

# PHP-GTK 2

Andrei Zmievski  
Yahoo! Inc

PHP Québec Conference ~ March 31, 2006



# Agenda

What is PHP-GTK?

Gtk+ 2 Changes

PHP-GTK 2 Changes

Code Examples

Future Work

# PHP-GTK

An extension interfacing with GTK+ library

Allows development of client-side GUI applications

Works on multiple platforms (Unix, Win32, Mac)

Very large extension

- ✓ 211 classes
- ✓ 2305 methods

# PHP-GTK

First release in 2001

Since then many small and large applications have appeared

It even has a job market now!

# PHP-GTK

## Main concepts

- ✓ Widgets - user interface elements
- ✓ Signals / callbacks - event system

# PHP-GTK

PHP-GTK 1 was hindered by PHP 4's object model

- ✓ No proper object destruction
- ✓ Crippled object overloading
- ✓ And slow object implementation in general

Partially provided motivation for new object model in PHP 5

# PHP-GTK 2

Based on PHP 5.1 and GTK+ 2.6

All the latest bells and whistles

Sets the baseline for future development

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

# PHP-GTK 2

Almost a complete rewrite

Mostly backwards compatible

Compatibility broken when necessary

Documentation being rewritten as well



# PHP 5

Flexible and powerful object model

Finally has destructors!

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

Has many hooks for extensibility

# PHP-GTK 2 Changes

Backwards compatibility is mostly preserved.

But some incompatible changes had to be introduced.

- ✓ internal extension name is 'php-gtk' instead of 'gtk'
- ✓ shared library is called php\_gtk2.so (dll)
- ✓ global constants are no more

# 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`  $\Leftrightarrow$  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 is deprecated in favor of a simpler one.

# 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_deleted', true);
```



# Connecting to Signals

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

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

$window = new GtkWindow();
$window->connect_simple('destroy', 'quit');
$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

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 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();
```

# Gtk+2

New flexible and powerful type system

Good introspection and extensibility

Can be easily mapped onto PHP object model

# 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 international 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 smooth 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

