10 things to know when programming in GTK+

Owen Taylor
Red Hat, Inc
otaylor@redhat.com

FOSDEM, 9 February 2003

GTK+ Background

Scope

- Full set of widgets
- Internationalization
- Accessibility
- Key navigation

Flexible

- Easy to write new widgets
- Many ways to hook into API
- Language bindings for many languages

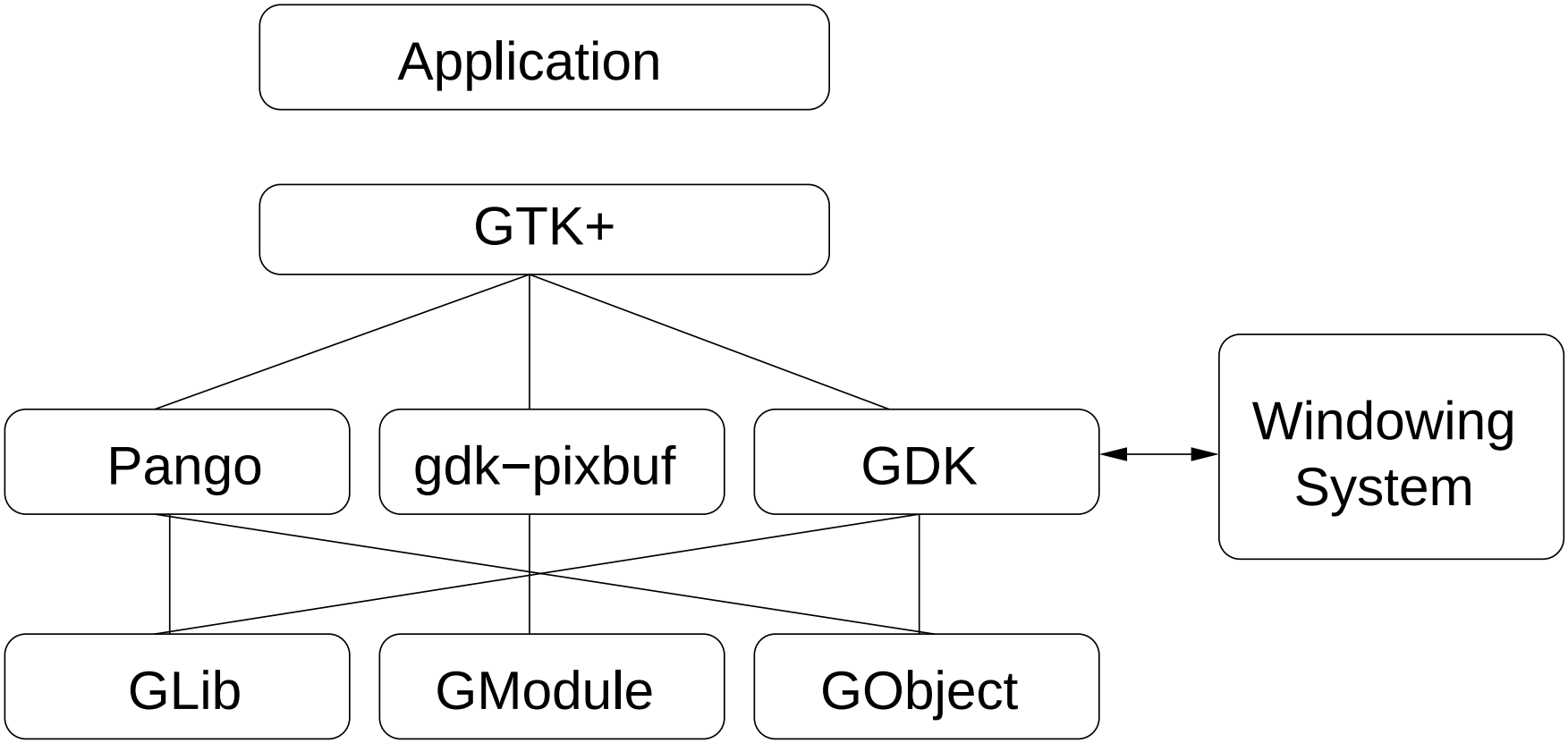
Licensing

- LGPL

Use

- GIMP, GNOME
- 100's of applications

The Libraries



The Libraries (2)

GLib - Utility functions, main loop
GObject - Object system
Pango - International text drawing
ATK - Accessibility interfaces
GDK - Windowing system interface
GTK - Widgets

