Say “Hello” to PHP-GTK 2

Andrei Zmievski
Yahoo! Inc.

Agenda

Current State

Gtk+ Changes

PHP-GTK Changes

“Real Code” Example

Future Work

PHP-GTK

An extension interfacing with Gtk+ library

Born as an experiment

First release in 2001

Apparently tapped into existing demand

PHP-GTK

4 years later..

Many small and large applications

Small job market has evolved

PHP-GTK

Hindered by PHP 4's object model

No proper object destruction

Crippled object overloading

PHP-GTK

Influenced development of PHP 5

Parameter parsing API

Partially provided motivation for new object model

PHP-GTK 2

Based on PHP 5.1 and Gtk+ 2.6

All the latest bells and whistles

Sets a baseline for future development

Both should be well entrenched by the time of first release of PHP-GTK 2

PHP-GTK 2

Almost a complete rewrite

Mostly backwards compatible

Compatibility broken when necessary

Documentation being rewritten as well

Meet PHP 5

Flexible and powerful object model

Finally has destructors!

Supports modern concepts such as exceptions, interfaces, overloading, etc

Has many hooks for extensibility

Meet Gtk+2

New flexible and powerful type system

Good introspection and extensibility

Can be easily mapped onto PHP object model

Notice the pattern?

Unicode

All user text data is handled in UTF-8

Expects input to be the same

PHP-GTK handles conversion of input and output strings based on a global codepage setting

Once PHP 5.2 is out, native Unicode support will take over

Pango

An open-source framework for layout and rendering of i18n text

Has SGML-like markup language for modifying text attributes (font, size, etc)

Uses Unicode and platform-specific font systems

Gdk 2

Based on the new object system

Classes can have signals, properties, etc

Has double-buffering for smoooth rendering

Much better Win32 support

GdkPixbuf

Merged into GDK, not a separate library anymore

Supports image saving, as well as loading

ATK

A toolkit for adding accessibility to applications

Allows accessibility tools to navigate UI

Complete keyboard navigation for Gtk+

Nearly all key bindings are now customizable

Gtk 2

Many cool new widgets

Many deprecated widgets

API has been cleaned up

Deprecated Widgets

PHP-GTK 2 will issue a warning if you try to use a deprecated issue or method.

In most cases you will be referred to the new widget or method.

GtkList

GtkTree

GtkCList

GtkPixmap

GtkItemFactory

GtkOptionMenu

GtkProgress

GtkPreview

GtkCTree

GtkText

Gtk 2

New stock item system

Has themeable stock icons

Application controls can have consistent and visually pleasing look

Possible to register custom stock icons that can be themed

Model-View Architecture

Provides foundation for data-driven widgets

Model is an abstract interface

Two models provided: list and tree store

Possible to write custom models

List/Tree Widgets

Editable cells

Each cell is drawn by a cell renderer

Included renderers can draw text, image, checkbox, progress bar, and combo box

Possible to write custom cell renderers

List/Tree Widgets

Flexible sorting with custom sort functions

On-the-fly model filtering: hide/display rows based on some condition, restructure existing model, etc

Built-in drag-n-drop

Text Widget

Uses model-view architecture

Full i18n support based on Pango engine

Display and editing of bidi and complex text

Text Widget

Mark objects allow “bookmarking” positions in the text

Text can have many complex attributes applied to it with tag objects: color, size, spacing

