

PHP ON THE DESKTOP AND PHP-GTK2

Elizabeth Marie Smith – Zend Uncon 2008



- AKA auroraeosrose
- Have a very very common name
- Was an English Major (minor in Sociology)
- Started using PHP for a Manga/Anime fan site (still alive)
- Have been using PHP since 4 beta 1 (23-Jul-1999)
- Have been using C since Nov of 2006
- Love the color Pink
- I hate computers
- I love programming
- Thinks open source on Windows is a “gateway drug”
- Works at OmniTI (<http://omniti.com>)
- Am a contributor on various open source projects (including PHP, PECL and PHP-GTK)



IS THE DESKTOP DEAD?



- People aren't always online
- Some applications would be bandwidth prohibitive
- Some data is too sensitive to be passed on the wire
- Embedded OSs don't always work well with websites
- Browsers are just Quirky (*I'm looking at you, IE6, IE7, Opera...*)
- Current Desktop RIA tools (AIR, Xulrunner, Mozilla Prism) are still young (and buggy)



- No Compiling
- Instant Changes
- No learning curve for a new language
- Plug into PHP extensions
- Easy to Use
- Fast to Write
- Use already familiar libraries and tools



- Speed
- Additional Extensions/Libraries needed
- Speed
- Distribution (phar is helpful there)
- Speed
- Security (of code – source code encoders might help here)
- No threading

See the theme?



- PHP does not care about HTML (really)
- PHP works through SAPIs (Server Application Programming Interface) – for the desktop the CLI (Command Line Interface) SAPI is our friend
- CLI has no headers, CLI responds to no requests
- You need a way to make GUI's if you don't want a console app – Just like we wrap other C/C++ libraries we can wrap GUI libraries



- Well Established
 - PHP-GTK2
 - Winbinder
- Works, but not Complete
 - PHP-QT (no windows support)
 - WxWidgets (only windows support)
 - Win::Gui (part of Win::API)

Bottom line? For Cross-Platform PHP-GTK works



- GTK was Gimp Tool Kit (long long ago)
- GTK is GUI Toolkit for Gnome (if you use gnome, you have it already)
- GTK has multiple backends (for native Windows and MacOSX apps – you don't need X on Mac anymore)
- To use PHP-GTK you need PHP CLI, GTK, and the PHP-GTK extension
- <http://oops.opsat.net/doc/install.html> - how to install (windows is unzip and run, mac has an installer, ubuntu has .debs – yes this should be in the php-gtk docs, we need volunteers who aren't afraid of xml)



LET'S DO PHP-GTK



- Main Loop
- Signals & Callbacks
- Widgets
- OOP to the Max
- Visibility



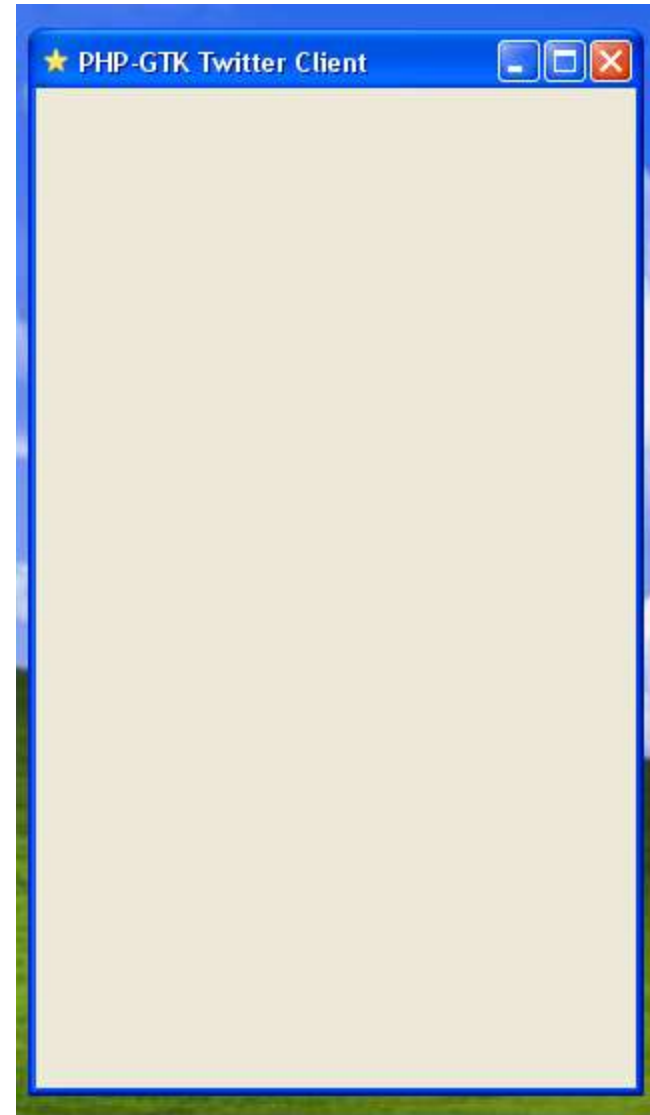
Create Our initial Window

```
<?php
class Php_Gtk_Twitter_Client extends GtkWindow {

    public function __construct() {
        parent::__construct();
        $this->set_icon($this->render_icon(
            Gtk::STOCK_ABOUT, Gtk::ICON_SIZE_DIALOG));
        $this->set_size_request(300, 500);
        $this->set_title('PHP-GTK Twitter Client');
        $this->connect_simple('destroy', array('Gtk', 'main_quit'));
    }
}

$window = new Php_Gtk_Twitter_Client;
$window->show_all();
Gtk::main();
```

