20th April 2009

Creating GtkTreeView with Glade-3, part 1

In today's bit, I'll show you how to create tree view and underlying model completely from glade GUI builder.

Contents:

Creating tree view

For this tutorial, you'll need glade >= 3.6, since creating non-widget objects is not possible in previous versions.

I would also recommend you to read GtkTreeView tutorial [http://scentric.net/tutorial/treeview-tutorial.html] and Micah Carrick's glade tutorial [http://www.micahcarrick.com/12-24-2007/gtk-glade-tutorial-part-1.html] before trying to follow this one, since I won't be discussing underlying principles much.

Do we start glade now? No, not yet. We need to create an outline of our application. We'll create simple application with GtkTreeView. Tree view will have 3 columns:

1. "Image" column that will display image and it's name
2. "Penetration" column that will display progress bar
3. "Description" column with some text

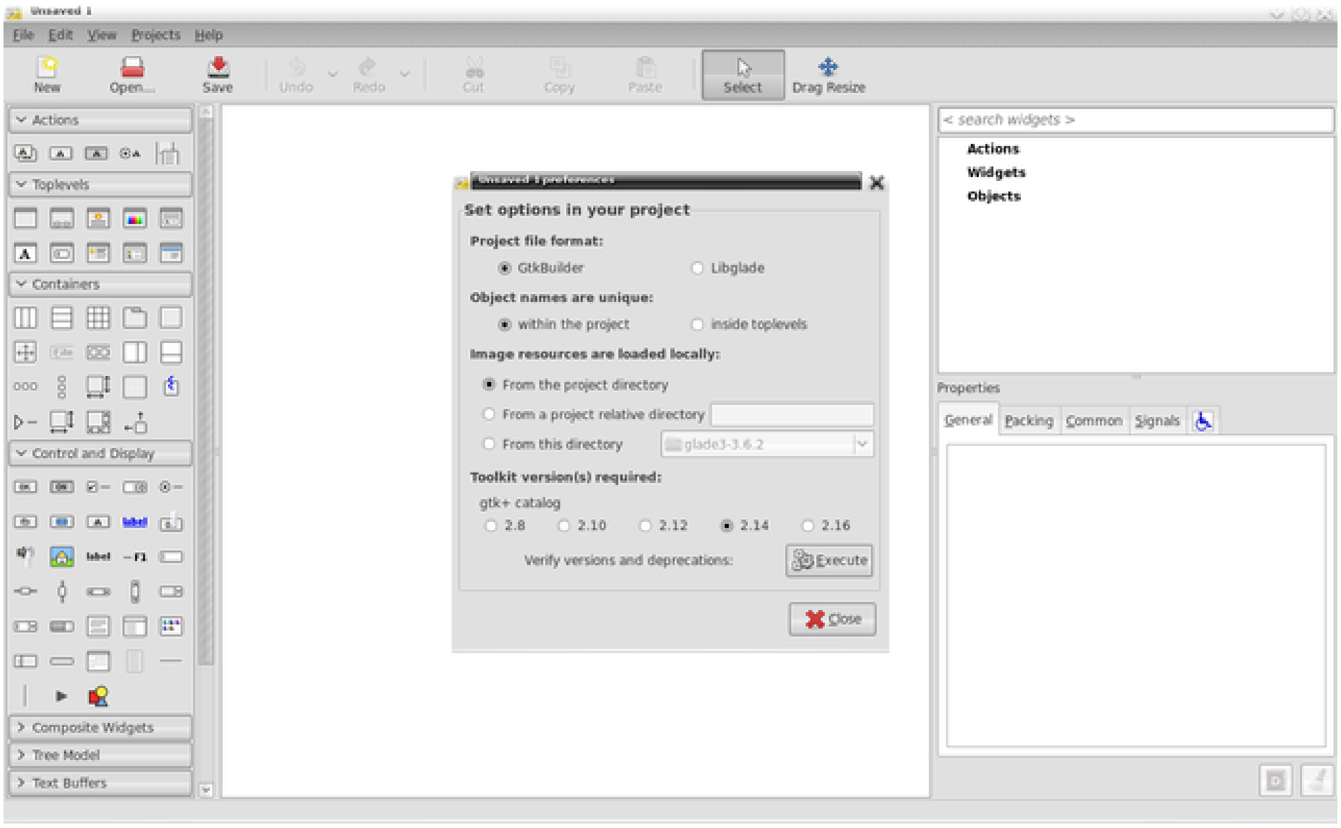
If you read the tree view tutorial, you know that when creating tree view, we need three components: model (store for our data), view (tree view columns and cell renderers) and controller (GtkTreeView). And according to our plan, we'll need three tree view columns and four cell renderers:

1. pixbuf cell renderer for displaying image (in first column)
2. text cell renderer for displaying image name (in first column)
3. progress cell renderer for progress bar and (in second column)
4. another text cell renderer for description (in third column)

We'll place data for those renderers inside GtkListStore with three columns: two text columns for stock ids and descriptions and one numeric column for progress values.

Now we can (finally;) start glade.

After startup, a small dialog ask us about some project properties. Most of the options are fine default, but just make sure you select proper gtk version or GtkBuilder will moan badly when trying to create GUI if your version is less that the one specified in builder file. You've been warned;)