Also behavioral features such as editability

Text Widget

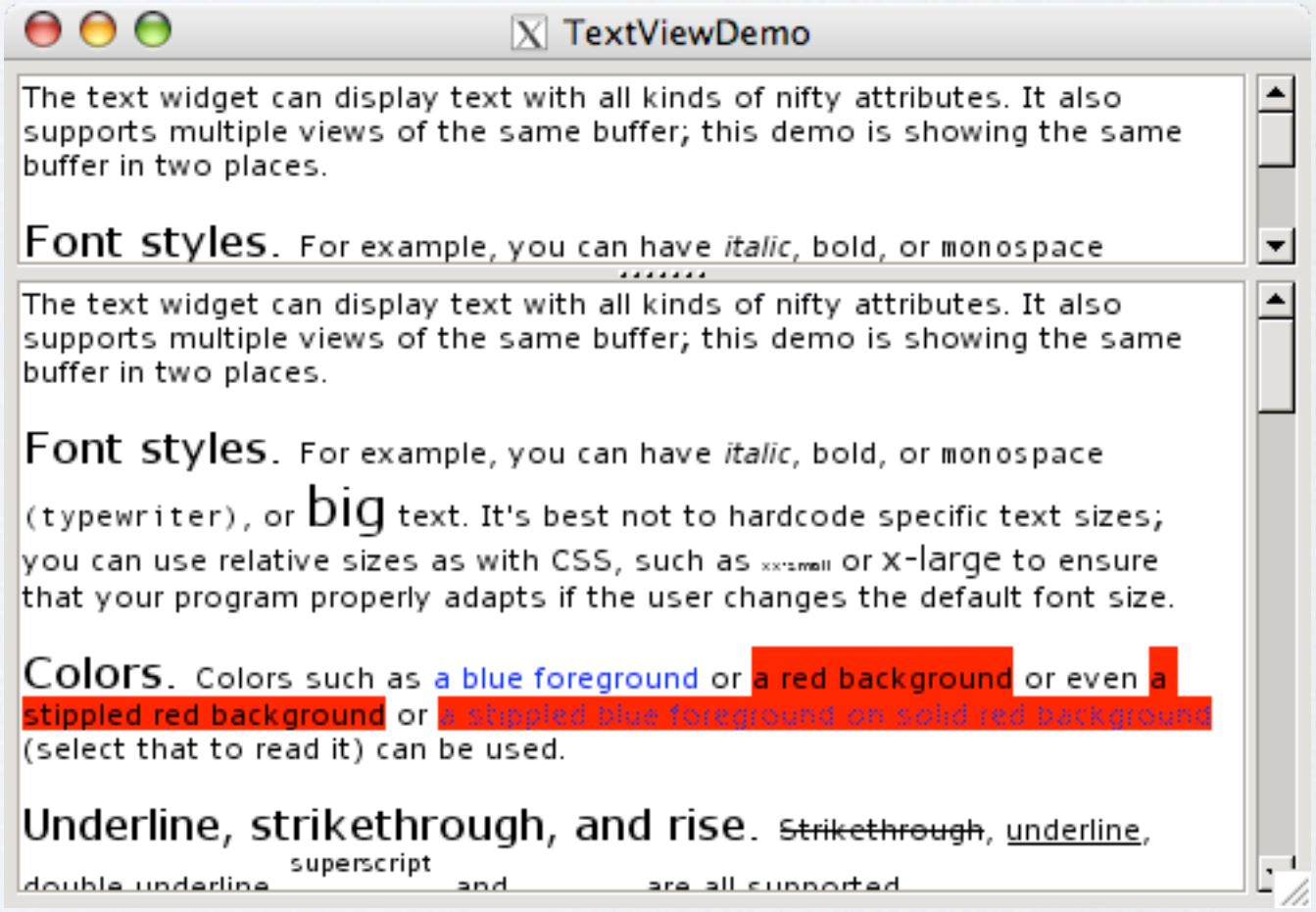
Selection drag-n-drop

Optional “side windows” allow display of additional information such as breakpoints, line numbers, etc

