16th April 2009

GtkComboBox widget - part 3

Welcome to the last part of our journey through wonders of GtkComboBox widgets. Today's lesson will be a bit different that previous two. Consider it a test at the end of the semester. When you understand every piece of code I wrote today, you'll be able to create almost any combo box you'll ever need.

Contents of this tutorial:

Test

Again, if you're not familiar with Model/View/Controller-design, read second part of my tutorial.

That being said, here comes the big momma of all combo boxes;) Enjoy dissecting this application.

Bye

```
/*
 * Test me with:
 *  gcc -o combo2 combo2.c $(pkg-config --cflags --libs gtk+-2.0) && ./combo2
 */

#include <gtk/gtk.h>

/* This enumeration simplifies using data store, since column numbers can be
 * accessed via more meaningful names. */
enum
{
    /* Text related columns */
    TEXT_C = 0,     /* Column with text strings */
    TEXT_VIS_C,     /* Visibility column for text strings */
    TEXT_COL_C,     /* Text color column */

    /* Image related columns */
    PIXBUF_C,      /* Column with GdkPixbufs */
    PIXBUF_VIS_C, /* Visibility column for pixbufs */

    /* Progress renderer related columns */
    PROGRESS_C,      /* Column with progress information [0, 100] */
    PROGRESS_VIS_C, /* Column with progress visibility */

    /* Last element of enumeration holds the number of columns */
    N_COLS
};

/* Structure that holds widgets we need in our callback functions. */
```

```
typedef struct _Data Data;
struct _Data
{
    GtkWidget    *combo;  /* Our combo box */
    GtkTreeModel *store;  /* Just a conveniance to avoid calling
                             gtk_combo_box_get_model every time we need to
                             access to data. */

    /* Check buttons that control visibility of renderers. */
    GtkWidget *vis_pixbuf;
    GtkWidget *vis_text;
    GtkWidget *vis_progress;

    /* Entries for modifying values */
    GtkWidget *e_pixbuf;
    GtkWidget *e_text;
    GtkWidget *e_progress;
};

/* Callback function for updating current item. */
static void
cb_clicked( GtkButton *button,
            Data      *data )
{
    const gchar *stock_id, *string;
    gint         value;
    gboolean     pix, text, prog;
    GtkTreeIter  iter;

    /* If nothing is selected, do nothing. */
    if( ! gtk_combo_box_get_active_iter( GTK_COMBO_BOX( data->combo ), &iter ) )
        return;

    /* Fill variables with proper data */
    stock_id = gtk_entry_get_text( GTK_ENTRY( data->e_pixbuf ) );
    string   = gtk_entry_get_text( GTK_ENTRY( data->e_text ) );
    value    = gtk_spin_button_get_value_as_int( GTK_SPIN_BUTTON( data->e_progress ) );
    pix  = gtk_toggle_button_get_active( GTK_TOGGLE_BUTTON( data->vis_pixbuf ) );
    text = gtk_toggle_button_get_active( GTK_TOGGLE_BUTTON( data->vis_text ) );
    prog = gtk_toggle_button_get_active( GTK_TOGGLE_BUTTON( data->vis_progress ) );

    /* Update data store for current iter */
    gtk_tree_store_set( GTK_TREE_STORE( data->store ), &iter,
                        TEXT_C, string,
                        TEXT_VIS_C, text,
                        PIXBUF_C, stock_id,
                        PIXBUF_VIS_C, pix,
                        PROGRESS_C, value,
                        PROGRESS_VIS_C, prog,
                        -1 );
}
```

```
/* Callback function for changed signal.
 * In this function, we'll set the widgets that control current line. */
static void
cb_changed( GtkComboBox *combo,
            Data        *data )
{
    /* sensitive flag */
    static gboolean sensitive = TRUE;

    /* Vars */
    GtkTreeIter  iter;
    gchar       *stock_id = NULL, *string = NULL;
    gint         value;
    gboolean     pix, text, prog;
    gboolean     active;

    /* Get active iter from combo box. If nothing is selected,
     * disable controls. */
    active = gtk_combo_box_get_active_iter( combo, &iter );
    if( active )
    {
        gtk_tree_model_get( data->store, &iter, TEXT_C, &string,
                                                TEXT_VIS_C, &text,
                                                PIXBUF_C, &stock_id,
                                                PIXBUF_VIS_C, &pix,