A first program

```
int
main (int argc, char **argv)
{
  GtkWidget *window, *button;

  gtk_init (&argc, &argv);

  window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
  g_signal_connect (window, "destroy",
                    G_CALLBACK (gtk_main_quit), NULL);

  button = gtk_button_new_with_label ("Hello");
  g_signal_connect (button, "clicked",
                    G_CALLBACK (hello_clicked), window);

  gtk_container_add (GTK_CONTAINER (window), button);
  gtk_widget_show_all (window);

  gtk_main ();

  return 0;
}
```

First program - method calls

Object represented by pointer

```
GtkWidget *window;
```

Inheritance - can "cast" to different types

```
GtkWindow    - GTK_WINDOW (window);
GtkContainer - GTK_CONTAINER (window);
GtkWidget - GTK_WIDGET (window);
GtkObject - GTK_GOBECT (window);
GObject - G_OBJECT (window);
```

Naming convention for methods

```
gtk_window_set_title (GTK_WINDOW (window))
gtk_container_add (GTK_CONTAINER (window), ...);
gtk_widget_show_all (window);
gtk_object_destroy (window);
g_object_get_data (window, "my-key");
```

A first program - signals

Way of getting callbacks when something happens

```
static void
hello_clicked (GtkWidget *button,
               GtkWidget *window)
{
  gtk_object_destroy (GTK_OBJECT (window));
}

g_signal_connect (button, "clicked",
                  G_CALLBACK (hello_clicked), window);
```

Not all signals have same signature

```
void row_activated (GtkTreeView       *tree_view,
                    GtkTreePath       *path,
                    GtkTreeViewColumn *column,
                    gpointer           data);
```

Object properties

Every GObect has a set of properties

Have type, documentation

Useful for GUI builders

```
gtk_window_set_title (GTK_WINDOW (window), "Hello");

g_object_set (G_OBJECT (window),
              "title", "Hello",
              NULL);
```

Containers

Geometry in GTK+ is defined by a heirarchy of widgets

(Different from class inheritence heirarchy)

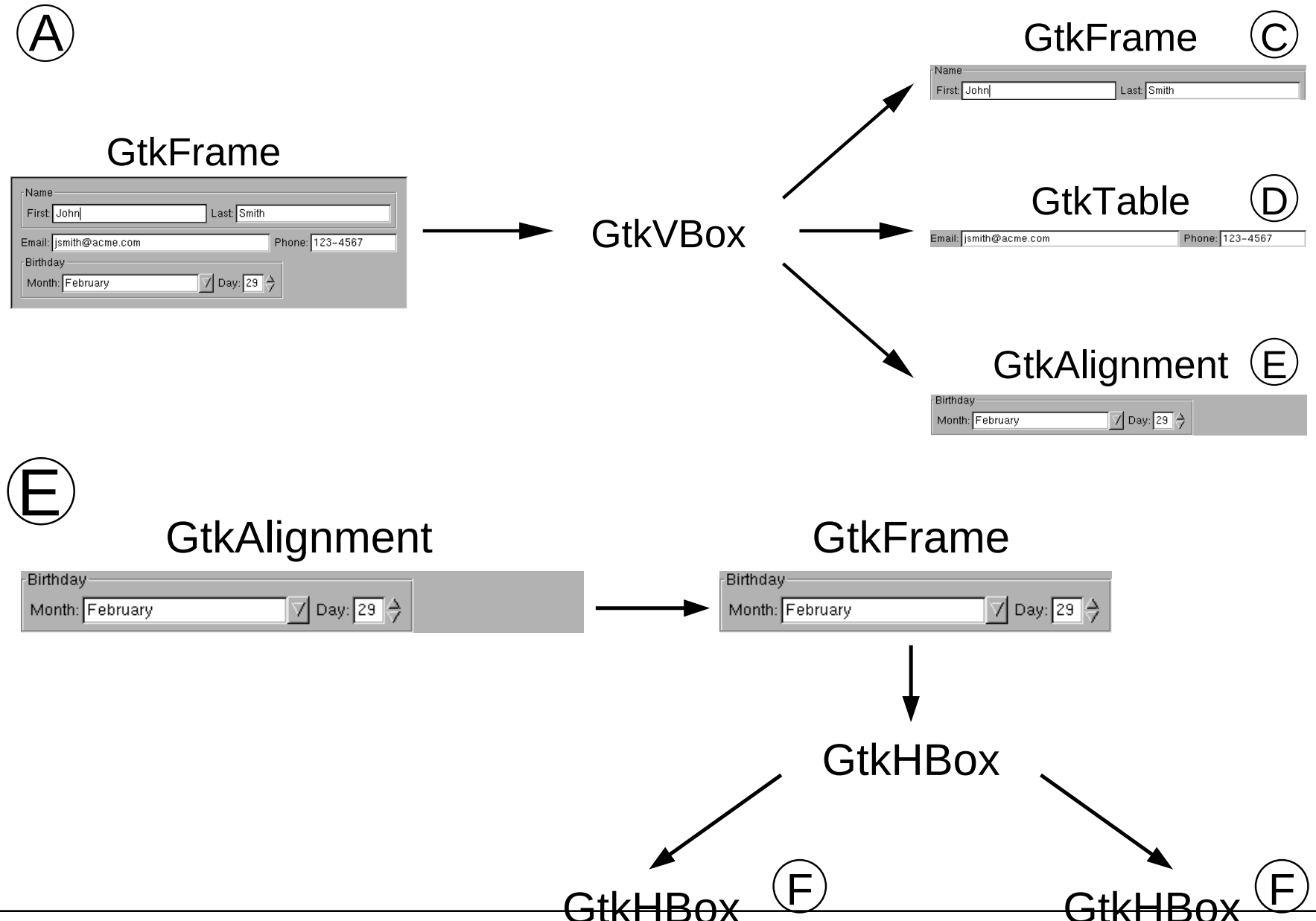
Some containers "decorate" their contents:

- GtkButton
- GtkFrame
  - Put a frame (and label) on the contents
- GtkAlignment
  - Position child widget

Most common containers arrange child widgets

- GtkHBox - horizontal box
- GtkVBox - vertical box
- GtkTable - grid layout of widgets

Containers (2)



Visualize, visualize, visualize

If you don’t know what your app looks like, you can’t code it

Draw mockups on paper

Do mockups with Glade

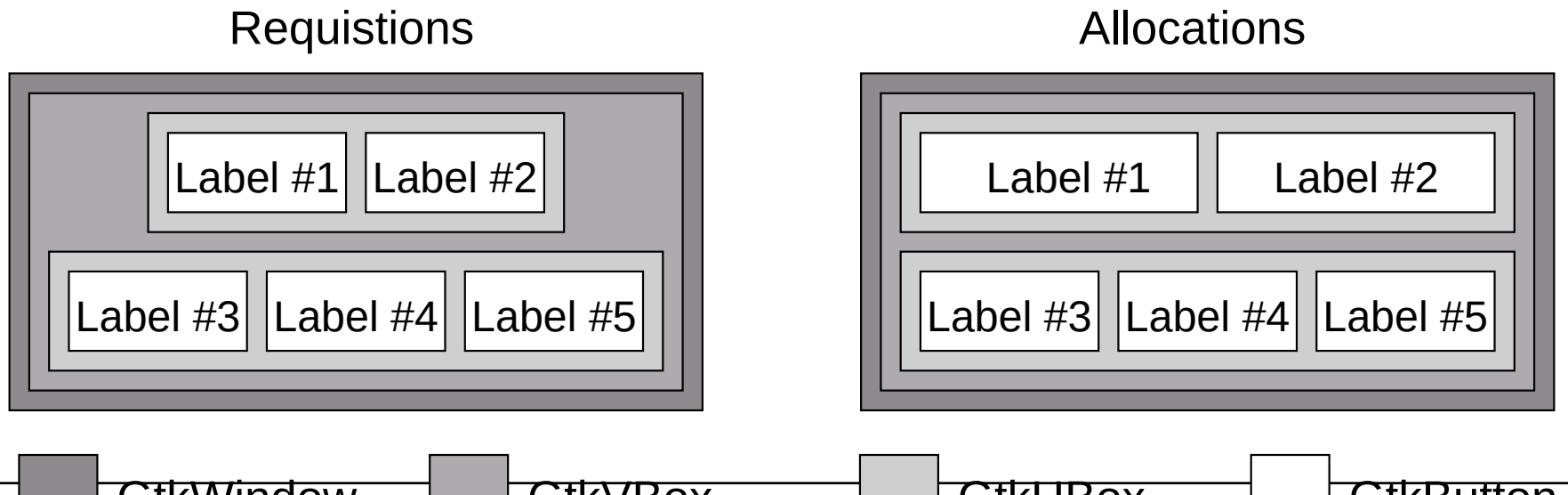
Allocation and Requisition

Requisition

- Each widget says how much space it needs
- Parent widget adds up space of children
- Figure out how much space is needed for toplevel