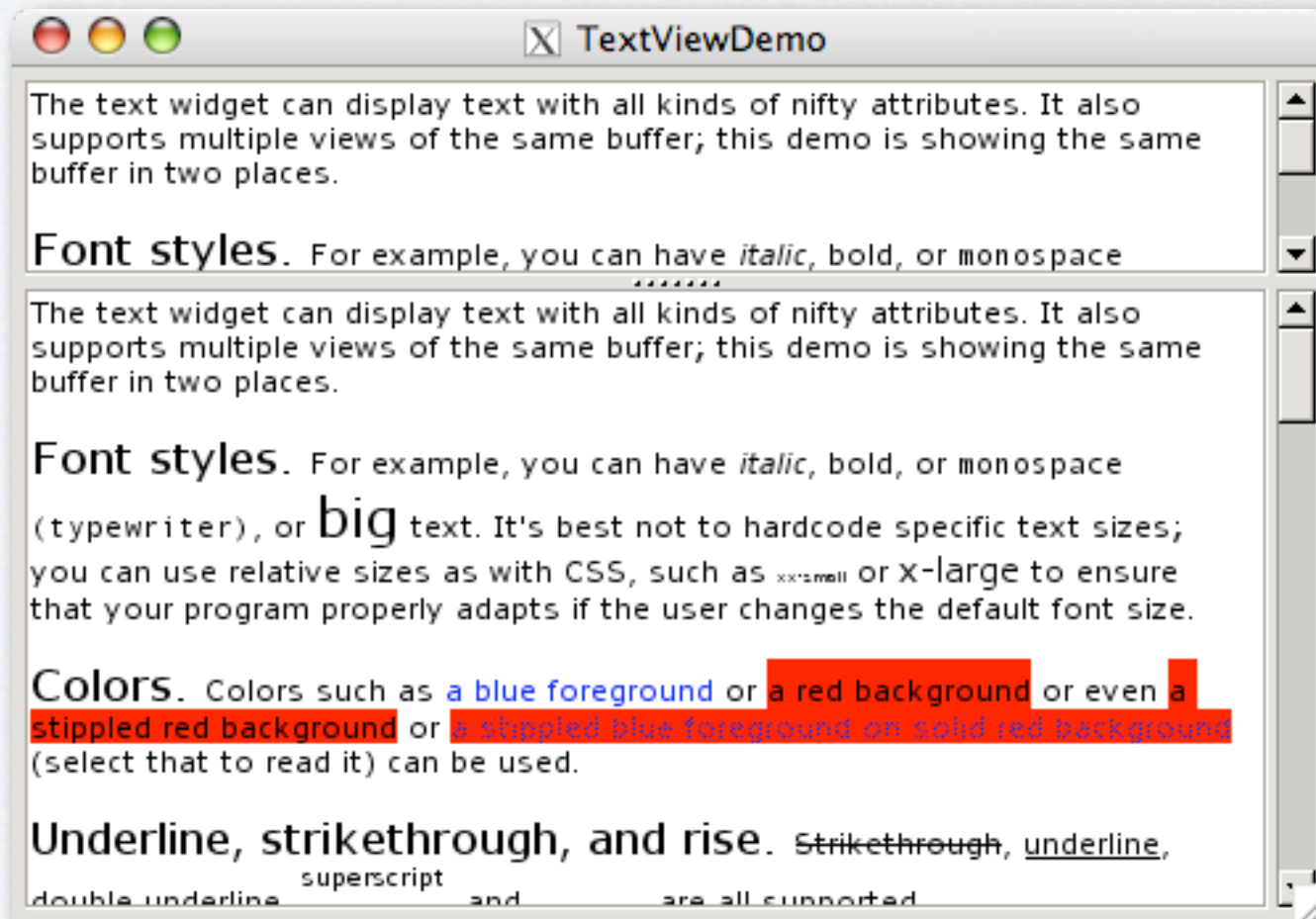
Selection drag-n-drop

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

Hooks for implementing undo

# New Widget Gallery

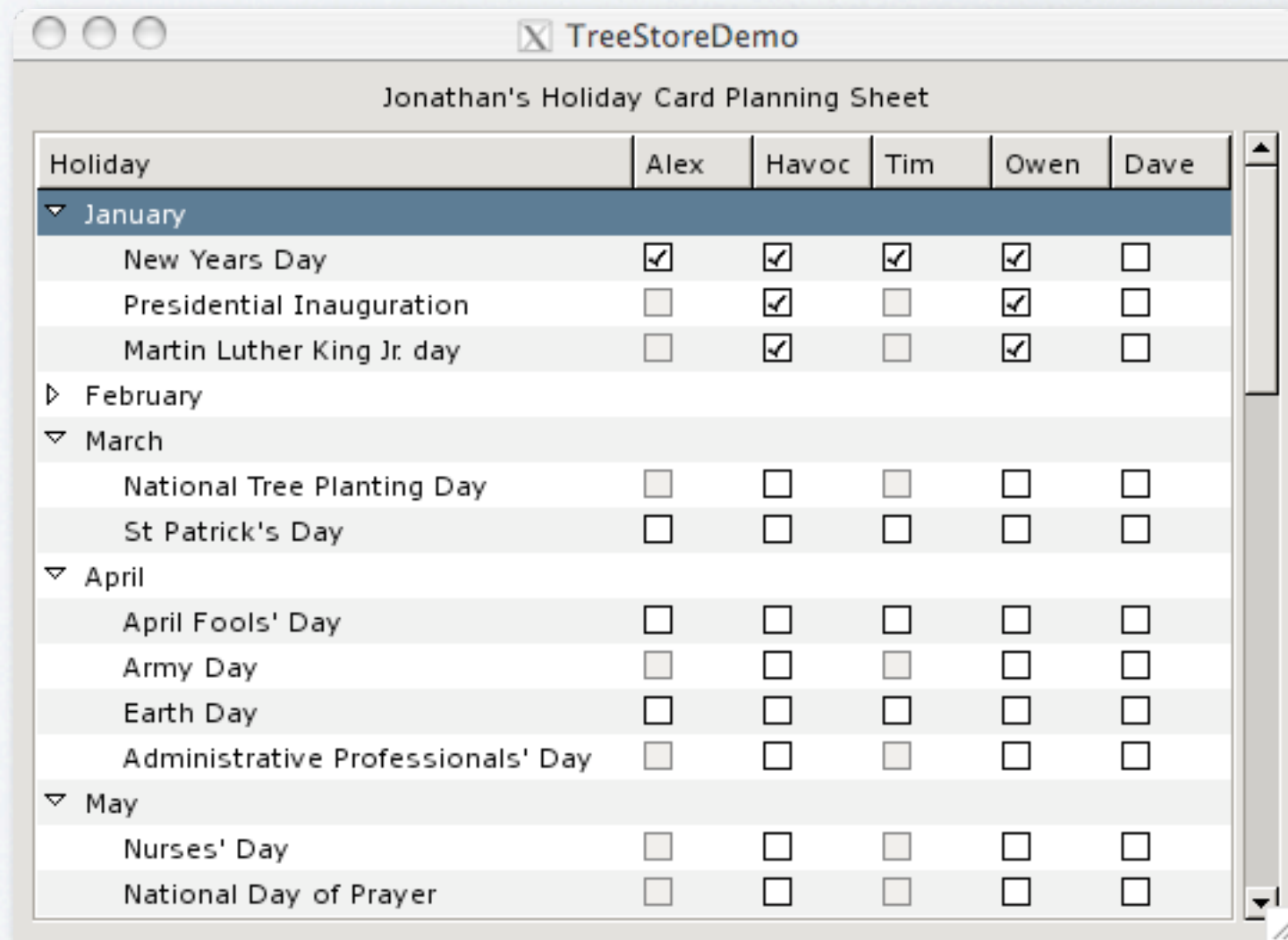
# GtkTextView



# GtkTextView



# GtkTreeView



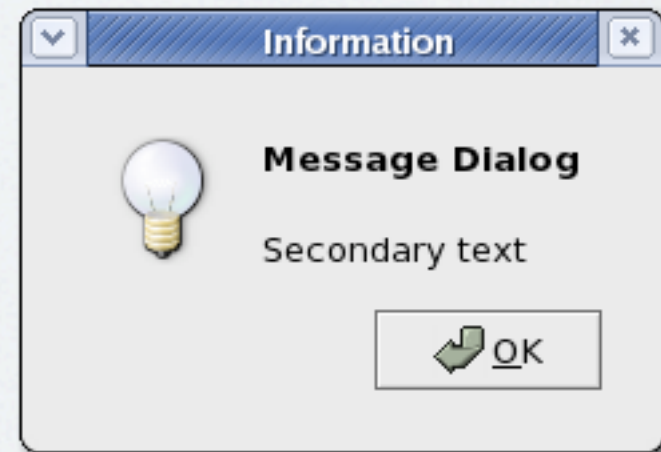
Jonathan's Holiday Card Planning Sheet

Holiday	Alex	Havoc	Tim	Owen	Dave
▼ January					
New Years Day	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Presidential Inauguration	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Martin