Unsaved 1 preferences

Set options in your project

Project file format: [✓ GtkBuilder] Libglade

Object names are unique: [✓ within the project] inside toplevels

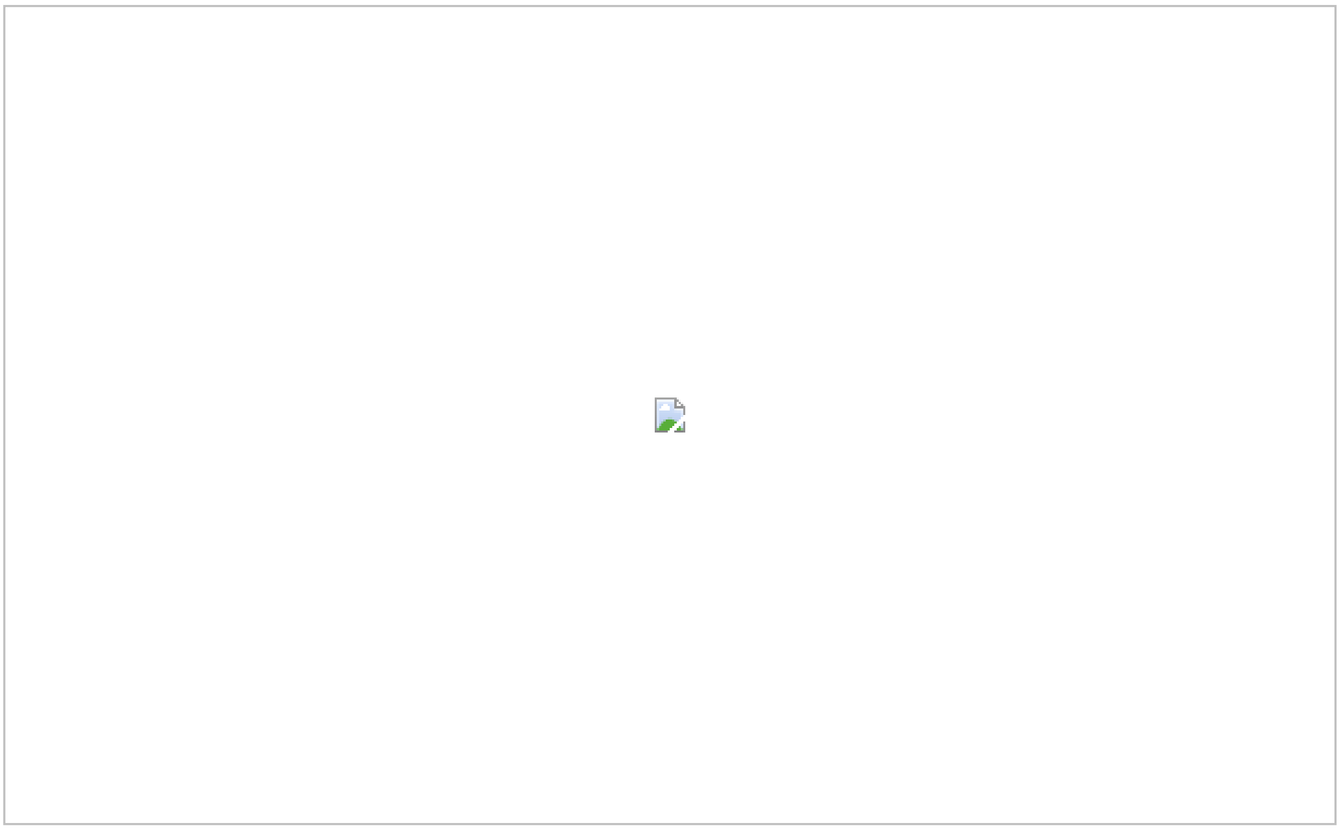
Image resources are loaded locally:
[✓ From the project directory]
From a project relative directory
From this directory glade3-3.6.2

Toolkit version(s) required:
gtk+ catalog 2.8 2.10 2.12 [✓ 2.14] 2.16

Verify versions and deprecations: Execute

Close

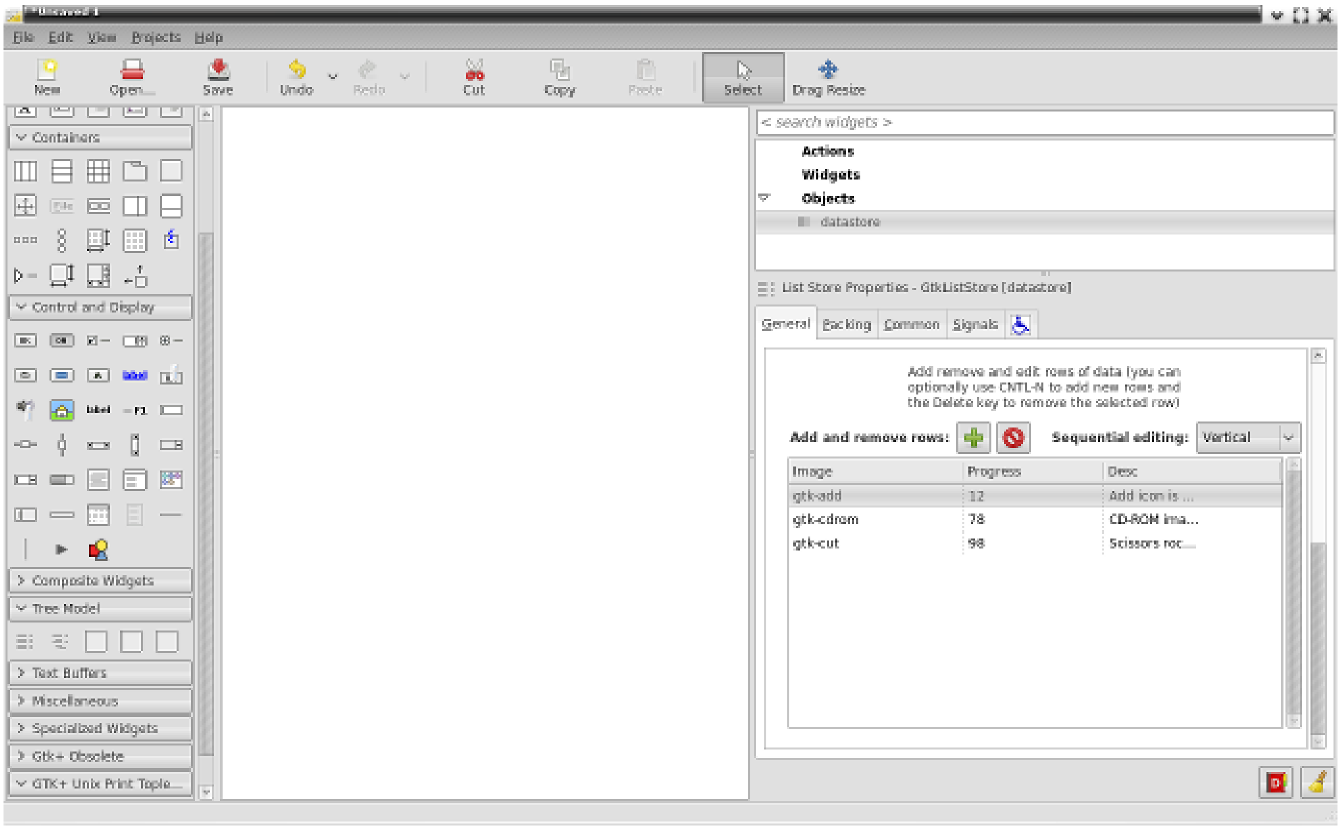
[http://img209.imageshack.us/img209/1888/20090419212310.png] We'll first create out model and populate it with data. Open the widget gallery and click Tree Model -> List Store. Under "Objects", we see our newly created GtkListStore. Let's modify it a bit by changing it's name to "datastore" and adding three columns:



[http://img259.imageshack.us/img259/7431/20090419212530.png] If you ever manually created GtkListStore, you may wonder where all of the G_TYPE_* macros go in column definitions. In glade, you can simply specify type without the macros. So GDK_TYPE_PIXBUF is in glade simply specified as GdkPixbuf, G_TYPE_STRING as gchararray, G_TYPE_INT as gint ... And you don't event need to know their exact name by heart, since glade offers helpful popup.