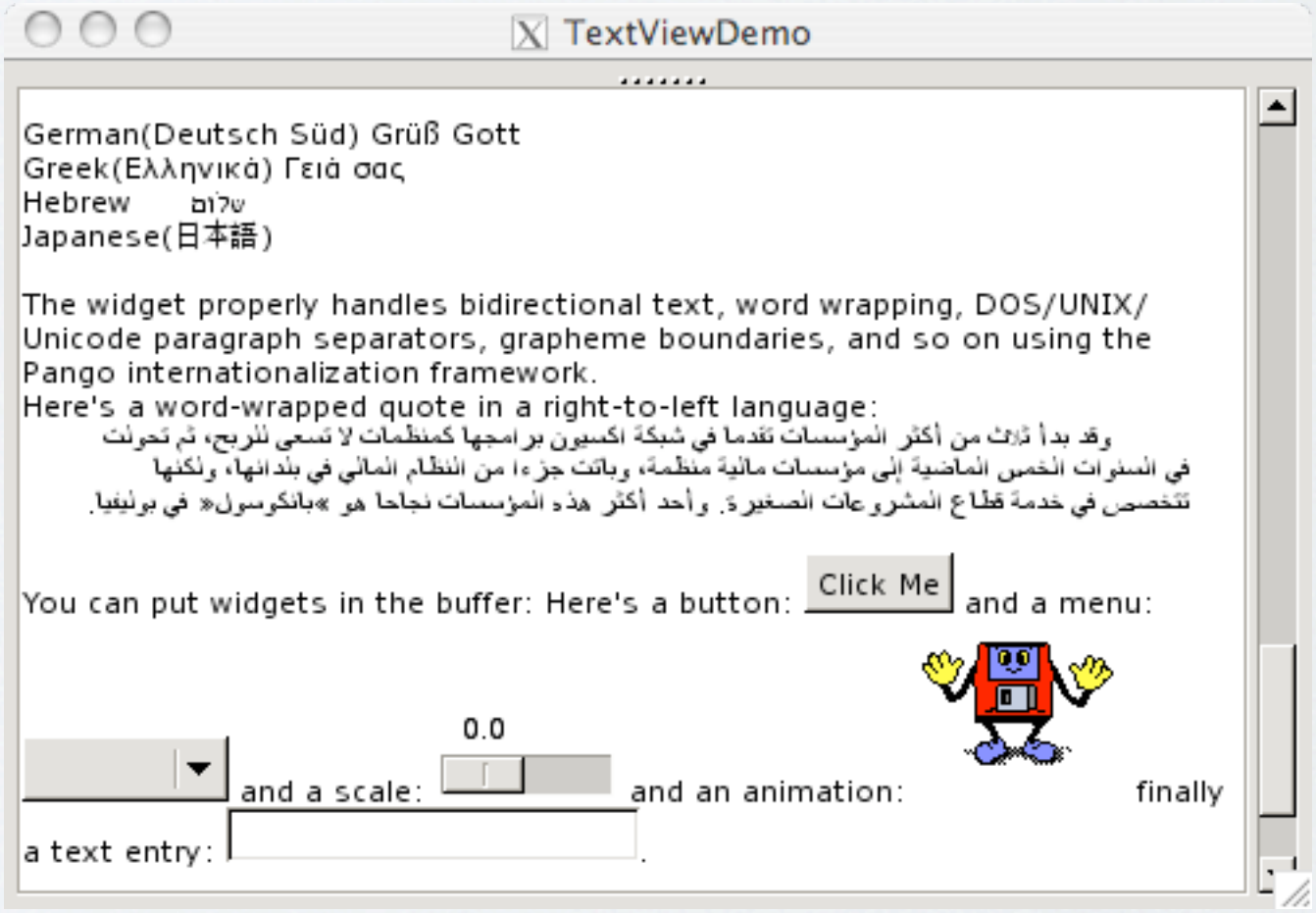
Hooks for implementing undo

New Widget Gallery

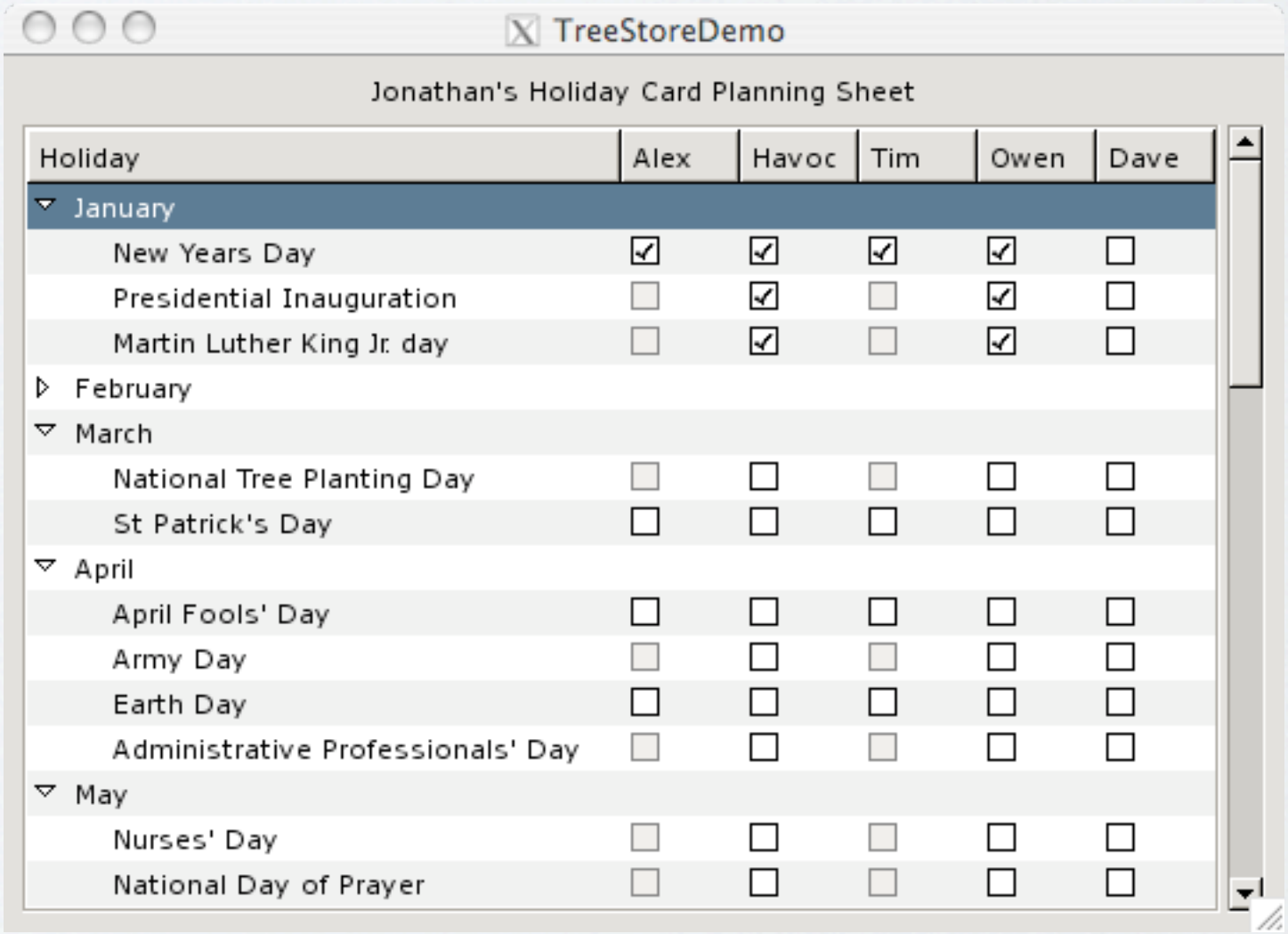
GtkTextView



GtkTextView



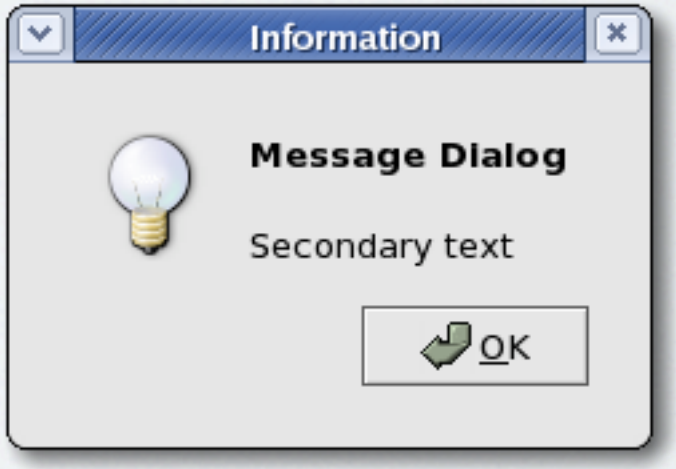
GtkTreeView



GtkMessageDialog

A convenience widget, displaying a message along with an informational icon.

Provides an easy way to display a message and get back user's response.



GtkAboutDialog

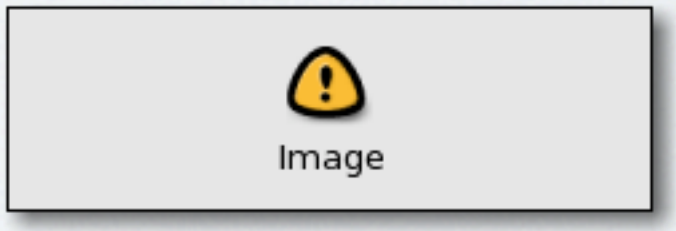
Offers a simple way to display information about a program.

Allows for display of license and program credits.

GtkImage

A widget for easy display of various pictorial information.

Can display pixbufs, pixmaps, icons, stock items, animations.



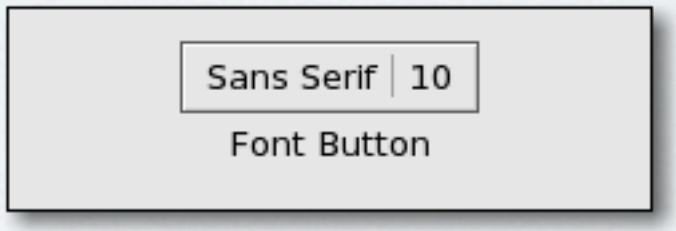
GtkClipboard

Not a widget, but an object simplifying access to system clipboard.