Luther King Jr. day	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
▶ February					
▼ March					
National Tree Planting Day	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
St Patrick's Day	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
▼ April					
April Fools' Day	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Army Day	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Earth Day	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Administrative Professionals' Day	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
▼ May					
Nurses' Day	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
National Day of Prayer	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

# 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.

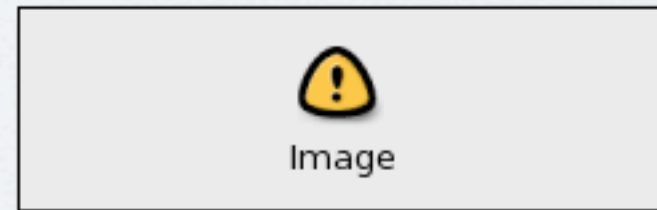
Allows for display of license and program credits.



# 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.

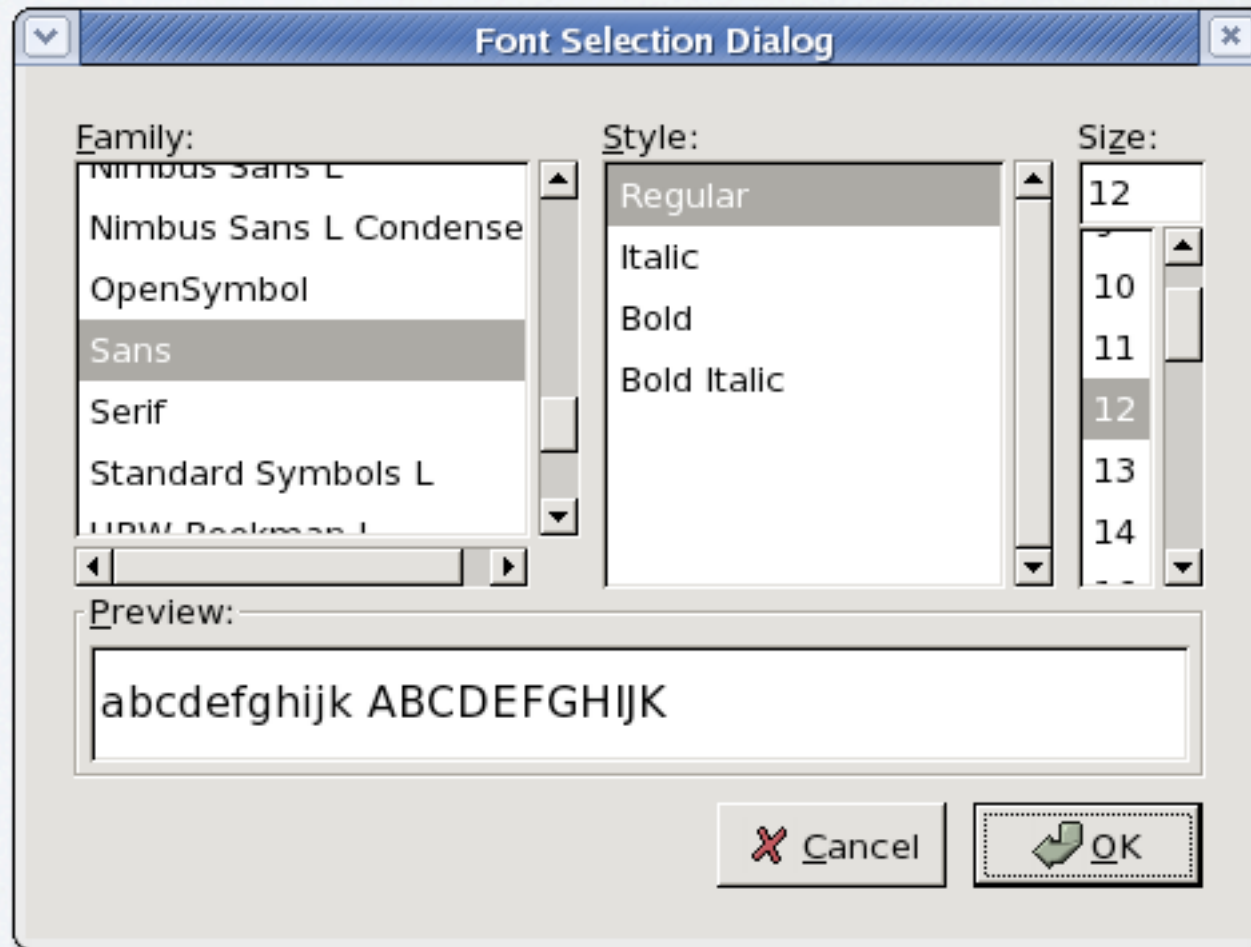
Custom cut, copy, & paste actions are no longer out of reach.