Allocation

- Start from toplevel
- Each container divides up allocated space among children



Allocation and Requisition (2)

Extra space

- If every widget gets the space it needs, easy
- But frequently, more space than needed
- Packing options control what to do with extra space

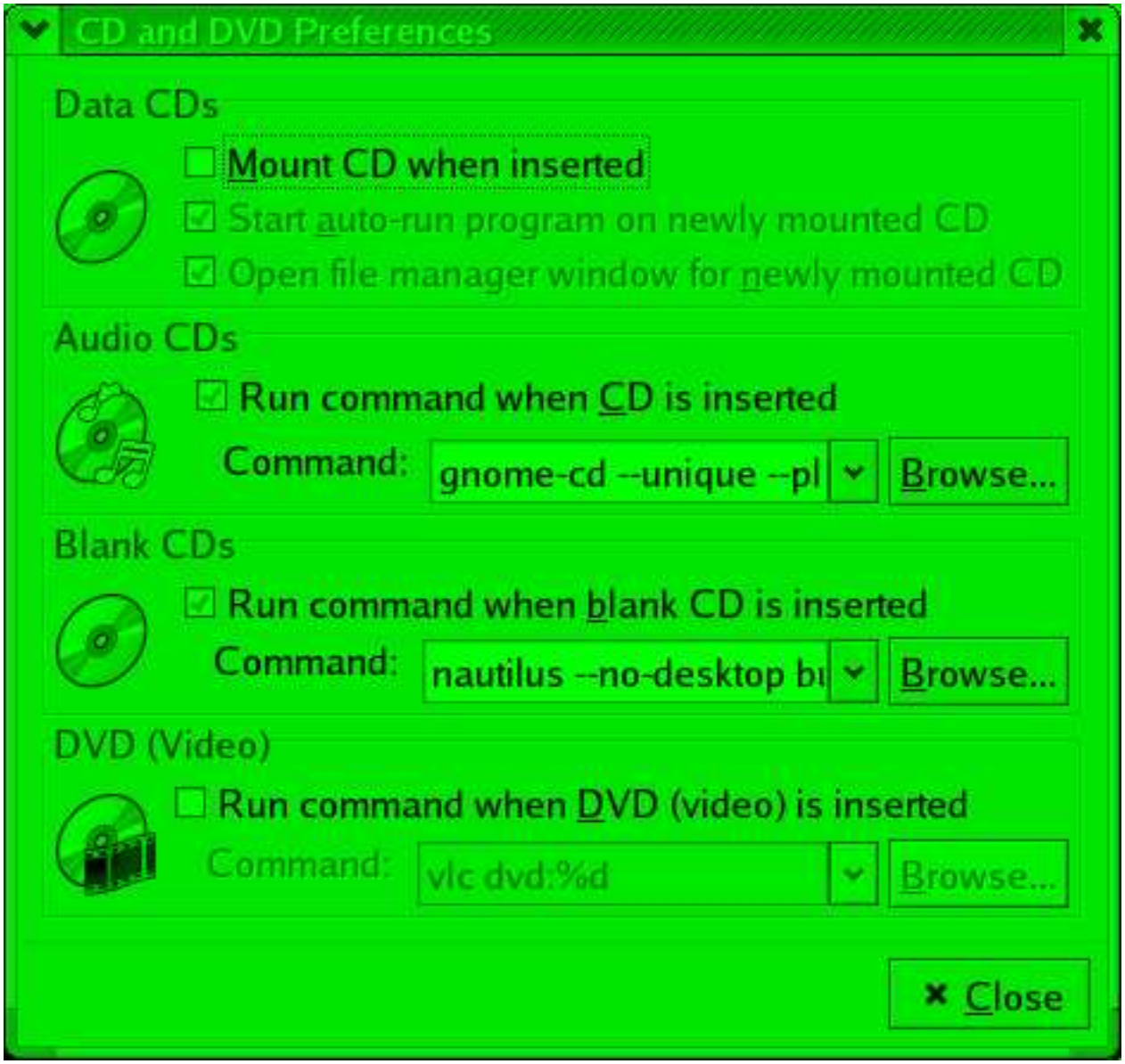
Expand

- Give extra space to this widget

Fill

- Give the extra space to the widget itself
- (rather than make it padding)