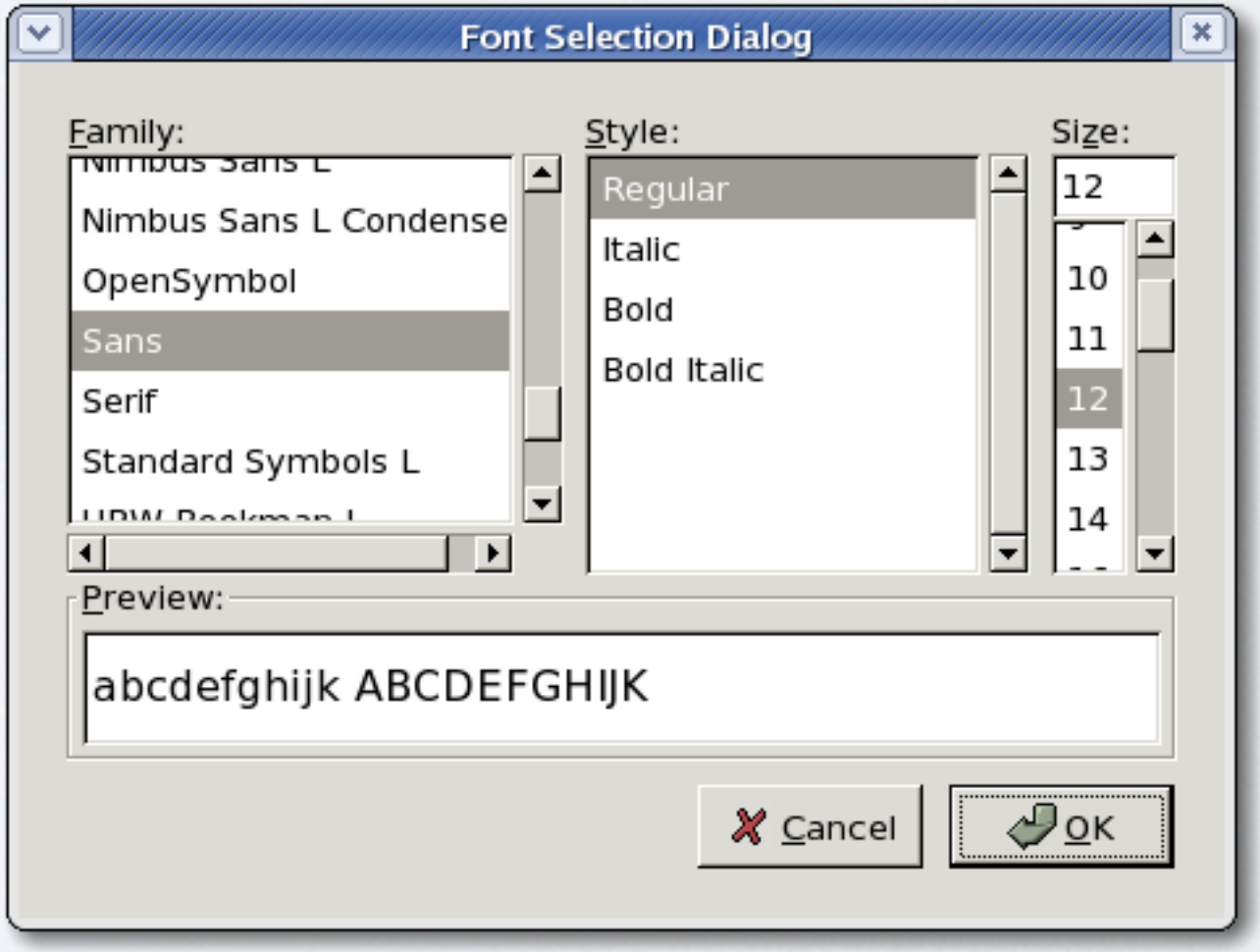
Custom cut, copy, & paste actions are no longer out of reach.

GtkFontButton

Displays the currently selected font and allows to open a font selection dialog to change the font.