# GtkFontButton

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



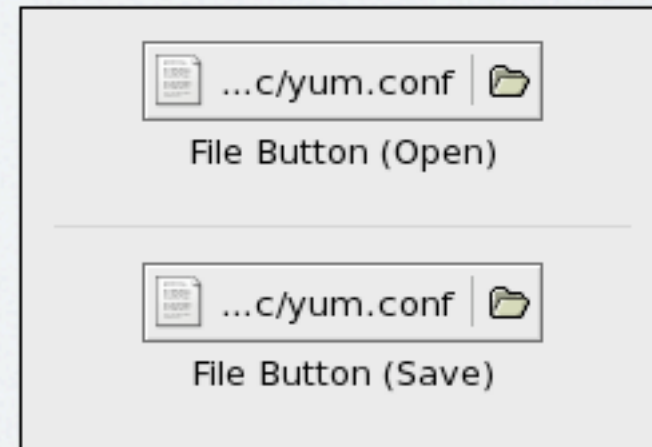
# GtkFontSelectionDialog



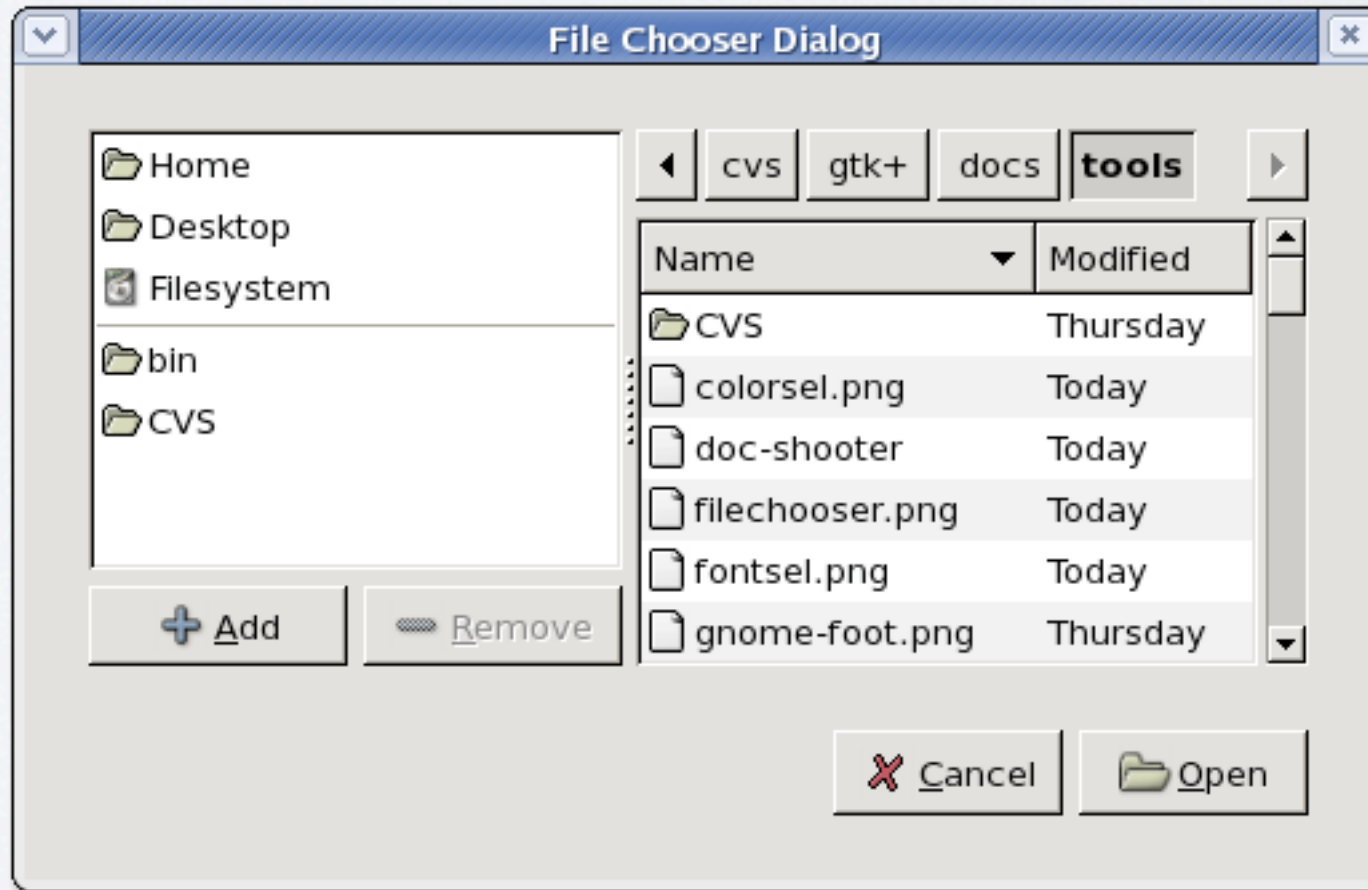
# GtkFileChooserButton

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

File selection dialog can have custom preview widget.