Pay attention to details



Pay attention to details

Pay attention to details

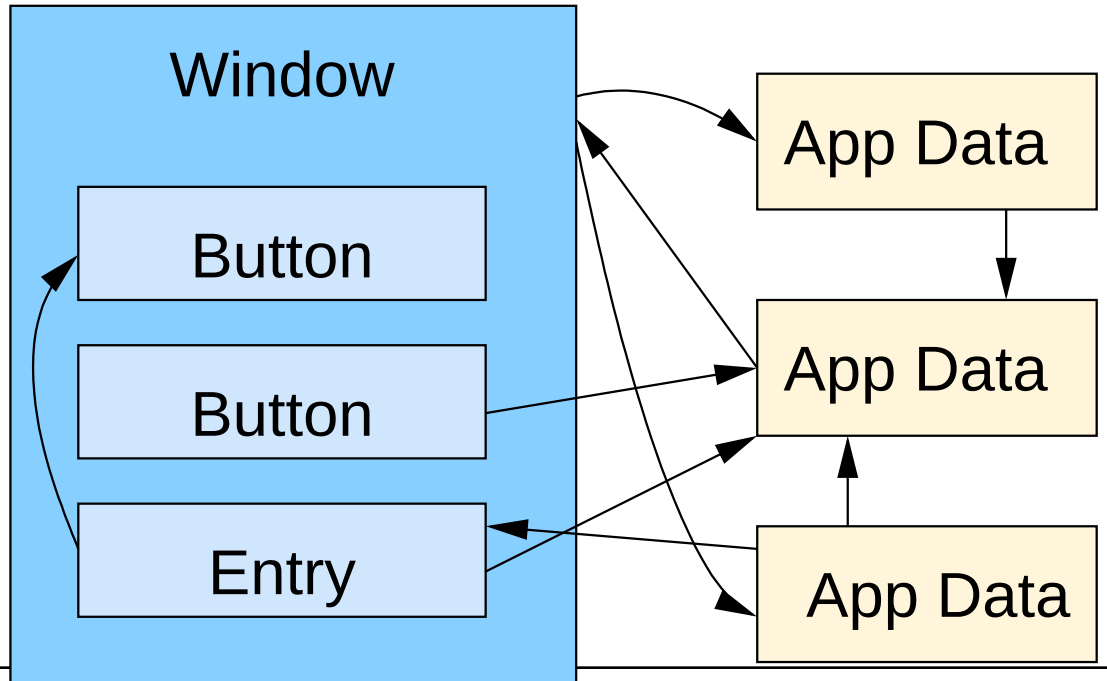
Object data

Store data:

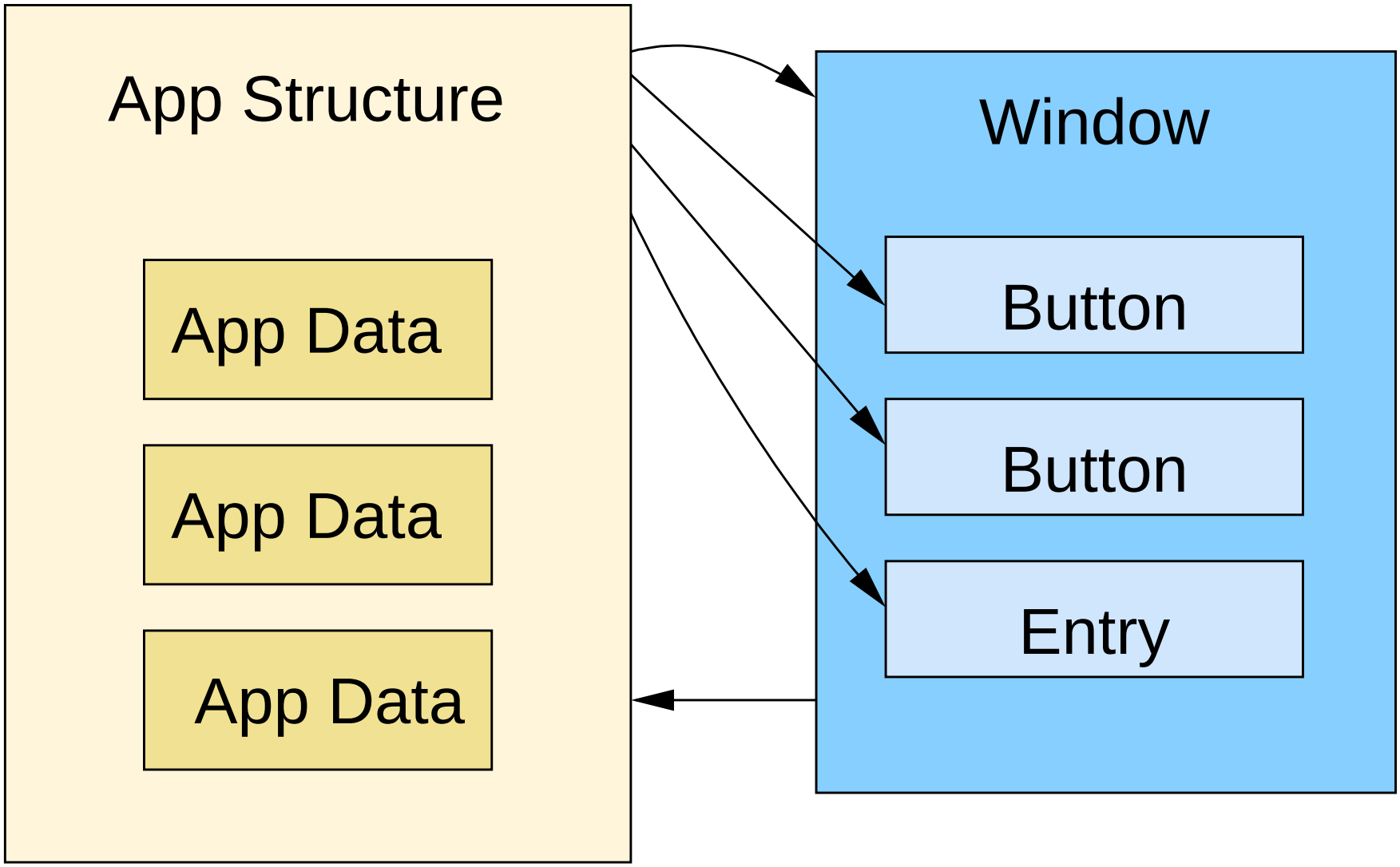
```
g_object_set_data (window,
                   "my-app-data",
                   my_app_data);
```

Retrieve data:

```
my_app_data = g_object_get_data (window,
                                 "my-app-data");
```



App defines structure



App defines structure

```
struct AddressBook
{
   GtkWidget *save_button;
   GtkWidget *open_button;
   GtkWidget *name_entry;

   char *filename;
   char *name;
   [...]
}

AddressBook *
get_address_book (GtkWidget *widget)
{
  GtkWidget *toplevel = gtk_widget_get_toplevel (widget);

  return g_object_get_data (toplevel, "address-book");
```

Basics of GtkTreeView

GtkTreeView used for list and tree display

Model (data store)