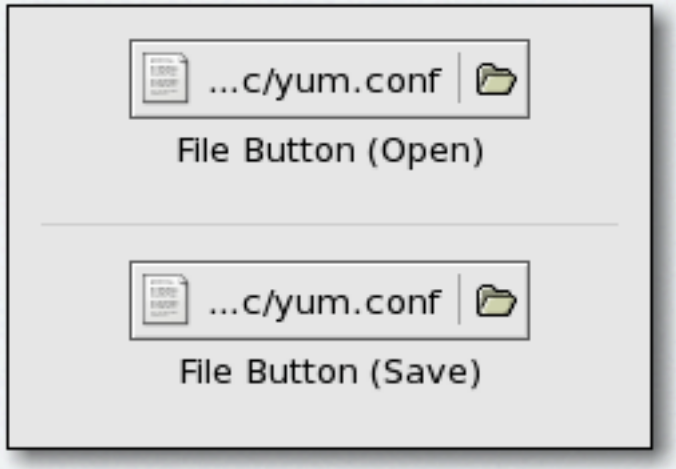
GtkFontSelectionDialog



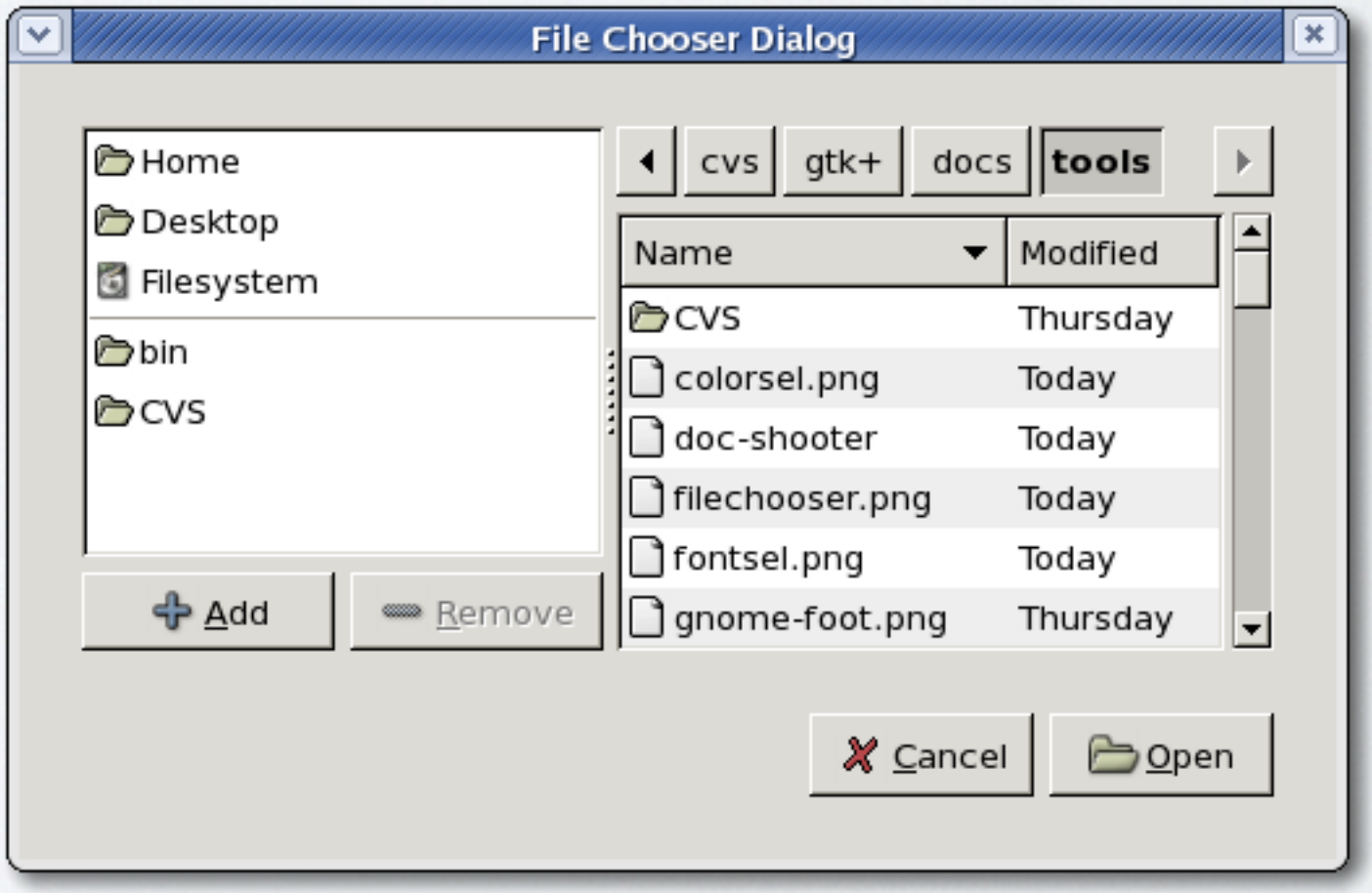
GtkFileChooserButton

Displays current file name and allows opening a file selection dialog to change it.

File selection dialog can have custom preview widget.