Now we'll scroll down a bit add some data into our model. Stock item ids are taken from here [http://library.gnome.org/devel/gtk/stable/gtk-Stock-Items.html] . And since glade hides some of the descriptions, I wrote down my original table here too.

```
.-----------+-----------+----------------------------------.
| Image     | Progress  | Desc                             |
+-----------+-----------+----------------------------------+
| gtk-add   |    12     | Add icon is bad at penetrating.  |
| gtk-cdrom |    78     | CD-ROM image fares pretty well.  |
| gtk-cut   |    98     | Scissors rock!!                  |
`-----------+-----------+----------------------------------'
```



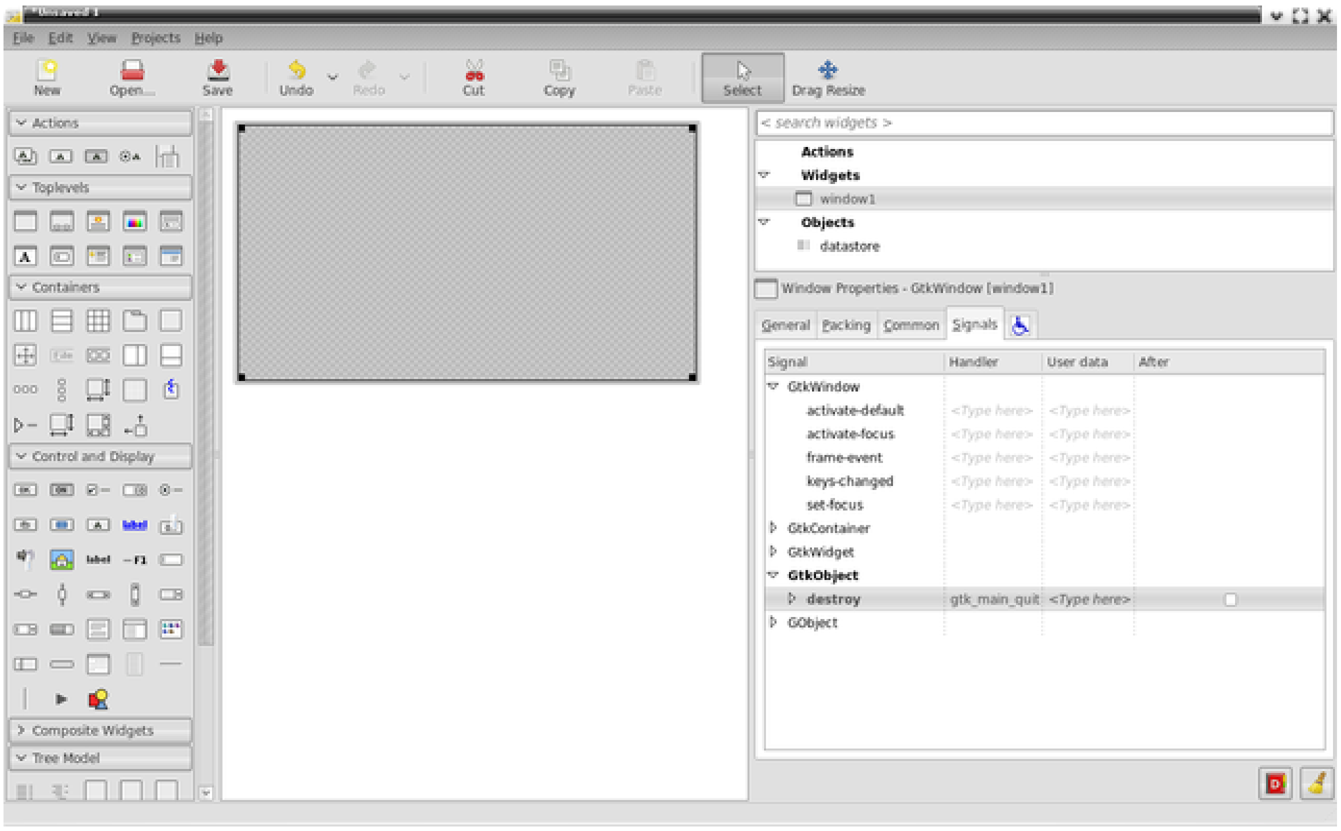
Add remove and edit rows of data (you can optionally use CNTL-N to add new rows and the Delete key to remove the selected row)

Add and remove rows: Sequential editing: Vertical

Image	Progress	Desc
gtk-add	12	Add icon is ...
gtk-cdrom	78	CD-ROM ima...
gtk-cut	98	Scissors roc...

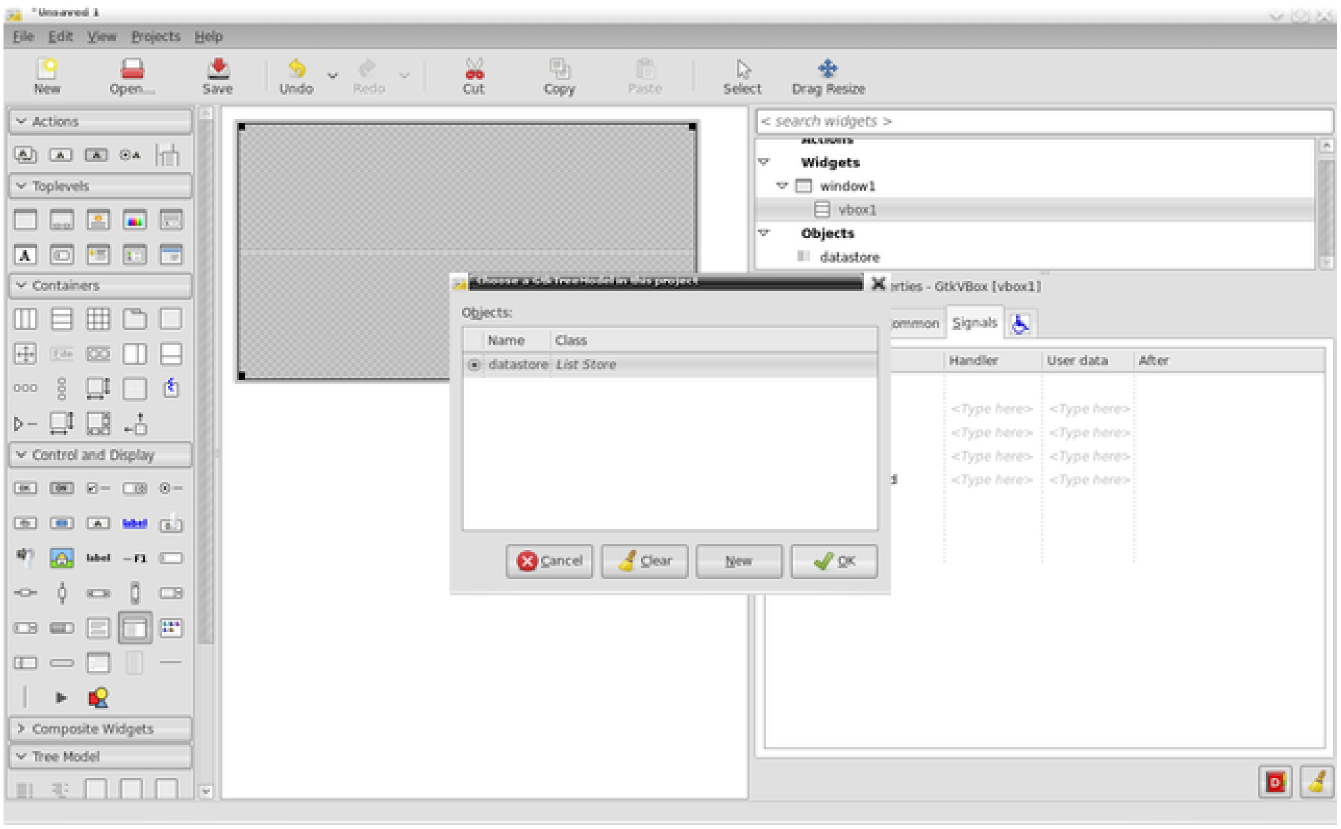
[http://img10.imageshack.us/img10/6893/20090419212810.png] And that's it. We now have our model part of Model/View/Controller (MVC) complex.