Resulting window



- Specialty Widgets are Cool
- GtkStatusIcon – 2.10+
- Know your Hierarchy
- Stock Icons
- Callback “bubbling”
- timeouts (timers)



StockItemBrowserDemo

Icon and Constant	Label	Accelerator	ID
Gtk::STOCK_ABOUT	_About		gtk-about
Gtk::STOCK_ADD	_Add		gtk-add
Gtk::STOCK_APPLY	_Apply		gtk-apply
Gtk::STOCK_BOLD	_Bold		gtk-bold
Gtk::STOCK_CANCEL	_Cancel		gtk-cancel
Gtk::STOCK_CDROM	_CD-Rom		gtk-cdrom
Gtk::STOCK_CLEAR	_Clear		gtk-clear
Gtk::STOCK_CLOSE	_Close	<Ctrl+W>	gtk-close
Gtk::STOCK_COLOR_PICKER			gtk-color-picker
Gtk::STOCK_CONNECT	C_onnect		gtk-connect
Gtk::STOCK_CONVERT	_Convert		gtk-convert
Gtk::STOCK_COPY	_Copy	<Ctrl+C>	gtk-copy
Gtk::STOCK_CUT	Cu_t	<Ctrl+X>	gtk-cut
Gtk::STOCK_DELETE	_Delete		gtk-delete
Gtk::STOCK_DIALOG_AUTHENTICATION			gtk-dialog-authentication
Gtk::STOCK_DIALOG_ERROR	Error		gtk-dialog-error
Gtk::STOCK_DIALOG_INFO	Information		gtk-dialog-info
Gtk::STOCK_DIALOG_QUESTION	Question		gtk-dialog-question
Gtk::STOCK_DIALOG_WARNING	Warning		gtk-dialog-warning
Gtk::STOCK_DIRECTORY			gtk-directory
Gtk::STOCK_DISCARD	_Discard		gtk-discard
Gtk::STOCK_DISCONNECT	_Disconnect		gtk-disconnect
Gtk::STOCK_DND			gtk-dnd
Gtk::STOCK_DND_MULTIPLE			gtk-dnd-multiple
Gtk::STOCK_EDIT	_Edit		gtk-edit
Gtk::STOCK_EXECUTE	_Execute		gtk-execute
Gtk::STOCK_FILE			gtk-file
Gtk::STOCK_FIND	_Find	<Ctrl+F>	gtk-find
Gtk::STOCK_FIND_AND_REPLACE	Find and _Replace	<Ctrl+R>	gtk-find-and-replace
Gtk::STOCK_FLOPPY	_Floppy		gtk-floppy
Gtk::STOCK_FULLSCREEN	_Fullscreen		gtk-fullscreen
Gtk::STOCK_GO_BACK	_Back		gtk-go-back
Gtk::STOCK_GO_DOWN	_Down		gtk-go-down
Gtk::STOCK_GO_FORWARD	_Forward		gtk-go-forward
Gtk::STOCK_GO_UP	_Up		gtk-go-up
Gtk::STOCK_GOTO_BOTTOM	_Bottom		gtk-goto-bottom
Gtk::STOCK_GOTO_FIRST	_First		gtk-goto-first
Gtk::STOCK_GOTO_LAST	_Last		gtk-goto-last
Gtk::STOCK_GOTO_TOP	_Top		gtk-goto-top
Gtk::STOCK_HARDDISK	_Harddisk		gtk-harddisk