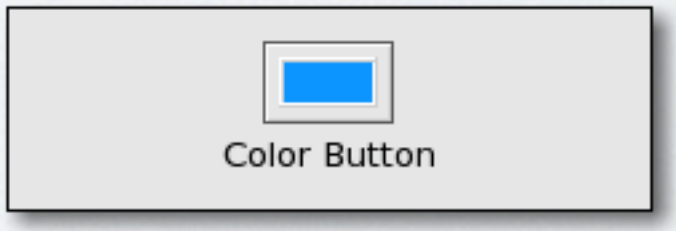


GtkFileChooserDialog

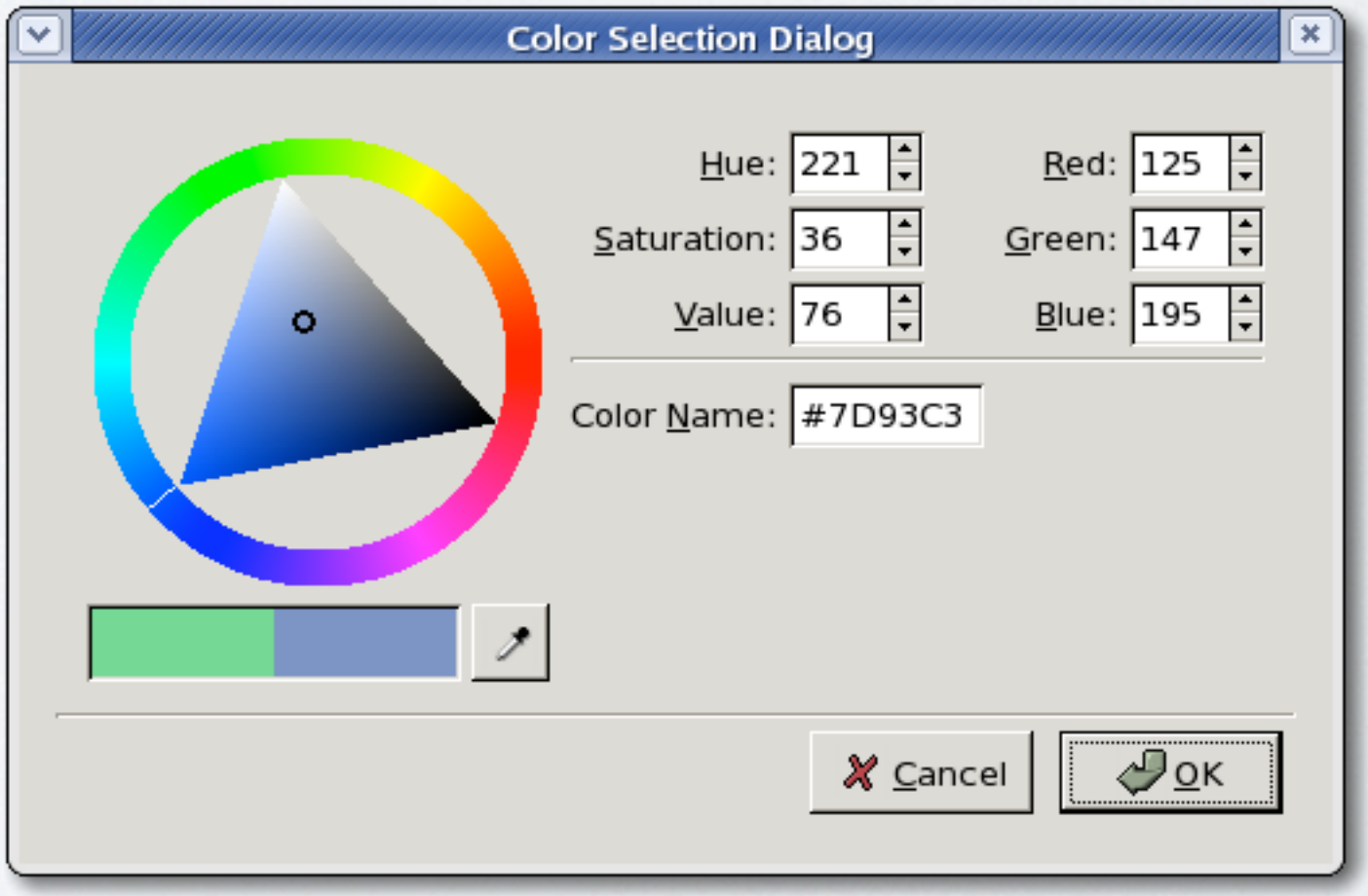


GtkColorButton

Displays the currently selected color and allows opening a color selection dialog to change the color.

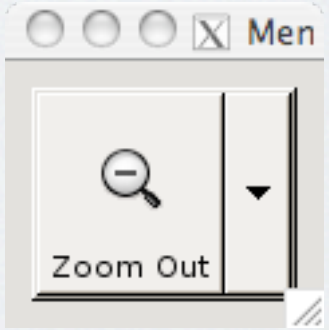


GtkColorSelectionDialog



GtkMenuToolButton

A tool button and a small additional button with an arrow. When clicked, the arrow button pops up a dropdown menu.

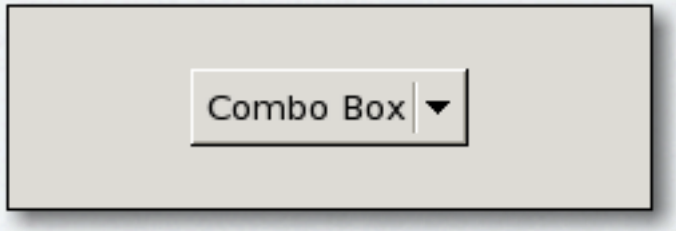


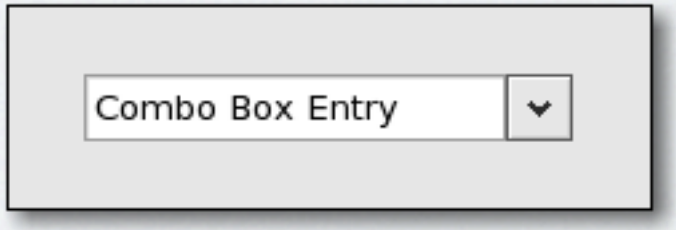
GtkComboBox

Uses model-view architecture, so that model and display can be customized.

Can display flat list or tree structure, for example.

GtkComboBoxEntry allows for editable text.





GtkUIManager

Provides a way to construct a user interface (menus and toolbars) from one or more XML UI definitions.

The most remarkable feature of is that it can overlay a set of menu items and tool items over another one, and de-merge them later.

```
<ui>
  <menubar name='MenuBar'>
    <menu action='FileMenu'>
      <menuitem action='New'/>
      <menuitem action='Open'/>
      <menuitem action='Save'/>
      <menuitem action='SaveAs'/>
      <separator/>
      <menuitem action='Quit'/>
    </menu>
 ...
```

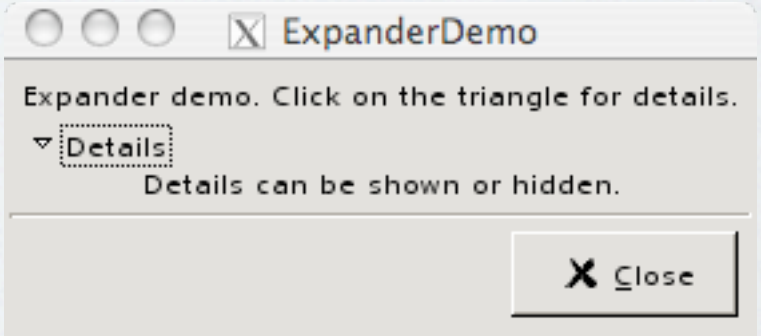
GtkIconView

Displays a list model as a grid of icons with labels.

Allows navigation and selection with arrows keys and rubber-band selection.

GtkExpander

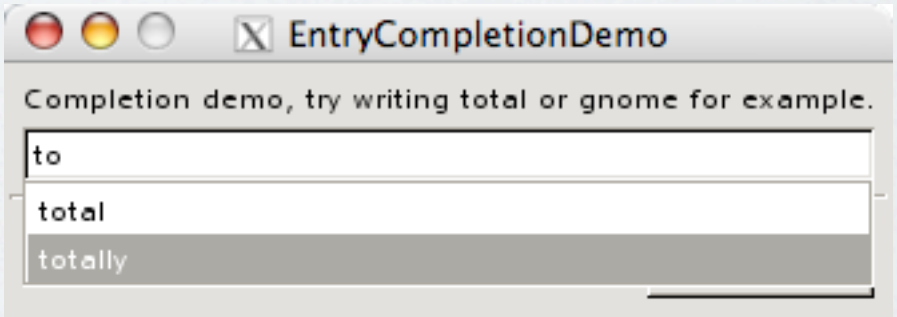
Provides a way to hide and show a child widget by clicking on the expander triangle.



