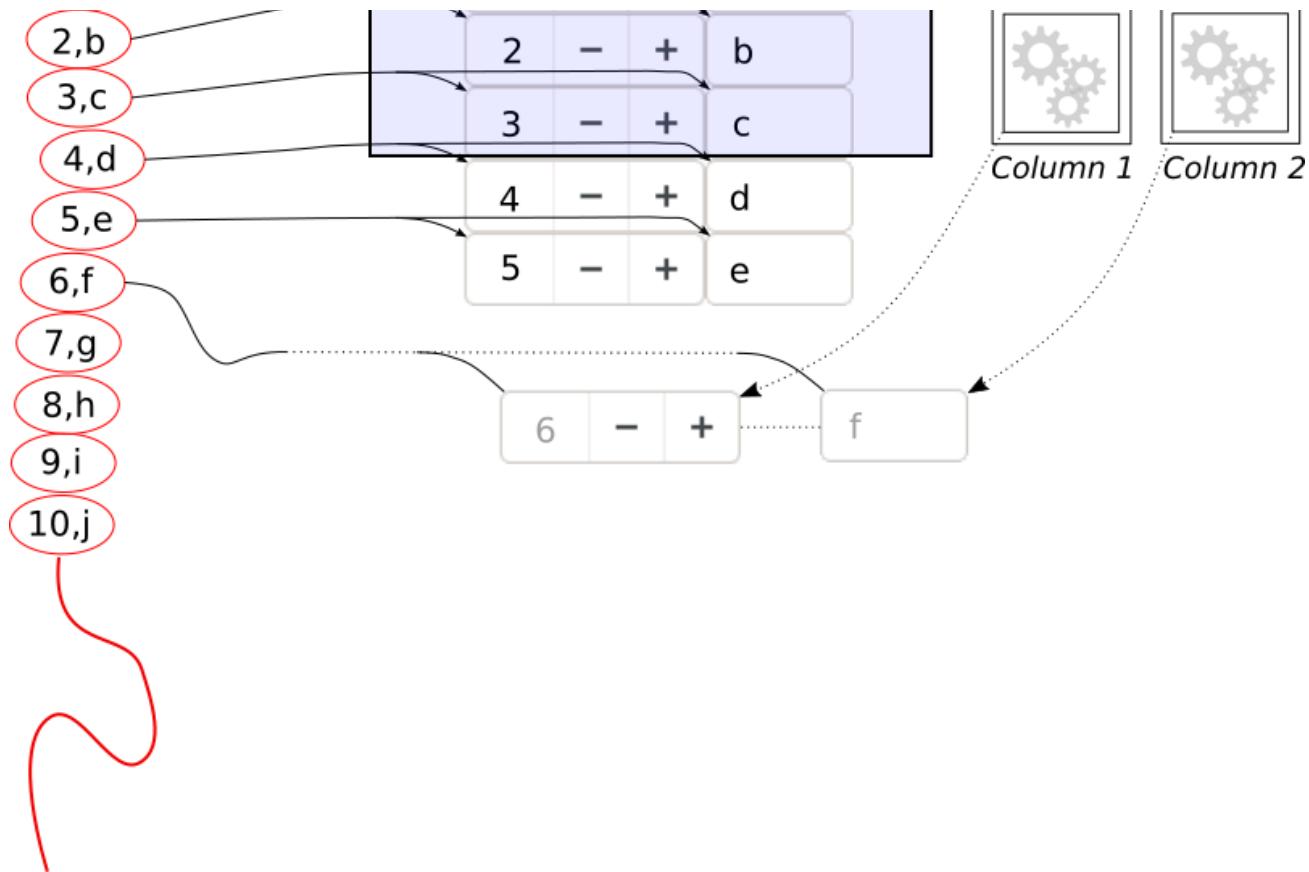
One thing that I left unfinished in my recent series on [list views](#) and [models](#) in GTK 4 is a detailed look at GtkColumnView. This will easily be the most complicated part of the series. We are entering into the heartland of GtkTreeView — anything aiming to replace most its features will be a complicated beast.

Overview

As we did for GtkListView, we'll start with a high-level overview and with a picture.

If you look back at the [listview](#) picture, you'll remember that we use a list item factory to create a widget for each item in our model that needs to be displayed.





In a column view, we need multiple widgets for each item—one for each column. The way we do this is by giving each column its own list item factory. Whenever we need to display a new item, we combine the widgets from each columns factory into a row for the new item.