Now we'll create a window and connect it's "destroy" signal to gtk_main_quit function, add vbox with two rows to it and pack GtkTreeView into upper section of vbox (lower section is only a placeholder for the next part of the tutorial, where we'll be adding some controls to our application). When we place our tree view into vbox, glade ask us about model. We'll click on "..." and select "datastore" model, click OK and OK again. And controller part of MVC is done too. Now for the last part - creating display components.



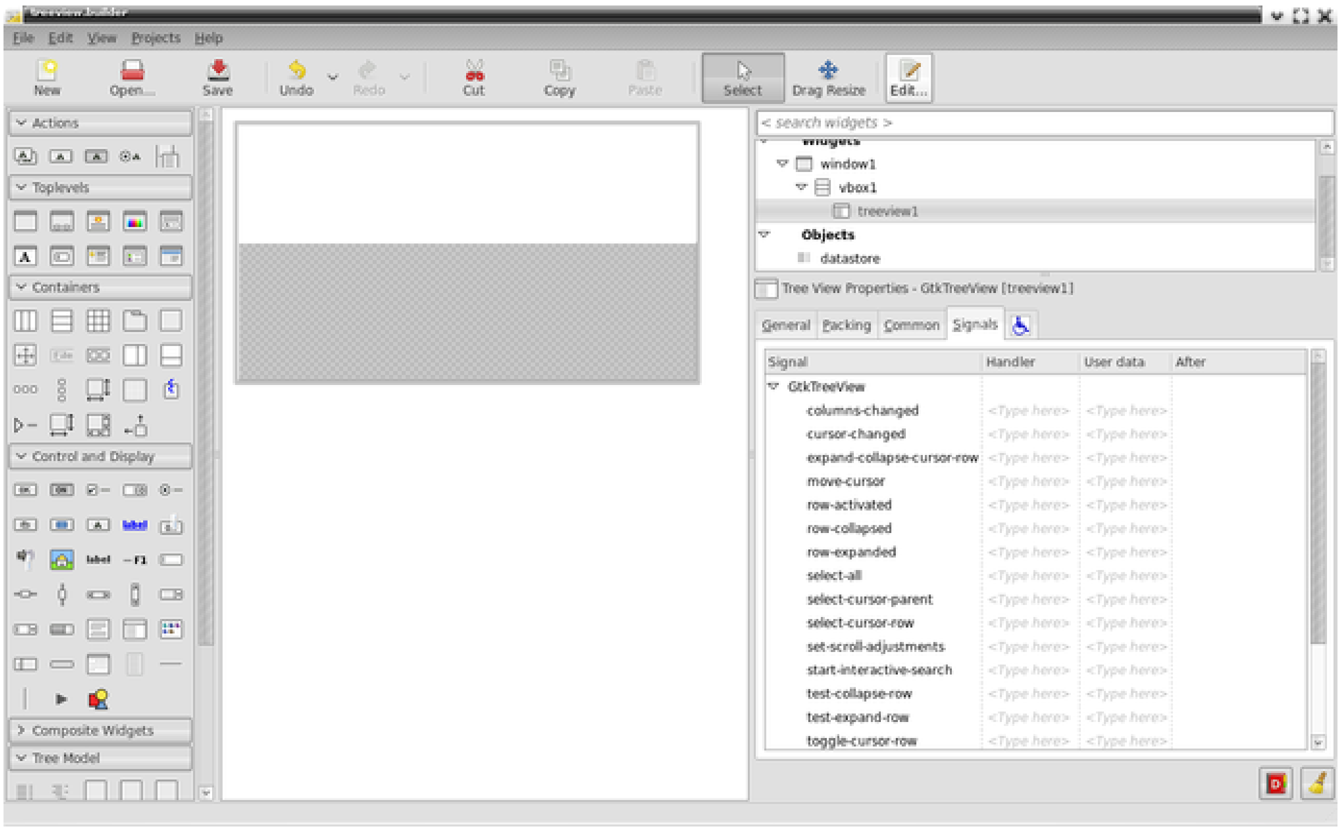
Signal	Handler	User data	After
GtkWindow			
activate-default	<Type here>	<Type here>	
activate-focus	<Type here>	<Type here>	
frame-event	<Type here>	<Type here>	
keys-changed	<Type here>	<Type here>	
set-focus	<Type here>	<Type here>	
GtkContainer			
GtkWidget			
GtkObject			
destroy	gtk_main_quit	<Type here>	☐
GObject			

[http://img9.imageshack.us/img9/1024/20090419212847.png]



Name	Class
datastore	List Store

[http://img524.imageshack.us/img524/7192/20090419212909.png]