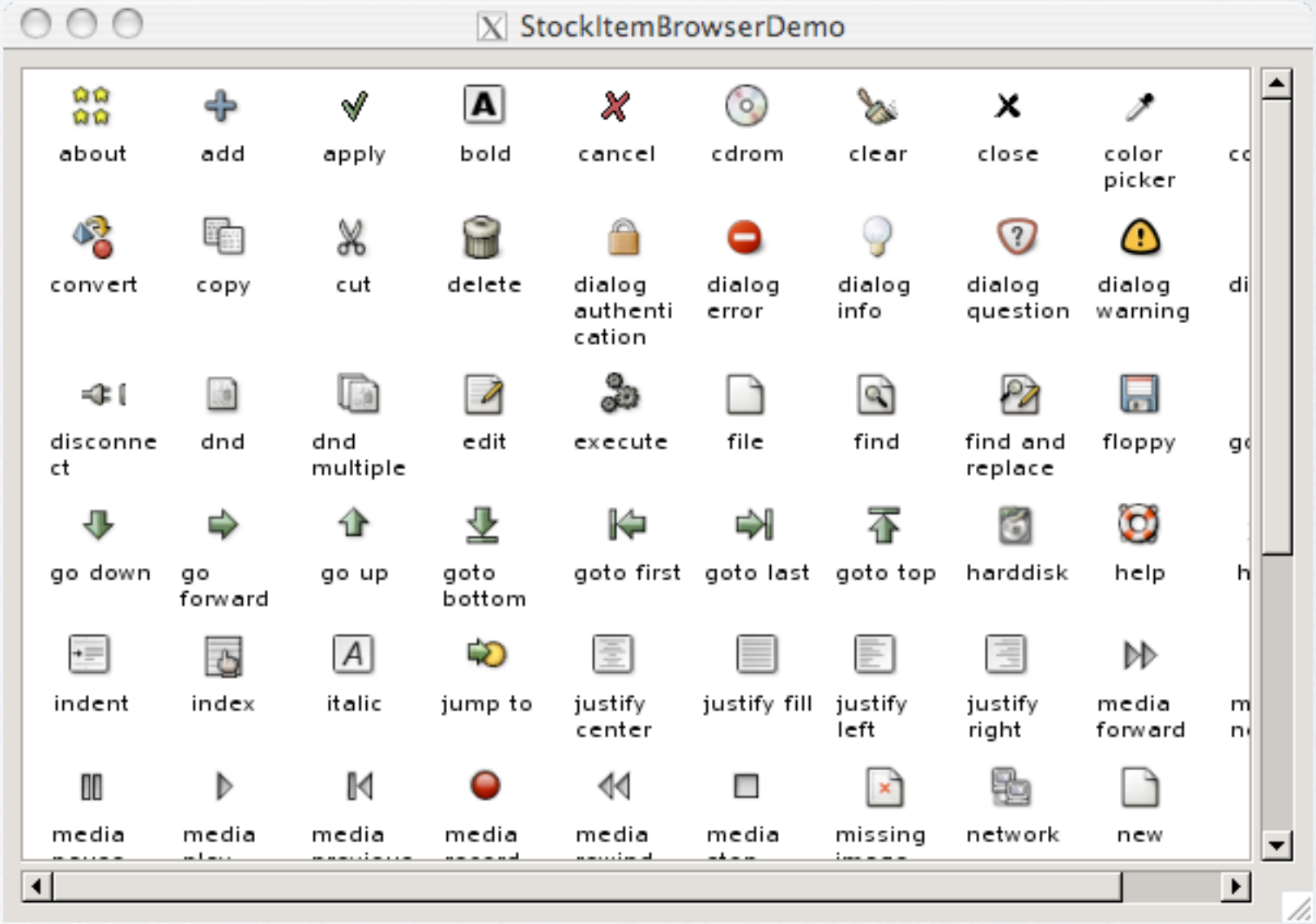
GtkEntryCompletion

Adds completion functionality to GtkEntry with custom layout and matching.

Can “remember” new entries and also display accelerator actions in the dropdown.



Stock Items



PHP-GTK 2 Changes

Backwards compatibility is mostly preserved.

But some incompatible changes may have to be introduced.

- internal extension name is ‘php-gtk’ instead of ‘gtk’
- shared library is called php_gtk2.so (dll)

PHP-GTK 2 Changes

Loading PHP-GTK with `dl()` is possible, but problems may result in some edge cases.

Loading via INI mechanism is preferable.

```
extension = php_gtk2.so
```

Use of Exceptions

PHP 5 supports exceptions, which we can take advantage of. The question is, what sort of conditions should generate an exception.

At the current point of development, exceptions are thrown:

- in object constructors on any sort of error
- in methods that use GError mechanism, such as `GdkPixbuf::new_from_file()`
- in codepage ⇔ UTF-8 conversions

Use of Exceptions

Constructor Exceptions

```
try {
    $store = new GtkListStore(GdkPixbuf::gtype, Gtk::TYPE_PHP_OBJECT);
} catch (PhpGtkConstructException $e) {
    echo $e->message();
}
```

GError Exceptions

```
try {
    $chooser = new GtkFileChooserWidget();
    $chooser->add_shortcut_folder($folder);
} catch (PhpGtkGErrorException $e) {
    die($e->message . "\n");
}
```

Constants

Globals constants are no more. This is one of those incompatible changes.

Instead, all enumerations, flags, and other constants are partitioned by top-level module class: Gtk, Gdk, Pango, Atk and the extension ones.

```
GTK_STATE_PRELIGHT → Gtk::STATE_PRELIGHT
GDK_2BUTTON_PRESS → Gdk::2BUTTON_PRESS
```

Connecting to Signals

PHP-GTK 1 had two connection methods():

- `connect()`
- `connect_object()`

The latter one was semantically confusing and will be deprecated.

PHP-GTK will have three connection methods. All will take custom user parameters which are passed at the end.