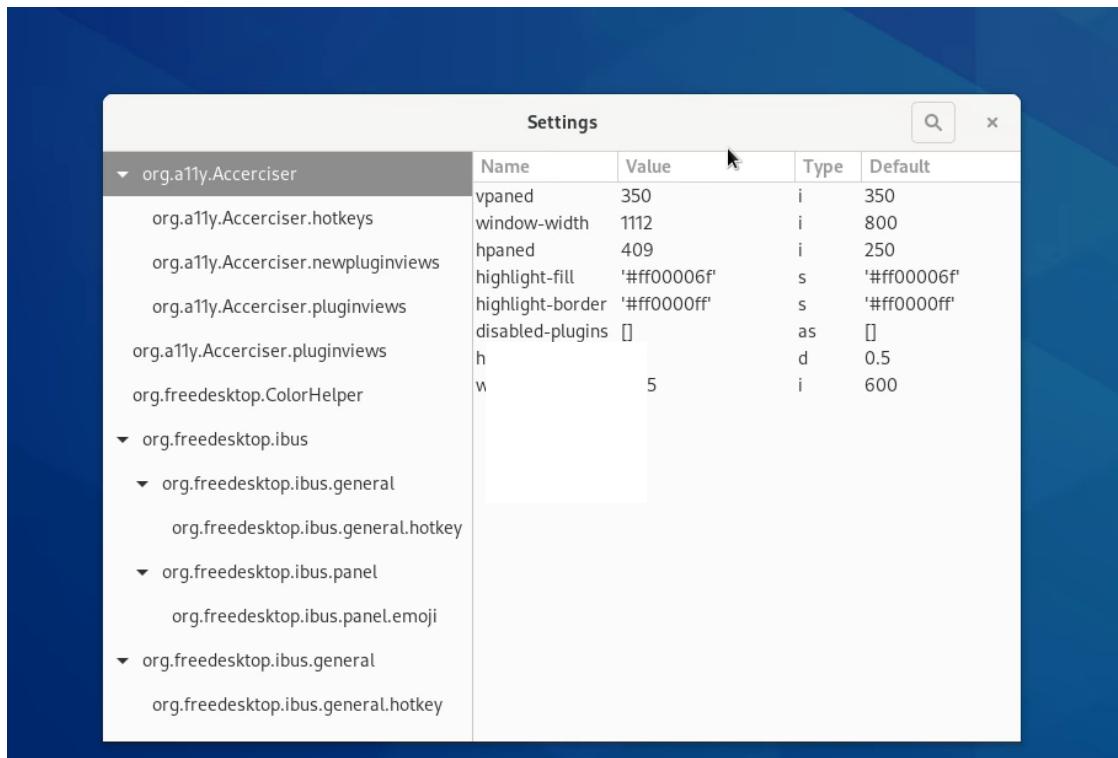
Internally, the column view is actually using a list view to hold the rows. This is nice in that all the things I explained in the previous post about item reuse and about how to use list item factories apply just the same.

Of course, some things *are* different. For example, the column view has organize the size allocation so that the widgets in all rows line up to form proper columns.

Note: Just like GtkListView, the colum view only creates widgets for the segment of the model that is currently in view, so it shares the *vertical* scalability. The same is not true in the *horizontal* direction—every row is fully populated with a widget for each column, even if they are out of view to the left or right. So if you add lots of columns, things will get slow.

Titles, and other complications

The column objects contain other data as well, such as titles. The column view is using those to display a header for each column. If the column view is marked as reorderable, you can rearrange the columns by drag-and-drop of the the header widgets. And if the columns are marked as resizable, you can drag the border between two columns to resize them.



00:00

00:09

If you payed attention, you may now wonder how this resizing goes together with the fact that the cells in the rows can be arbitrary widgets which expect to have at least their minimum size available for drawing their content. The answer is that we are using another new feature of the GTK 4 rendering machinery: Widgets can control how drawing outside their boundaries (by child widgets) is treated, with

```
gtk_widget_set_overflow (cell, GTK_OVERFLOW_HIDDEN)
```

Sorting, selections, and the quest for treeview parity

Since we want to match GtkTreeview, feature-wise, we are not done yet. Another thing that users like to do in tree views is to click on headers, to sort the content by that column. GtkColumnView headers allow this, too.

You may remember from the last post that sorting is done by wrapping your data in a GtkSortListModel, and giving it a suitable sorter object. Since we want to have a different sort order, depending on what column header you clicked, we give each column its own sorter, which you can set with

```
gtk_column_view_column_set_sorter (column, sorter)
```

But how do we get the right sorter from the column you just clicked, and attach it to the sort model? Keep in mind that the sort model is *not* going to be the outmost model that we pass to the column view, since that is always a selection model, so the column view can't just switch the sorter on the sort list model on its own.

The solution we've come up with is to make the column view provide a sorter that internally uses the column sorters, with

```
gtk_column_view_get_sorter (view)
```

You can give this sorter to your sort model once, when you set up your model, and then things will automatically update when the user clicks on column headers to activate different column sorters.

This sounds complicated, but it works surprisingly well. A nice benefit of this approach is that we can actually sort by more than one column at a time—since we have all the column sorters available, and we know which one you clicked last.

Selection handling is easy, by comparison. It works just the same as it does in GtkListView.

Summary

GtkColumnView is a complex widget, but I hope this series of posts will make it a little easier to start using it.



mclasen / September 21, 2020 / Uncategorized / gtk4, guides, in depth, lists

[GTK Development Blog](#) / Proudly powered by [WordPress](#)