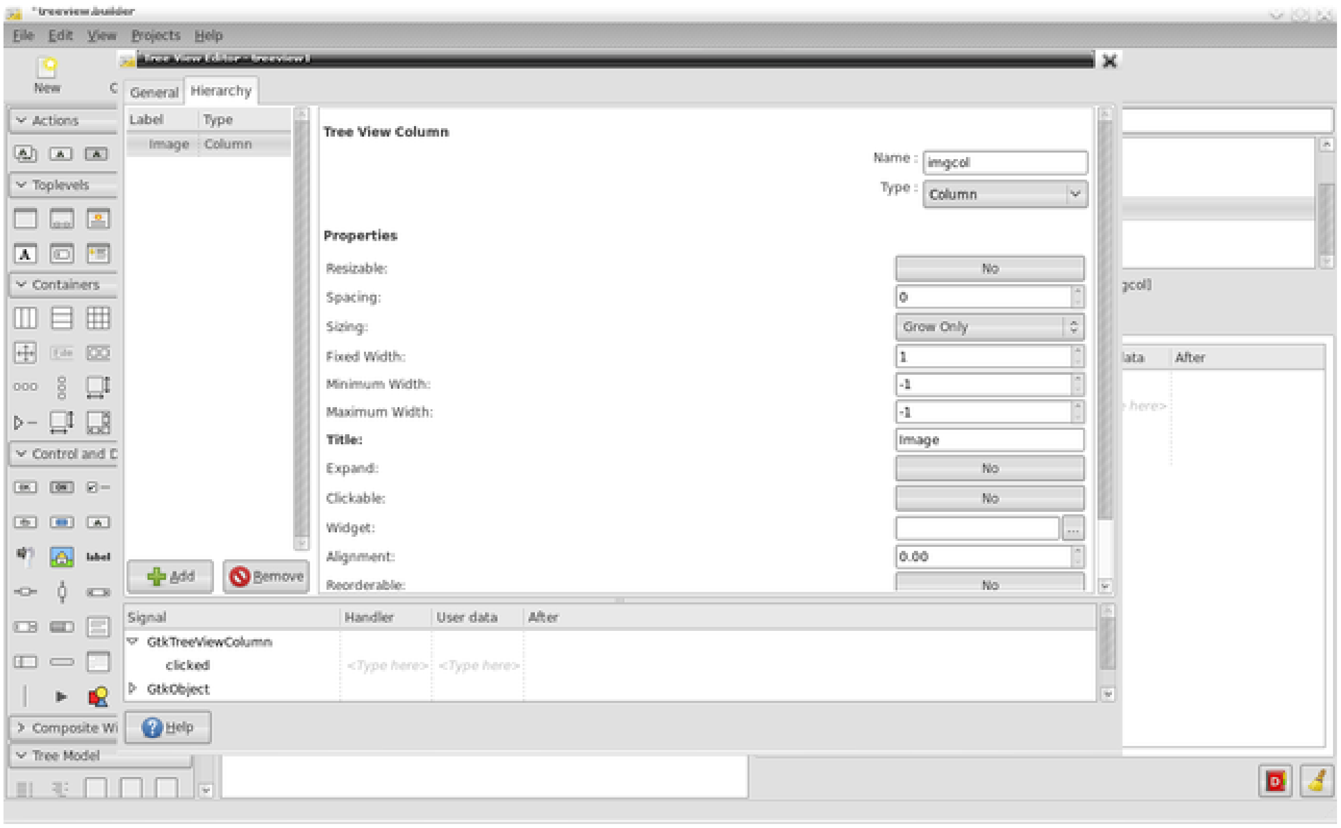


Signal	Handler	User data	After
GtkTreeView			
columns-changed	<Type here>	<Type here>	
cursor-changed	<Type here>	<Type here>	
expand-collapse-cursor-row	<Type here>	<Type here>	
move-cursor	<Type here>	<Type here>	
row-activated	<Type here>	<Type here>	
row-collapsed	<Type here>	<Type here>	
row-expanded	<Type here>	<Type here>	
select-all	<Type here>	<Type here>	
select-cursor-parent	<Type here>	<Type here>	
select-cursor-row	<Type here>	<Type here>	
set-scroll-adjustments	<Type here>	<Type here>	
start-interactive-search	<Type here>	<Type here>	
test-collapse-row	<Type here>	<Type here>	
test-expand-row	<Type here>	<Type here>	
toggle-cursor-row	<Type here>	<Type here>	

[http://img259.imageshack.us/img259/1/20090419212955.png] To add columns and renderers to out tree view, we need to select it and the click "Edit ..." button. Another window will open, looking something like this:

[http://img11.imageshack.us/img11/4673/20090419213009.png] In general tab, we'll only rename "treeview1" to "treeview". We could change entries in our data store in this window too, but we already did that when creating model.

Now we'll open "Hierarchy" tab and start creating columns and renderers. First we need to add column for images and their names. We click on "Add" button and edit newly created GtkTreeViewColumn by renaming it to "imgcol" and setting title to "Image".



Label	Type
Image	Column

Tree View Column

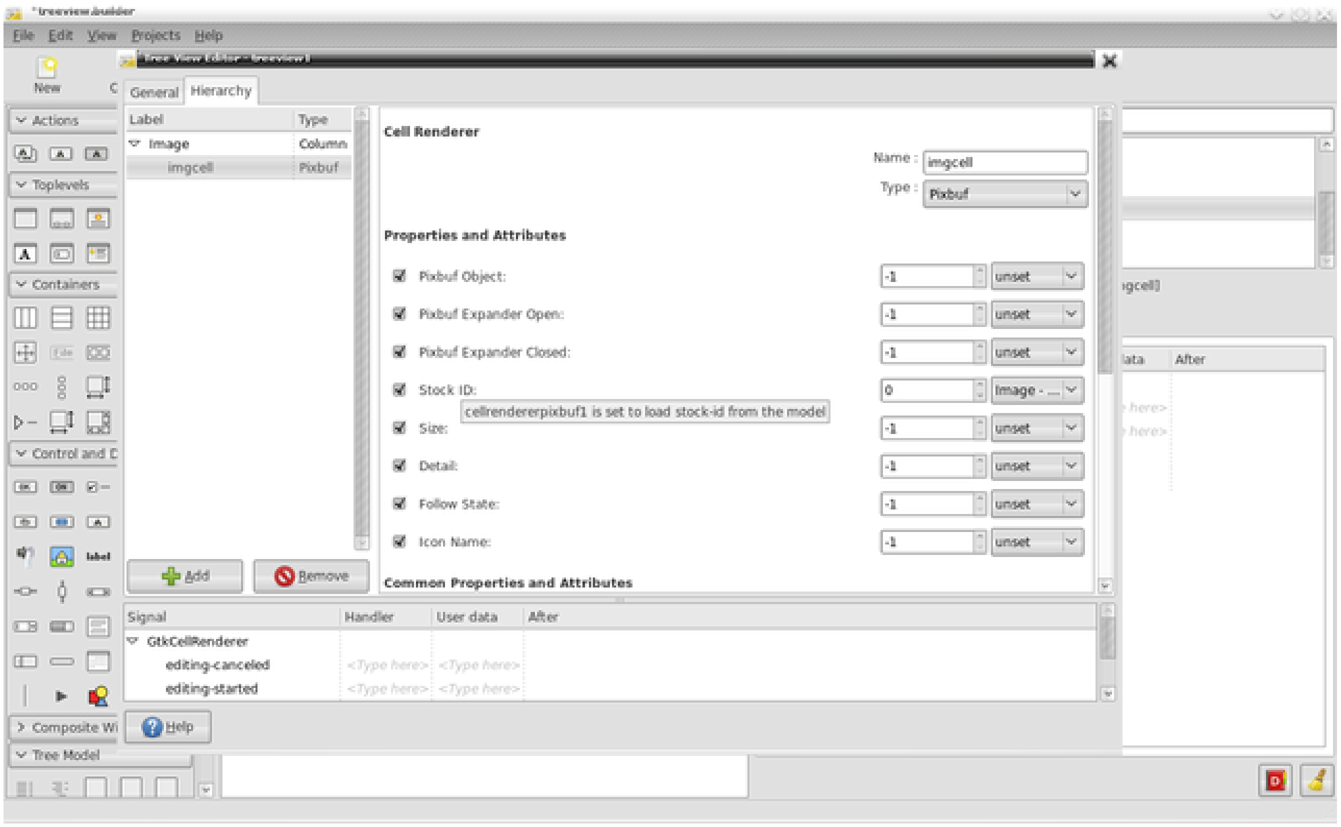
Name : imgcol
Type : Column

Properties

Resizable:	No
Spacing:	0
Sizing:	Grow Only
Fixed Width:	1
Minimum Width:	-1
Maximum Width:	-1
Title:	Image
Expand:	No
Clickable:	No
Widget:	
Alignment:	0.00
Reorderable:	No