Connecting to Signals

`connect()` - connect normally, passing signal widget and signal-specific parameters to callback

```
function on_inserted($entry, $start, $end, $my_data)
{ ... }

$entry = new GtkEntry();
$entry->connect('delete-text', 'on_inserted', true);
```

Connecting to Signals

`connect_swapped()` - substitute user-specified object instead of the signal one, and pass additional signal-specific parameters

```
function on_clicked($button)
{
  // here $button is $button2
}

$button1 = new GtkButton('Button 1');
$button2 = new GtkButton('Button 2');
$button1->connect_swapped('clicked', 'on_clicked', $button2);
```

Connecting to Signals

`connect_simple()` - do not pass signal object or signal parameters

```
function exit()
{
    gtk::main_quit();
}

$window = new GtkWindow();
$window->connect_simple('destroy', 'exit');
$button = new GtkButton('Quit');
$window->add($button);

// connect to built-in method
$button->connect('clicked', array($window, 'destroy'));
```

PHP-GTK Fixes

In PHP-GTK 1, due to problems with the underlying object system, allocated memory could not be properly released at the end of object's lifetime. Long-running scripts could consume a lot of memory.

Now PHP's object store is much more flexible and all these problems have been eradicated. Objects are destroyed as soon as they stop being used.

```
while (1) {
    $pixbuf = GdkPixbuf::new_from_file($file);
}
```

PHP-GTK Fixes

PHP-GTK 1 required assigning objects instantiated with the `new` operator by reference, except for a few cases:

- using new in function parameters
- assigning to globals from inside functions
- assigning to object properties
- yuck!

Very confusing and prone to errors.

With PHP-GTK 2 you can use regular assignment.

PHP-GTK Fixes

In PHP-GTK 1, this example would result in hard error.

```
$text = $button->child->get_text();
```

You had to do some code gymnastics.

```
$child = button->child;
$text = $child->get_text();
```

Has been fixed in PHP 5. Even this works:

```
$text = $button->get_child()->get_text();
```

“Real Code” Example

Let’s look at stock item browser demo, ported from PyGtk.

Illustrates some important principles of new model-view architecture behind behind lists, trees, and other widgets.

And also a good way to see what stock items are available.

“Real Code” Example

We need a class to store information about a stock item.

```
class StockItemInfo {
    public $stock_id = '';
    public $stock_item = null;
    public $small_icon = null;
    public $constant = '';
    public $accel_str = '';
  ...
}
```

“Real Code” Example

We need to create a data model. We will have two columns in the model: a PHP object, and a string.

```
private function create_model()
{
  $store = new GtkListStore(Gtk::TYPE_PHP_VALUE, Gtk::TYPE_STRING);
  ...
```

“Real Code” Example

Now we iterate through all the stock IDs and lookup information such as their icons and associated accelerators.

```
ids = Gtk::stock_list_ids();
sort($ids);

foreach ($ids as $id) {
    $info = new StockItemInfo($id);
    $stock_item = Gtk::stock_lookup($id);
...
    $info->small_icon = $this->render_icon($info->stock_id, $size);
...
    $info->accel_str = '<'.Gtk::accelerator_get_label($info-
>stock_item[3], $info->stock_item[2]).'>';
...
```

“Real Code” Example

Then we insert the information into the data store. We do this using ‘iterators’.

```
// still in create_model()

    $iter = $store->append();
    $store->set($iter, 0, $info, 1, $id);
  } // end foreach()

  return $store;
}
```

“Real Code” Example

Once we have the model, we can create the widget that will display it.

```
model = $this->create_model();
$treeview = new GtkTreeView($model);
```

“Real Code” Example

The view widget will display “columns” of data. Each column can display one or more fields from the model.

```
$column = new GtkTreeViewColumn();
$column->set_title('Icon and Constant');
```

“Real Code” Example

The actual rendering of data is done by a cell renderer. The first column of our view will display the stock item icon and the constant used to access it.

```
$cell_renderer = new GtkCellRendererPixbuf();
$column->pack_start($cell_renderer, false);
$column->set_attributes($cell_renderer, 'stock-id', 1);
```

“Real Code” Example

Setting attributes is not the only way to give the data to the renderer.

```
$cell_renderer = new GtkCellRendererText();
$column->pack_start($cell_renderer, true);
$column->set_cell_data_func($cell_renderer, 'constant_setter');

function constant_setter($column, $cell, $model, $iter)
{
    $info = $model->get_value($iter, 0);
    $cell->set_property('text', $info->constant);
}
```

Finally, we append the column to the view.

```
$treeview->append_column($column);
```

“Real Code” Example

The rest of the columns will display a single piece of information, so we use a shortcut.

```
$cell_renderer = new GtkCellRendererText();
$treeview->insert_column_with_data_func(-1, 'Label',
    $cell_renderer, 'label_setter');

$cell_renderer = new GtkCellRendererText();
$treeview->insert_column_with_data_func(-1, 'Accelerator',
    $cell_renderer, 'accel_setter');

$cell_renderer = new GtkCellRendererText();
$treeview->insert_column_with_data_func(-1, 'ID', $cell_renderer,
    'id_setter');
```

“Real Code” Example

We’d like to do something when user clicks on a row, so we set up a selection object.

```
$selection = $treeview->get_selection();
$selection->set_mode(Gtk::SELECTION_SINGLE);

$selection->connect('changed',
                    array($this, 'on_selection_changed'));

function on_selection_changed($selection)
{
    $treeview = $selection->get_tree_view();
    list($model, $iter) = $selection->get_selected();
    if ($iter) {
       // something is selected
    } else {
       // no selection
    }
}
```

“Real Code” Example

That’s it! Now you know enough to be dangerous.

Distribution

Most of distribution mechanism efforts have focused on Windows.

- NSIS
- SetupStream32
- PriadoBlender
- RoadSend?

Future

API Completeness

Providing abilities to:

- write custom models and cell renderers
- creating new custom widgets with signals and properties
- override virtual methods
- implement interfaces

Performance optimizations

And much more

Thank You!

have a nice day