                                                PROGRESS_C, &value,
                                                PROGRESS_VIS_C, &prog,
                                                -1 );

        gtk_toggle_button_set_active( GTK_TOGGLE_BUTTON( data->vis_pixbuf ), pix );
        gtk_toggle_button_set_active( GTK_TOGGLE_BUTTON( data->vis_text ), text );
        gtk_toggle_button_set_active( GTK_TOGGLE_BUTTON( data->vis_progress ), prog );
        gtk_entry_set_text( GTK_ENTRY( data->e_pixbuf ), stock_id );
        gtk_entry_set_text( GTK_ENTRY( data->e_text ), string );
        gtk_spin_button_set_value( GTK_SPIN_BUTTON( data->e_progress ), value );

        /* Free strings */
        g_free( stock_id );
        g_free( string );
    }

    if( sensitive != active )
    {
        gtk_widget_set_sensitive( data->vis_pixbuf, active );
        gtk_widget_set_sensitive( data->vis_text, active );
        gtk_widget_set_sensitive( data->vis_progress, active );
        gtk_widget_set_sensitive( data->e_pixbuf, active );
        gtk_widget_set_sensitive( data->e_text, active );
        gtk_widget_set_sensitive( data->e_progress, active );
        sensitive = active;
    }
}

/* This function creates tree data structure and fills it with data. */
static GtkTreeModel *
create_model( Data *data )
{
    GtkTreeStore *store;
    GtkTreeIter   parent, child;

    /* Create data store. We'll be using GtkTreeStore today, to show you how
     * combo box handles tree structures. */
    store = gtk_tree_store_new( N_COLS, G_TYPE_STRING,  /* text */
                                        G_TYPE_BOOLEAN, /* text visibility */
                                        G_TYPE_STRING,  /* text color */
                                        G_TYPE_STRING,  /* pixbufs */
                                        G_TYPE_BOOLEAN, /* pixbuf visibility */
                                        G_TYPE_INT,     /* progress bar % */
                                        G_TYPE_BOOLEAN  /* progress vis */
                                        );

    /* Fill our store with some data. */
    gtk_tree_store_append( store, &parent, NULL );
    gtk_tree_store_set( store, &parent, TEXT_C, "Root 1",
                                        TEXT_VIS_C, TRUE,
                                        TEXT_COL_C, "black",
                                        PIXBUF_C, GTK_STOCK_OK,
                                        PIXBUF_VIS_C, TRUE,
                                        PROGRESS_C, 100,
                                        PROGRESS_VIS_C, TRUE,
                                        -1 );

    gtk_tree_store_append( store, &child, &parent );
    gtk_tree_store_set( store, &child, TEXT_C, "Leaf 1.1",
                                       TEXT_VIS_C, TRUE,
                                       TEXT_COL_C, "red",
                                       PIXBUF_C, GTK_STOCK_ADD,
                                       PIXBUF_VIS_C, TRUE,
                                       PROGRESS_C, 100,
                                       PROGRESS_VIS_C, TRUE,
                                       -1 );

    gtk_tree_store_append( store, &child, &parent );
    gtk_tree_store_set( store, &child, TEXT_C, "Leaf 1.2",
                                       TEXT_VIS_C, TRUE,
                                       TEXT_COL_C, "green",
                                       PIXBUF_C, GTK_STOCK_APPLY,
                                       PIXBUF_VIS_C, TRUE,
                                       PROGRESS_C, 100,
                                       PROGRESS_VIS_C, TRUE,
                                       -1 );

    gtk_tree_store_append( store, &child, &parent );
    gtk_tree_store_set( store, &child, TEXT_C, "Leaf 1.3",
                                       TEXT_VIS_C, TRUE,
                                       TEXT_COL_C, "blue",
                                       PIXBUF_C, GTK_STOCK_CDROM,
                                       PIXBUF_VIS_C, TRUE,
                                       PROGRESS_C, 100,
                                       PROGRESS_VIS_C, TRUE,
                                       -1 );

    gtk_tree_store_append( store, &child, &parent );