Signal	Handler	User data	After
GtkTreeViewColumn			
clicked	<Type here>	<Type here>	
GtkObject			

[http://img209.imageshack.us/img209/5345/20090419213047.png] After we're done with editing, we need to right-click on our column and select "Add child Pixbuf item". This will add GtkCellRendererPixbuf renderer to the column. We'll rename this renderer to "imgcell" and set it's stock id property to Image model column.



Label	Type
Image	Column
imgcell	Pixbuf

Cell Renderer

Name : imgcell
Type : Pixbuf

Properties and Attributes

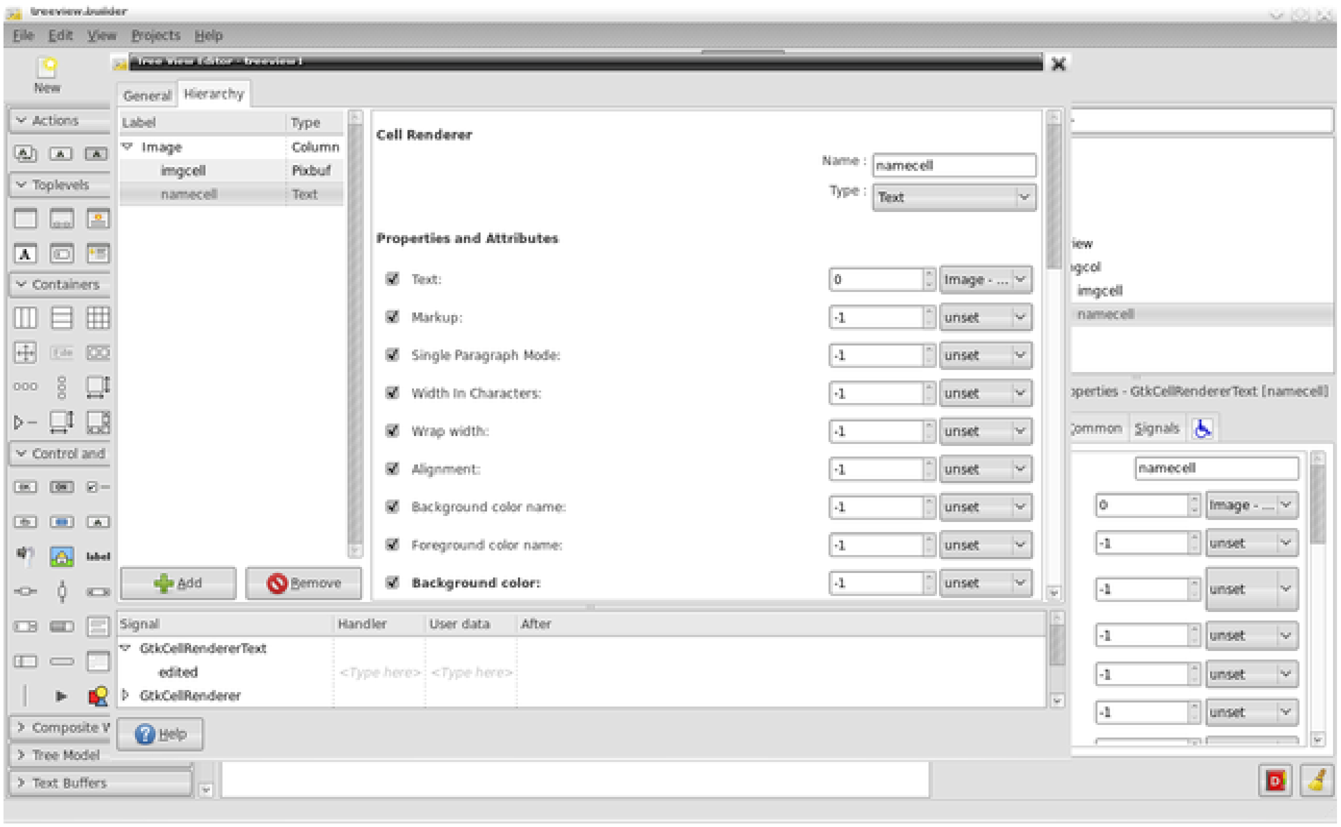
Pixbuf Object:	-1	unset
Pixbuf Expander Open:	-1	unset
Pixbuf Expander Closed:	-1	unset
Stock ID:	0	Image - ...
Size:	-1	unset
Detail:	-1	unset
Follow State:	-1	unset
Icon Name:	-1	unset

cellrendererpixbuf1 is set to load stock-id from the model

Common Properties and Attributes

Signal	Handler	User data	After
GtkCellRenderer			
editing-canceled	<Type here>	<Type here>	
editing-started	<Type here>	<Type here>	

[http://img410.imageshack.us/img410/764/20090419213158.png] Now we need to add text renderer which will display image's name, but this time we'll be adding it a little different. First, we select previous cell renderer and then click "Add" button. This created new cell renderer, without the defined type. We'll rename this cell renderer to "namecell", set it's type to "Text" and it's text property to "Image" model column.