- GtkListStore - flat list
- GtkTreeStore - heirarchical data

View

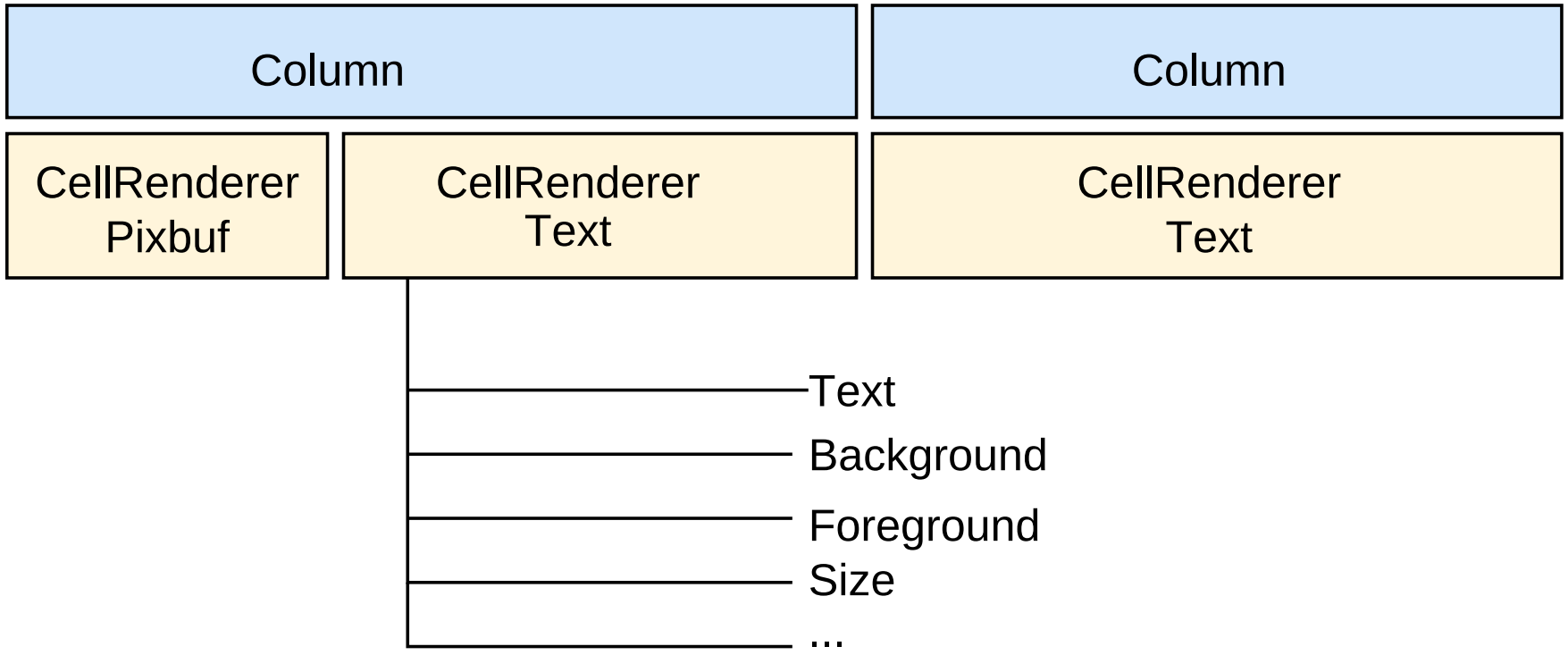
- GtkTreeView widget
- Can have many views of same data