    gtk_tree_store_set( store, &child, TEXT_C, "Leaf 1.4",
                                       TEXT_VIS_C, TRUE,
                                       TEXT_COL_C, "orange",
                                       PIXBUF_C, GTK_STOCK_QUIT,
                                       PIXBUF_VIS_C, TRUE,
                                       PROGRESS_C, 100,
                                       PROGRESS_VIS_C, TRUE,
                                       -1 );

    gtk_tree_store_append( store, &parent, NULL );
    gtk_tree_store_set( store, &parent, TEXT_C, "Root 2",
                                        TEXT_VIS_C, TRUE,
                                        TEXT_COL_C, "black",
                                        PIXBUF_C, GTK_STOCK_FILE,
                                        PIXBUF_VIS_C, TRUE,
                                        PROGRESS_C, 100,
                                        PROGRESS_VIS_C, TRUE,
                                        -1 );

    gtk_tree_store_append( store, &child, &parent );
    gtk_tree_store_set( store, &child, TEXT_C, "Leaf 2.1",
                                       TEXT_VIS_C, TRUE,
                                       TEXT_COL_C, "blue",
                                       PIXBUF_C, GTK_STOCK_EXECUTE,
                                       PIXBUF_VIS_C, TRUE,
                                       PROGRESS_C, 100,
                                       PROGRESS_VIS_C, TRUE,
                                       -1 );

    gtk_tree_store_append( store, &child, &parent );
    gtk_tree_store_set( store, &child, TEXT_C, "Leaf 2.2",
                                       TEXT_VIS_C, TRUE,
                                       TEXT_COL_C, "red",
                                       PIXBUF_C, GTK_STOCK_HOME,
                                       PIXBUF_VIS_C, TRUE,
                                       PROGRESS_C, 100,
                                       PROGRESS_VIS_C, TRUE,
                                       -1 );
    gtk_tree_store_append( store, &child, &parent );
    gtk_tree_store_set( store, &child, TEXT_C, "Leaf 2.3",
                                       TEXT_VIS_C, TRUE,
                                       TEXT_COL_C, "gray",
                                       PIXBUF_C, GTK_STOCK_INFO,
                                       PIXBUF_VIS_C, TRUE,
                                       PROGRESS_C, 100,
                                       PROGRESS_VIS_C, TRUE,
                                       -1 );

    gtk_tree_store_append( store, &child, &parent );
    gtk_tree_store_set( store, &child, TEXT_C, "Leaf 2.4",
                                       TEXT_VIS_C, TRUE,
                                       TEXT_COL_C, "green",
                                       PIXBUF_C, GTK_STOCK_PRINT,
                                       PIXBUF_VIS_C, TRUE,
                                       PROGRESS_C, 100,
                                       PROGRESS_VIS_C, TRUE,
                                       -1 );

    return( GTK_TREE_MODEL( store ) );
}

/* Main */
int
main( int    argc,
      char **argv )
{
    GtkWidget       *window;
    GtkWidget       *table;
    GtkWidget       *button;
    GtkCellRenderer *cell;
    Data             data;

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

    /* Main window */
    window = gtk_window_new( GTK_WINDOW_TOPLEVEL );
    g_signal_connect( G_OBJECT( window ), "destroy",
                      G_CALLBACK( gtk_main_quit ), NULL );
    gtk_container_set_border_width( GTK_CONTAINER( window ), 10 );

    /* Table */
    table = gtk_table_new( 4, 3, FALSE );
    gtk_container_add( GTK_CONTAINER( window ), table );

    /* Create combo box */
    data.combo = gtk_combo_box_new();
    g_signal_connect( G_OBJECT( data.combo ), "changed",
                      G_CALLBACK( cb_changed ), &data );
    gtk_table_attach( GTK_TABLE( table ), data.combo, 0, 3, 0, 1,
                      GTK_EXPAND | GTK_FILL, GTK_SHRINK, 0, 0 );

    /* Create data store */
    data.store = create_model( &data );

    /* Add data store to combo box */
    gtk_combo_box_set_model( GTK_COMBO_BOX( data.combo ), data.store );