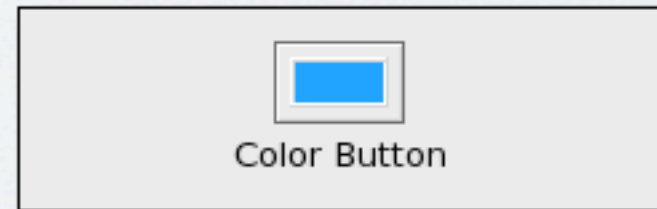


# GtkFileChooserDialog



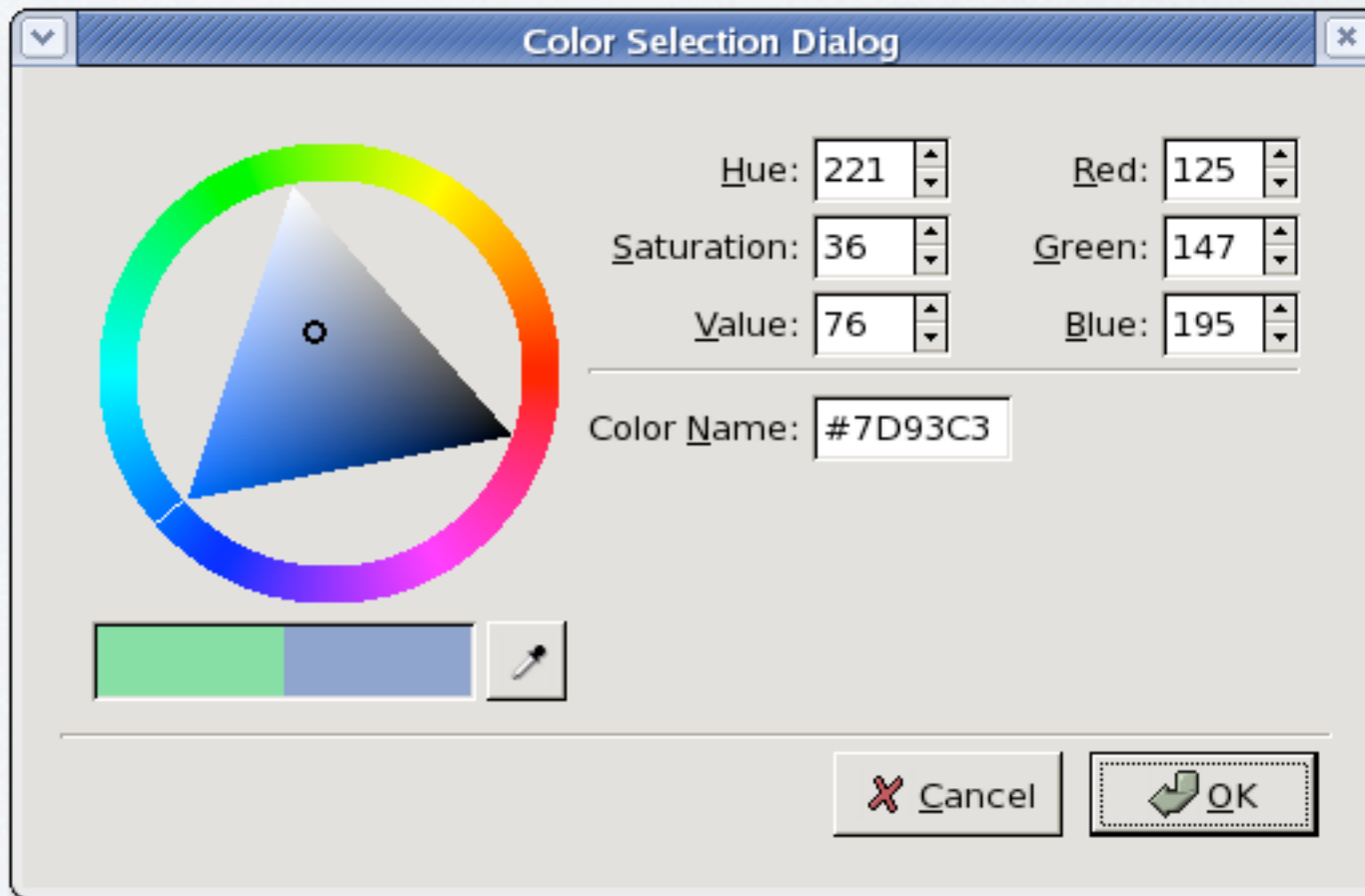
# GtkColorButton

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



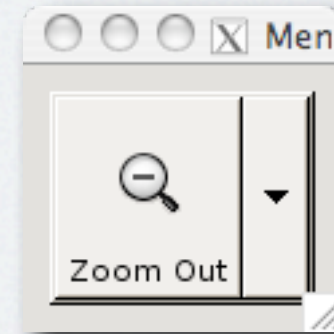


# GtkColorSelectionDialog



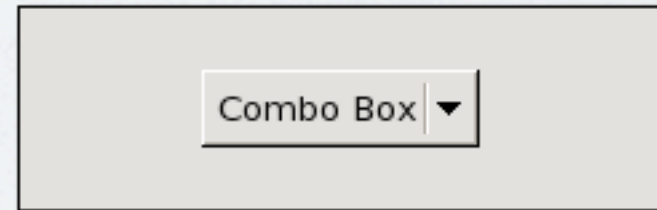
# GtkMenuToolButton

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



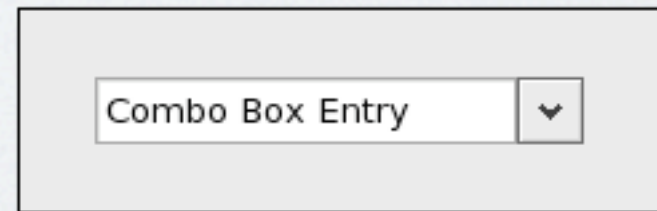
# GtkComboBox

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

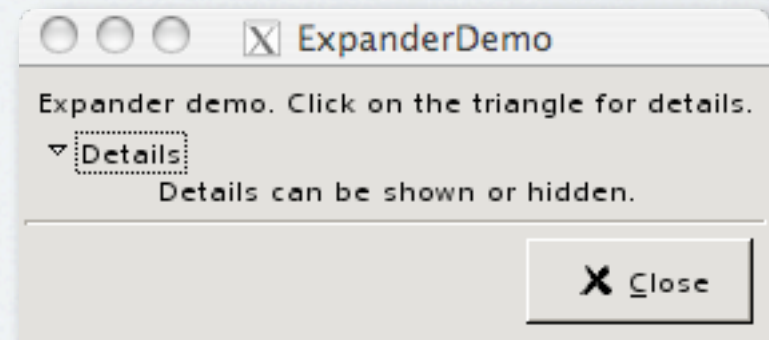
Displays a list model as a grid of icons with labels.

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



# GtkExpander

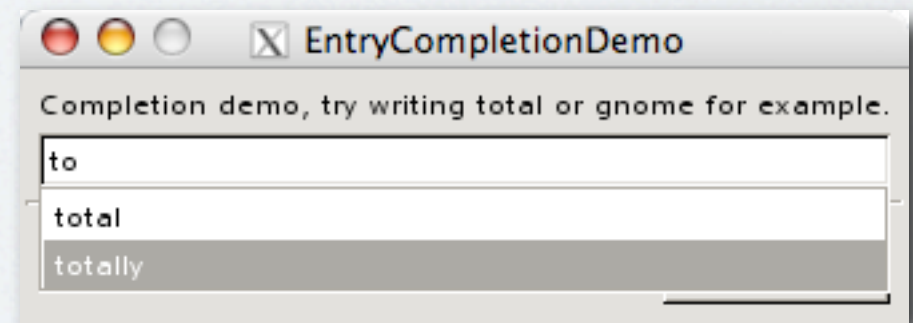
Provides a way to hide and show a child widget by clicking on the expander triangle.