Selection Info

Item and Icon

About

Gtk::STOCK_ABOUT

gtk-about



Minimize to Status Icon

```
<?php
class Php_Gtk_Twitter_Icon extends GtkStatusIcon {

    protected $alive;
    protected $lockout;

    public function __construct() {
        parent::__construct();

        $this->set_from_stock(Gtk::STOCK_ABOUT);
        $this->set_tooltip('PHP-GTK Twitter Client');
        while (Gtk::events_pending() || Gdk::events_pending()) {
            Gtk::main_iteration_do(true);
        }
        $this->is_ready();
        return;
    }

    public function is_ready() {
        $this->alive = true;
        if ($this->lockout < 5 && !$this->is_embedded()) {
            Gtk::timeout_add(750,array($this,'is_ready'));
            ++$this->lockout;
            $this->alive = false;
        } else if (!$this->is_embedded()) {
            die("Error: Unable to create Tray Icon. Please insure that your system's tray is enabled.\n");
            $this->alive = false;
        }

        return;
    }

    public function activate_window($icon, $window) {
        if ($window->is_visible()) {
            $window->hide();
        } else {
            $window->deiconify();
            $window->show();
        }
    }
}

class Php_Gtk_Twitter_Client extends GtkWindow {

    protected $statusicon;

    public function __construct() {
        parent::__construct();
        $this->set_icon($this->render_icon(Gtk::STOCK_ABOUT, Gtk::ICON_SIZE_DIALOG));
        $this->set_size_request(300, 500);
        $this->set_title('PHP-GTK Twitter Client');
        $this->connect_simple('destroy', array('Gtk', 'main_quit'));

        $this->statusicon = new Php_Gtk_Twitter_Icon;
        $this->statusicon->connect('activate', array($this->statusicon,
            'activate_window'), $this);
        $this->set_skip_taskbar_hint(true);
        $this->connect('window-state-event', array($this, 'minimize_to_tray'));
    }

    public function minimize_to_tray($window, $event) {
        if ($event->changed_mask == Gdk::WINDOW_STATE_ICONIFIED &&
            $event->new_window_state & Gdk::WINDOW_STATE_ICONIFIED) {
            $window->hide();
        }
        return true; //stop bubbling
    }
}

$window = new Php_Gtk_Twitter_Client;
$window->show_all();
Gtk::main();
```

Resulting Window



- Remember this is PHP
- Treeview has a model and view
- ListStore is a single level model
- Treeviews need Columns
- Columns need Renderers
- Different types of Renderers



I really don't believe in wheel inventing, especially when I have little experience with the subject. However there is an issue – every existing twitter API uses curl – and I don't want to depend on an extension that isn't always installed.

```
protected function process($url, $date = 0, $type = 'GET', $data = null) {
    // add caching header
    $this->headers[0] = 'If-Modified-Since: ' . date(DATE_RFC822, $date);

    $options = array(
        'http' => array(
            'method' => $type,
            'header' => $this->headers
        )
    );
    if (!is_null($data)) {
        $options['http']['content'] = http_build_query($data);
    }
    $context = stream_context_create($options);
    if ($this->username && $this->password) {
        $base = 'http://' . urlencode($this->username) . ':' . urlencode($this->password)
            . '@twitter.com/';
    } else {
        $base = 'http://twitter.com/';
    }
    set_error_handler(array($this,'swallow_error'));
    $string = file_get_contents($base . $url, false, $context);
    restore_error_handler();
    return json_decode($string);
}
```



Put a Treeview in the Window – this goes in __construct

```
$this->temp = sys_get_temp_dir() . 'php-gtk-twitter-api-cache\\';
if (!file_exists($this->temp)){
    mkdir($this->temp, null, true);
}

$this->twitter = new Php_Gtk_Twitter_Api;

// User image pixbuf, user image string, user name,
// user id, text, favorited, created_at, id
$store = new GtkListStore(GdkPixbuf::gtype, GObject::TYPE_STRING,
    GObject::TYPE_STRING, GObject::TYPE_LONG, GObject::TYPE_STRING,
    GObject::TYPE_BOOLEAN, GObject::TYPE_STRING, GObject::TYPE_LONG);
$store->set_sort_column_id(7, Gtk::SORT_DESCENDING);

$list = $this->twitter->get_public_timeline();

// stuff the store
foreach($list as $object) {
    $store->append(array(null, $object->user->profile_image_url, $object->user->name,
        $object->user->id, $object->text, $object->favorited, $object->created_at,
        $object->id));
}

$this->public_timeline_timeout = Gtk::timeout_add(61000,
    array($this, 'update_public_timeline')); // every 60 seconds

$scrolled = new GtkScrolledWindow();
$scrolled->set_policy(Gtk::POLICY_NEVER, Gtk::POLICY_ALWAYS);
$this->add($scrolled);
$this->treeview = new GtkTreeView($store);
$scrolled->add($this->treeview);
$this->treeview->set_property('headers-visible', false);
$this->treeview->set_rules_hint(true);

$picture_renderer = new GtkCellRendererPixbuf();
$picture_column = new GtkTreeViewColumn('Picture', $picture_renderer, 'pixbuf', 0);
$picture_column->set_cell_data_func($picture_renderer, array($this, 'show_user'));
$this->treeview->append_column($picture_column);

$message_renderer = new GtkCellRendererText();
$message_renderer->set_property('wrap-mode', Gtk::WRAP_WORD);
$message_renderer->set_property('wrap-width', 200);
$message_renderer->set_property('width', 10);

$message_column = new GtkTreeViewColumn('Message', $message_renderer);
$message_column->set_cell_data_func($message_renderer,
    array($this, 'message_markup'));
$this->treeview->append_column($message_column);

$this->treeview->set_resize_mode(Gtk::RESIZE_IMMEDIATE);
```