    g_object_unref( G_OBJECT( data.store ) );

    /* Create pixbuf cell renderer */
    cell = gtk_cell_renderer_pixbuf_new();

    /* Add cell renderer to combo box */
    gtk_cell_layout_pack_start( GTK_CELL_LAYOUT( data.combo ), cell, FALSE );

    /* Connect cell renderer with data from store */
    gtk_cell_layout_set_attributes( GTK_CELL_LAYOUT( data.combo ), cell,
                                    "stock-id", PIXBUF_C,
                                    "visible", PIXBUF_VIS_C,
                                    NULL );

    /* Create text cell renderer */
    cell = gtk_cell_renderer_text_new();
    gtk_cell_layout_pack_start( GTK_CELL_LAYOUT( data.combo ), cell, FALSE );
    gtk_cell_layout_set_attributes( GTK_CELL_LAYOUT( data.combo ), cell,
                                    "text", TEXT_C,
                                    "visible", TEXT_VIS_C,
                                    "foreground", TEXT_COL_C,
                                    NULL );

    /* Create progress renderer */
    cell = gtk_cell_renderer_progress_new();
    gtk_cell_layout_pack_start( GTK_CELL_LAYOUT( data.combo ), cell, TRUE );
    gtk_cell_layout_set_attributes( GTK_CELL_LAYOUT( data.combo ), cell,
                                    "value", PROGRESS_C,
                                    "visible", PROGRESS_VIS_C,
                                    NULL );

    /* Create check buttons for controling visibility */
    data.vis_pixbuf = gtk_check_button_new_with_label( "Image visible" );
    gtk_table_attach( GTK_TABLE( table ), data.vis_pixbuf, 0, 1, 1, 2,
                      GTK_EXPAND | GTK_FILL, GTK_SHRINK, 0, 0 );

    data.vis_text = gtk_check_button_new_with_label( "Text visible" );
    gtk_table_attach( GTK_TABLE( table ), data.vis_text, 1, 2, 1, 2,
                      GTK_EXPAND | GTK_FILL, GTK_SHRINK, 0, 0 );

    data.vis_progress = gtk_check_button_new_with_label( "Progress visible" );
    gtk_table_attach( GTK_TABLE( table ), data.vis_progress, 2, 3, 1, 2,
                        GTK_EXPAND | GTK_FILL, GTK_SHRINK, 0, 0 );

    /* Create entries for modifying values */
    data.e_pixbuf = gtk_entry_new();
    gtk_table_attach( GTK_TABLE( table ), data.e_pixbuf, 0, 1, 2, 3,
                      GTK_EXPAND | GTK_FILL, GTK_SHRINK, 0, 0 );

    data.e_text = gtk_entry_new();
    gtk_table_attach( GTK_TABLE( table ), data.e_text, 1, 2, 2, 3,
                      GTK_EXPAND | GTK_FILL, GTK_SHRINK, 0, 0 );

    data.e_progress = gtk_spin_button_new_with_range( 0, 100, 1 );
    gtk_spin_button_set_numeric( GTK_SPIN_BUTTON( data.e_progress ), TRUE );
    gtk_table_attach( GTK_TABLE( table ), data.e_progress, 2, 3, 2, 3,
                      GTK_EXPAND | GTK_FILL, GTK_SHRINK, 0, 0 );

    /* Create button for applying changes */
    button = gtk_button_new_from_stock( GTK_STOCK_APPLY );
    g_signal_connect( G_OBJECT( button ), "clicked",
                      G_CALLBACK( cb_clicked ), &data );
    gtk_table_attach( GTK_TABLE( table ), button, 0, 3, 3, 4,
                      GTK_EXPAND | GTK_FILL, GTK_SHRINK, 0, 0 );

    /* Manually call cb_changed function to set controllers to right state. */
    cb_changed( GTK_COMBO_BOX( data.combo ), &data );

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

    return( 0 );
}
```

Posted 16th April 2009 by Tadej Borovšak

Labels: GTK+, GtkComboBox, tutorial

2 View comments

mazlov 09 July, 2010 13:00

Great! Keep up with this!
This has been really helpful.

Just one thing: a link to a .c file would be nice :)

Reply

Fred 22 April, 2011 19:41

This tutorial is quite nice - thanks!

I'm trying to use GtkComboBox and auto-matically move to the next widget upon receiving the "changed" signal. In my signal handler, I have:

gtk_widget_child_focus (gtk_widget_get_toplevel(wdg)), GTK_DIR_TAB_FORWARD);

Which works for every other widget I use (GtkEntry, GtkButton, others), but not for GtkComboBox. Actual keyboard tabbing from the GtkComboBox widget works fine.

Also ... I'm trying to get focus-in-event & focus-out-event signals for the GtkComboBox widget, but can't.

Any ideas/suggestions?

Thanks!

Reply

Enter your comment...

Comment as: Google Accou ▼

Publish Preview