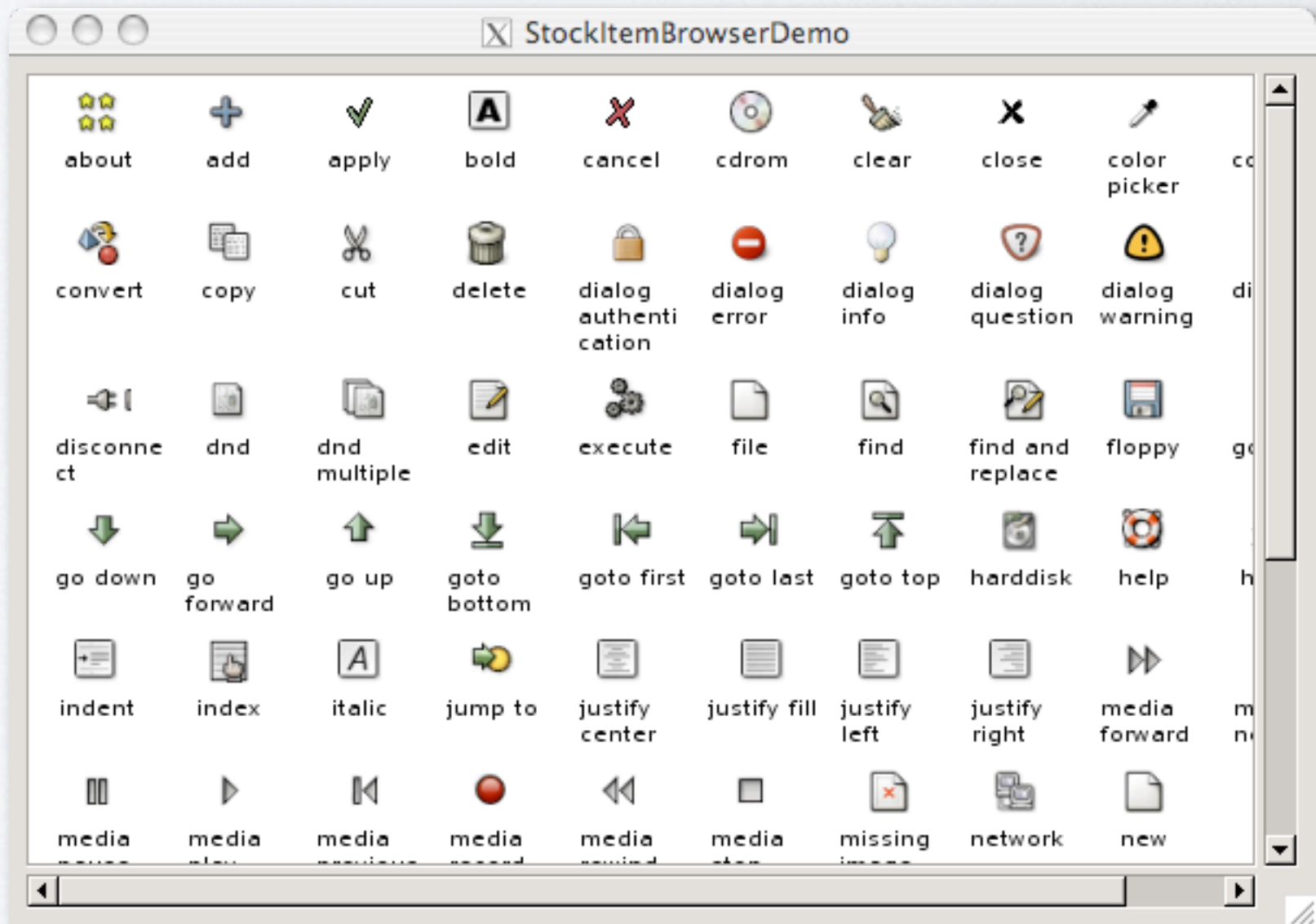
# GtkEntryCompletion

Adds completion functionality to GtkEntry with custom layout and matching.

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



# Stock Items





# Model/View Example

Let's look at a simple example that illustrates some important principles of new model-view architecture behind lists, trees, and other widgets.

Warning, code ahead...

# Model/View Example

We need to create a data model. We will use `GtkListStore` since we do not need hierarchical data. The model will have several fields.

```
$store = new GtkListStore(  
    GdkPixbuf::gtype, // icon  
    Gtk::TYPE_STRING, // title  
    Gtk::TYPE_STRING, // category  
    Gtk::TYPE_BOOLEAN, // works?  
    Gtk::TYPE_LONG      // development progress (percent)  
);
```

# Model/View Example

We will need another list store that will contain items for the combo box. It only needs one field.

```
$categories = new GtkListStore(Gtk::TYPE_STRING);  
$categories->append(array('Editors'));  
$categories->append(array('Games'));  
$categories->append(array('Utilities'));  
$categories->append(array('Office'));
```

# Model/View Example

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

```
//View is needed to display them  
$view = new GtkTreeView($store);
```

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

# Model/View Example

The actual rendering of data is done by an object called **cell renderer**.