Label	Type
Image	Column
imgcell	Pixbuf
namecell	Text

Cell Renderer

Name : namecell

Type : Text

Properties and Attributes

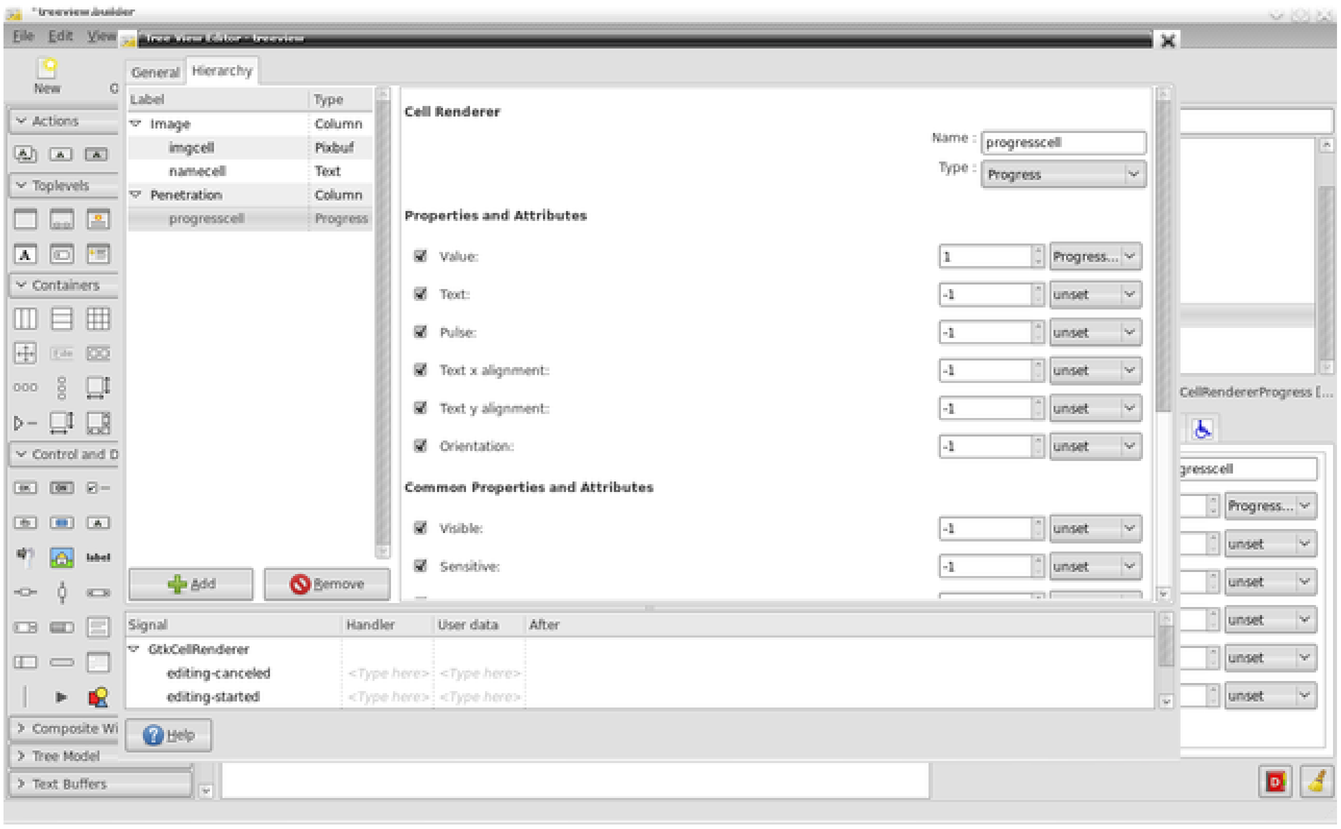
✓ Text:	0	Image - ...
✓ Markup:	-1	unset
✓ Single Paragraph Mode:	-1	unset
✓ Width In Characters:	-1	unset
✓ Wrap width:	-1	unset
✓ Alignment:	-1	unset
✓ Background color name:	-1	unset
✓ Foreground color name:	-1	unset
✓ Background color:	-1	unset

Signal	Handler	User data	After
GtkCellRendererText			
edited	<Type here>	<Type here>	
GtkCellRenderer			

iew
gcol
imgcell
namecell

operties - GtkCellRendererText [namecell]

[http://img2.imageshack.us/img2/1674/20090419213206.png] Now we need to create second column for progress bar. Select the last tree view column inside the left part of the window where columns and renderers reside and click "Add" button. Then rename newly created column to "progresscol" like we renamed first column and set title to "Penetration". Now create new progress cell renderer. Did you manage to do it by yourself? If you did, congratulations. If not, you should right-click your column and select "Add child Progress item". Now rename this cell renderer to "progresscell" and set it's "Value" property to "Progress" model column.



Label	Type
Image	Column
imgcell	Pixbuf
namecell	Text
Penetration	Column
progresscell	Progress

Cell Renderer

Name : progresscell
Type : Progress

Properties and Attributes

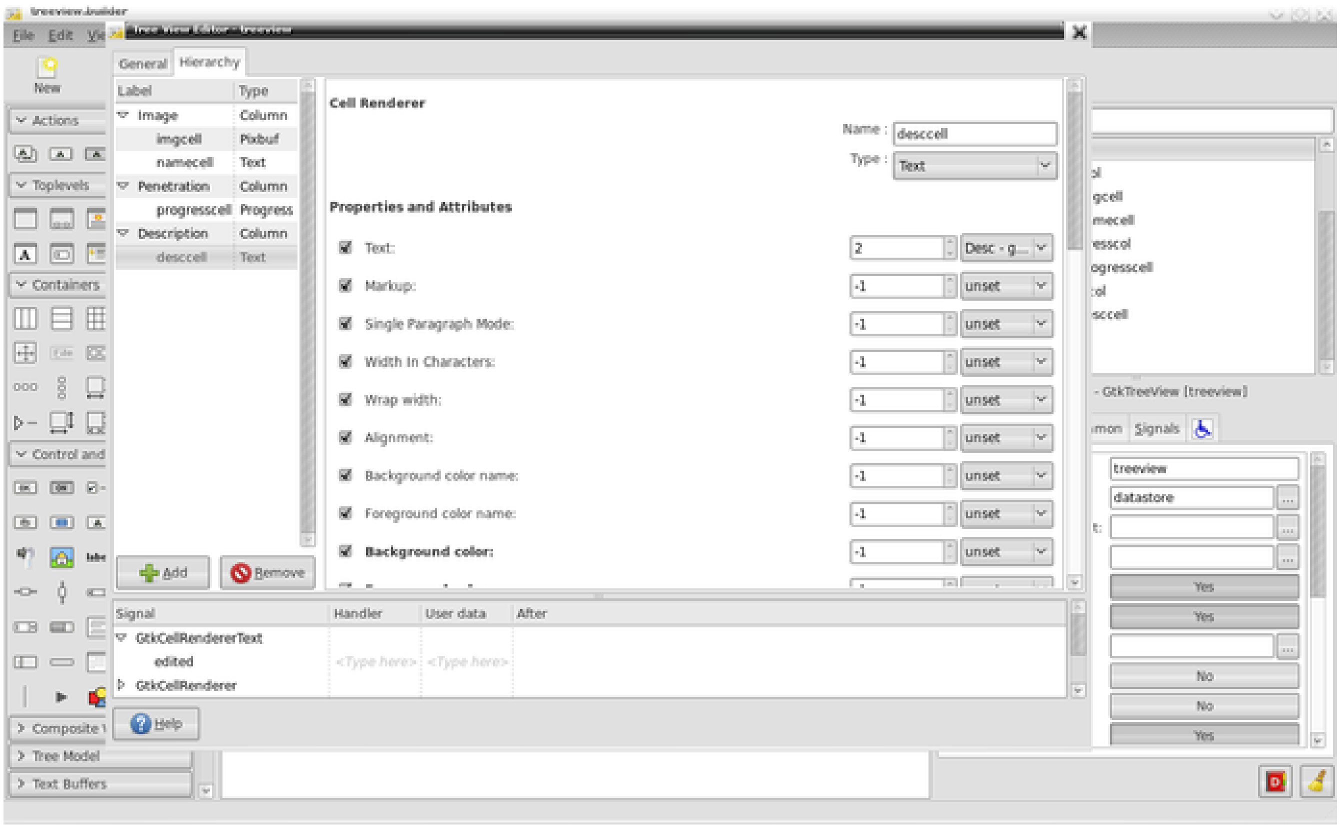
Value:	1	Progress...
Text:	-1	unset
Pulse:	-1	unset
Text x alignment:	-1	unset
Text y alignment:	-1	unset
Orientation:	-1	unset