View columns and Cell Renderers

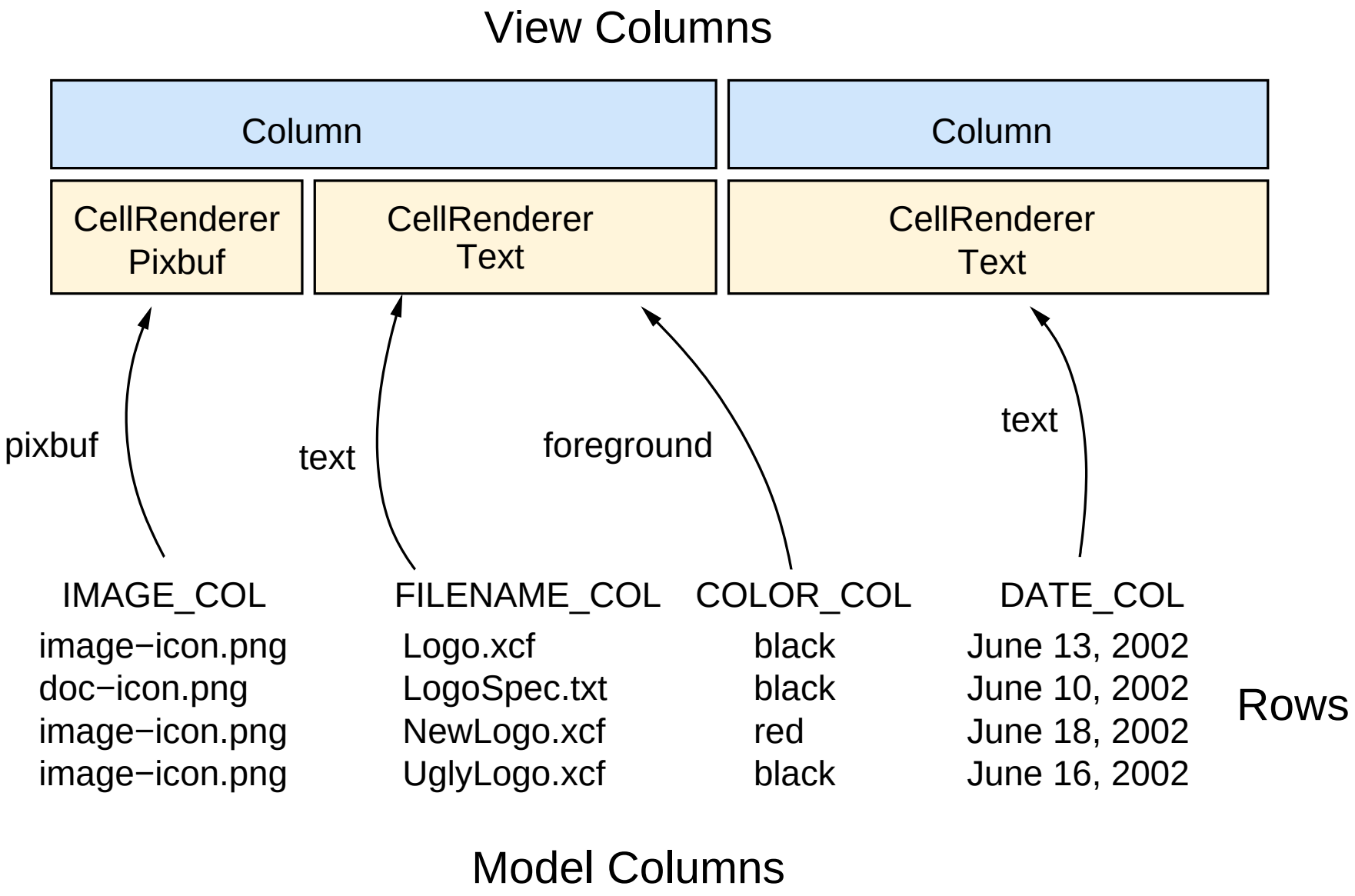
GtkCellRenderer

One or more for each column
(Might have image and text in same column)

Have properties - Text, Color, Style, ...



Setting data from a GtkTreeModel



Live within the toolkit

Using standard widgets gives:

- Proper theming
- Automatic accessibility
- Access to upstream enhancements

Saves work

- Less code to debug
- Writing widgets is hard
  - Learn how to use GObject
  - Lots and lots of details
    (GtkEntry - 4300+ lines of code)

Take advantage of warnings

GTK+ carefully checks all arguments

```
g_return_if_fail (GTK_IS_WIDGET (widget));

(sample:6663): Gtk-CRITICAL **:
file gtkwidget.c: line 1821 (gtk_widget_show_all):
assertion ‘GTK_IS_WIDGET (widget)’ failed
```

Shows:

- In program called ’sample’
- Process ID 6663
- At line 1821 of gtk_widget_show_all()
- The check that the ’widget’ parameter is a widget failed

Take advantage of warnings (2)

--g-fatal-warnings command line option useful

```
$ gdb sample
(gdb) r --g-fatal-warnings
Starting program: /home/otaylor/t/fosdem/sample --g-fatal-warnings

Gtk-CRITICAL **: file gtkwidget.c; line 1821 (gtk_widget_show_all):
assertion 'GTK_IS_WIDGET (widget)' failed
aborting...

Program received signal SIGABRT, Aborted.
0x4041a151 in kill () from /lib/libc.so.6
(gdb) bt
 #0  0x4041a151 in kill () from /lib/libc.so.6
[...]
 #4  0x403b6493 in g_log (log_domain=0x401dcdf3 "Gtk",
    log_level=G_LOG_LEVEL_CRITICAL,
    format=0x401dd000 "file %s: line %d (%s): assertion '%s' failed")
    at gmessages.c:527
 #5  0x401c504f in gtk_widget_show_all (widget=0x0) at gtkwidget.c:1826
 #6  0x080489f4 in main (argc=1, argv=0xbffff964) at sample.c:26
```

Use glade

Use glade to design your dialogs

Use libglade to load up XML files

- (Don’t generate code)

Advantages

- Faster to write
- Easier to play around and improve the GUI
- Easy to set accessibility properties, etc.

Use the right language

```
import gtk

window = gtk.Window ();
window.connect ("destroy", gtk.mainquit)

def hello_clicked(button):
    window.destroy();

button = gtk.Button ("Hello")
button.connect ("clicked", hello_clicked);

window.add (button)

window.show_all()

gtk.main()
```

Use the right language (2)

GTK+ has bindings for:

- Python
- C++
- gtkmm
- Perl
- Ruby
- etc.

C is a hard language

- (pointers, memory management)

C involves lots of typing

More information

Main site

`http://www.gtk.org`

API Docs

`http://developer.gnome.org/doc/API/`