- ✓ knows how to draw information of specific type
- ✓ configured through **attributes**, the values for which come from data model

```
// Column: Icon
$rendererPixbuf = new GtkCellRendererPixbuf();
$columnIcon = new GtkTreeViewColumn(
    'Icon',           // title
    $rendererPixbuf, // the renderer
    'pixbuf',        // use this property
    0                // data is in this model field
);
$view->append_column($columnIcon);
```

# Model/View Example

Here is the next column. Here we want to make text editable.

```
// Column: Name
$rendererText = new GtkCellRendererText();
$rendererText->set_property('editable', true);
$rendererText->connect('edited', 'name_edited', $store);

$columnName = new GtkTreeViewColumn(
    'Name',          // title
    $rendererText,  // the renderer
    'text',         // use this property
    1               // data is in this model field
);
$view->append_column($columnName);
```

# Model/View Example

This is the function that will be called we finish editing the name.

```
function name_edited($renderer, $row, $text, $model) {  
    // set model field 'name' (1)  
    // to the uppercased edited text  
    $model->set($model->get_iter($row), 1, strtoupper($text));  
}
```

Or, through magic of PHP object model:

```
function name_edited($renderer, $row, $text, $model) {  
    $model[$row][1] = strtoupper($text);  
}
```

# Model/View Example

Category column is a little more complex, since we need to hook up combo box with another model.

```
// Column: Category
$rendererCategory = new GtkCellRendererCombo();
$rendererCategory->set_property('model'      , $categories);
$rendererCategory->set_property('text-column', 0);
$rendererCategory->set_property('editable'   , true);
$rendererCategory->connect('edited', 'category_edited', $store);

$columnCategory = new GtkTreeViewColumn(
    'Category',          // title
    $rendererCategory,  // the renderer
    'text',              // use this property
    2                    // data is in this model column
);
$columnCategory->set_expand(true);
$columnCategory->set_property('resizable', TRUE);
$view->append_column($columnCategory);
```



# Model/View Example

The toggle renderer uses boolean model field. We connect a callback to change the value on click.

```
// Column: Works?
$rendererWorks = new GtkCellRendererToggle();
$rendererWorks->set_property('activatable', true);
$rendererWorks->connect('toggled', 'works_toggled', $store);

$columnWorks = new GtkTreeViewColumn(
    'Works?',           // title
    $rendererWorks,    // the renderer
    'active',          // use this property
    3                   // data is in this model field
);
$view->append_column($columnWorks);

function works_toggled($renderer, $row, $store) {
    $store[$row][3] = !$store[$row][3]; // invert value
}
```

# Model/View Example

The progress info column is easy.

```
// Column: Development progress
$rendererProgress = new GtkCellRendererProgress();

$columnProgress = new GtkTreeViewColumn(
    'Progress',          // title
    $rendererProgress, // the renderer
    'value',            // use this property
    4                   // data is in this model column
);
$columnProgress->set_expand(true);
$view->append_column($columnProgress);
```

# Model/View Example

Finally, we can add some data to the model. We use the view to render a stock icon.

```
$store->append(array(
    $view->render_icon(Gtk::STOCK_ABOUT,
                      Gtk::ICON_SIZE_SMALL_TOOLBAR), //icon
    'Game_Minesweeper', // name
    'Games',           // category
    true,              // works?
    100                // progress (in percent)
));

// and so on...
```

# Model/View Example

All that's left is set up the main window and put the view in it.

```
$wnd = new GtkWindow();  
$wnd->set_default_size(400, -1);  
$wnd->set_title('Applications');  
$wnd->connect_simple('destroy', array('Gtk', 'main_quit'));  
$wnd->add($view);  
$wnd->show_all();  
Gtk::main();
```

# Model/View Example

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

# Deployment

Once you have an app, you need an easy way to deploy it.

On Win32 platforms, the best option is Gnope.

Allows transparent installation of PHP-GTK 2 and applications through Gnope PEAR channel.

<http://gnope.org/>

# Future

API Completeness

Providing abilities to:

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

Performance optimizations

And much more

# Resources

<http://gtk.php.net/>

<http://gtk.php.net/manual/>

<http://www.gnope.org/>

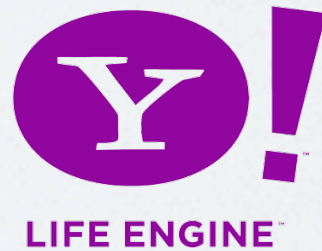
<http://www.cweiske.de/phpgtk.htm>

<http://mail.gnome.org/archives/gtk-list/>

**#php-gtk** IRC channel on Freenode network



Thank You!  
Merci!



<http://www.gravitonic.com/talks/>