Common Properties and Attributes

Visible:	-1	unset
Sensitive:	-1	unset

Signal	Handler	User data	After
GtkCellRenderer			
editing-canceled	<Type here>	<Type here>	
editing-started	<Type here>	<Type here>	

[http://img25.imageshack.us/img25/2788/20090419213714.png] And now it's time for you to create last column, rename it to "desccol", set it's title to "Description"; add text cell renderer to it, rename it to "desccell" and set it's "text" property to "Desc" model column. After you're finished, your Hierarchy tab should look something like this:



Label	Type
Image	Column
imgcell	Pixbuf
namecell	Text
Penetration	Column
progresscell	Progress
Description	Column
desccell	Text

Cell Renderer

Name : desccell
Type : Text

Properties and Attributes

Text:	2	Desc - g...
Markup:	-1	unset
Single Paragraph Mode:	-1	unset
Width In Characters:	-1	unset
Wrap width:	-1	unset
Alignment:	-1	unset
Background color name:	-1	unset
Foreground color name:	-1	unset
Background color:	-1	unset

Signal	Handler	User data	After
GtkCellRendererText			
edited	<Type here>	<Type here>	
GtkCellRenderer			

[http://img396.imageshack.us/img396/7581/20090419213906.png] Close the Edit window and you should see your tree view already populated with some data. And that is actually all that we needed to do today. All that is left for us to do is to store glade project and test our product. I wrote a simple C application that will load our builder file and create an application from it.

```
/*
 * treeview.c - Application for testing builder file.
 *
 * Compile with:
 *  gcc -o treeview treeview.c $(pkg-config --cflags --libs gtk+-2.0 gmodule-export-2.0)
 */

#include <gtk/gtk.h>

int
main( int    argc,
      char **argv )
{
    /* Vars */
    GtkWidget  *window;
    GtkBuilder *builder;
    GError     *error;

    /* Initialization */
    gtk_init( &argc, &argv );

    /* Create builder and load UI file */
    builder = gtk_builder_new();
    if( ! gtk_builder_add_from_file( builder, "treeview.builder", &error ) )
    {
        g_print( "Error occured while loading UI file!\n" );
        g_print( "Message: %s\n", error->message );
        g_free( error );

        return( 1 );
    }

    /* Get main window and connect signals */
    window = GTK_WIDGET( gtk_builder_get_object( builder, "window1" ) );
    gtk_builder_connect_signals( builder, NULL );

    /* Destroy builder */
    g_object_unref( G_OBJECT( builder ) );

    /* Show main window and start main loop */
    gtk_widget_show( window );
    gtk_main();

    return( 0 );
}
```

I hope you enjoyed this tutorial and be back for the second part, where we'll also learn how to manipulate data inside our store. Bye.

Posted 20th April 2009 by Tadej Borovšak

Labels: GTK+, GtkTreeView, tutorial

11 View comments

capitantyler 29 October, 2009 03:09

I ve read your tutorial and is very clear!

I have done my front end application (whitout code, only app.glade)
so, I copied your script from above in a text archive, I saved and run it in a terminal, but it doesn t respond. Why?

Reply

JockeTF 13 November, 2009 01:00

Thanks for the nice tutorial!

Reply

Ivan Baldo 13 November, 2009 21:34

Hey thanks a lot! I was wondering how to add the cell renderers, didn't know I had to right click on the columns to add them.
Nice article, concise, clear, to the point, excellent!

Reply

Anonymous 22 December, 2009 20:26

Hi,

Great tutorial.
I got it finaly working.

Got the following hickups.
1) I coudn't find the edit mode of the treeview. I had to rightclick the treeview, and choose edit.
2) I coudn't find the the link between de column and the cellrender. Had to set my text property from -1 to 0 or 1 or whatever.

It is all there... i know... but just to let jou know where i missed the boat.

Reply

Joe 27 December, 2009 02:35

Thanks!

Reply

---

ubuntrucchi 20 February, 2010 00:02

thank you for your tutorial.
even if I'm using python it's really clear and useful.

Reply

Anonymous 05 May, 2010 14:30

Excellent tutorial. A couple of nits from a Fedora 12 installation.

(1) Initialize GError * error = 0 to avoid an assertion failure when loading the UI.

(2) Use g_error_free(error), not g_free() to free the error if there is a failure loading the UI.

(3) (Appropriate to the second part of this tutorial). The new "horizontal" glade-3 will require that the 'vbox1' object have its 'orientation' property set to 'vertical' to get the Up/Down buttons below the tree view.

Reply

Anonymous 06 May, 2010 16:21

The tutorial is very helpful for gtk newbies. Thanks you.

Reply

Anonymous 12 May, 2010 22:56

For the newbie (like me) - it must be made clear that the GtkTreeView is a "view". You can use any renderer you like, but if you do not map back to the model you will not have values displayed.

Text and Combo renderers need the 'Text' property set and Toggle renderers need the 'Toggle state' property set.

It seems obvious to me now, but I had a conceptual disconnect trying to attach a Combo renderer. The 'Text column' identifies the source of the options in the 'Model' help by the renderer, but the 'Text' property must be set to map the selection to the column in the model held by the view.

Reply

Anonymous 30 December, 2010 09:40

Thanks! I was a little lost about glade/treeviews before reading this.

Reply

Ilyas 20 July, 2011 10:23

Hi,

I had the following error message when executing the code. Any idea how to solve the problem? Thank you and great tutorial!

```
(treeview:3885): Gtk-CRITICAL **: gtk_builder_add_from_file: assertion `error == NULL || *error == NULL' failed
Error occured while loading UI file!
Message: PQR�T$ �D$
������Z�
$� $�D$ �

*** glibc detected *** ./treeview: free(): invalid pointer: 0x08049ff4 ***
======= Backtrace: =========
/lib/tls/i686/cmov/libc.so.6[0xb7751604]
/lib/tls/i686/cmov/libc.so.6(cfree+0x96)[0xb77535b6]
/usr/lib/libglib-2.0.so.0(g_free+0x36)[0xb7887126]
./treeview(main+0x74)[0x80489f8]
/lib/tls/i686/cmov/libc.so.6(__libc_start_main+0xe5)[0xb76f8775]
```

```
./treeview[0x80488f1]
======= Memory map: ========
```

Reply

Enter your comment...

Comment as: Google Accou ▼

Publish Preview