Callbacks for showing Images and Messages

```
public function show_user($column, $cell, $store, $position) {
    $pic = $store->get_value($position, 1);
    $name = $this->temp . md5($pic);
    if (isset($this->pic_queue[$name])) {
        return;
    } elseif (isset($this->pic_cached[$name])) {
        $store = $this->treeview->get_model();
        if (is_null($store->get_value($position, 0))){
            $pixbuf = GdkPixbuf::new_from_file($name . '.jpg');
            $store->set($position, 0, $pixbuf);
            $cell->set_property('pixbuf', $pixbuf);
        }
        return;
    }
    $this->pic_queue[$name] = array('name' => $name, 'url' => $pic,
        'pos' => $position, 'cell' => $cell);
    if (empty($this->load_images_timeout)) {
        $this->load_images_timeout = Gtk::timeout_add(500, array($this, 'pic_queue'));
    }
}

public function pic_queue() {
    $pic = array_shift($this->pic_queue);
    if (empty($pic)) {
        $this->load_images_timeout = null;
        return true;
    }
    if (!file_exists($pic['name'])) {
        file_put_contents($pic['name'] . '.jpg', file_get_contents($pic['url']));
        $this->pic_cached[$pic['name']] = $pic['url'];
    }
    return true; // keep the timeout going
}

public function message_markup($column, $cell, $store, $position) {
    $user = utf8_decode($store->get_value($position, 2));
    $message = utf8_decode($store->get_value($position, 4));
    $time = $this->distance($store->get_value($position, 6));

    $message = htmlspecialchars_decode($message, ENT_QUOTES);
    $message = str_replace(array('@' . $user, '&nbsp;', '&#x27;', '&#x26;'), array('<span foreground="#FF6633">@' .
        $user . '</span>', '&#x27;', '&#x26;'), $message);
    $cell->set_property('markup', "<b>$user</b>:\n$message\n<small>$time</small>");
}

protected function distance($from) {
    $minutes = round(abs(time() - strtotime($from)) / 60);

    switch (true) {
        case ($minutes == 0):
            return 'less than 1 minute ago';
        case ($minutes < 1):
            return '1 minute ago';
        case ($minutes <= 55):
            return $minutes . ' minutes ago';
        case ($minutes <= 65):
            return 'about 1 hour ago';
        case ($minutes <= 1439):
            return 'about ' . round((float) $minutes / 60.0) . ' hours';
        case ($minutes <= 2879):
            return '1 day ago';
        default:
            return 'about ' . round((float) $minutes / 1440) . ' days ago';
    }
}
```





- GTK is not “pixel perfect”
- Packing is not as hard as it looks
- Containers can hold one or many
- Remember that a container expands to the size of it’s contents



Pack the Statusbar, Treeview, and Toolbar in a VBox

```
$this->statusbar = new GtkStatusBar();  
  
// Create a toolbar with login button  
$tb = new GtkToolBar();  
$tb->set_show_arrow(false);  
$this->loginbutton = GtkToolButton::new_from_stock(Gtk::STOCK_JUMP_TO);  
$this->loginbutton->set_label('Login');  
$this->loginbutton->connect_simple('clicked', array($this, 'login'));  
$tb->insert($this->loginbutton, -1);  
  
// logout button  
$this->logoutbutton = GtkToolButton::new_from_stock(Gtk::STOCK_CLOSE);  
$this->logoutbutton->set_label('Logout');  
$this->logoutbutton->connect_simple('clicked', array($this, 'logout'));  
$tb->insert($this->logoutbutton, -1);  
$this->logoutbutton->set_sensitive(false);
```

With Toolbar, Treeview, and Statusbar

```
$vbox = new GtkVBox();  
$this->add($vbox);  
$vbox->pack_start($tb, false, false);  
$vbox->pack_start($scrolled);  
$vbox->pack_start($this->statusbar, false, false);
```

Update the Header and Public Timeline using a Gtk::timeout

```
publicfunction update_public_timeline() {  
    $this->pic_queue = array();  
    $list = $this->twitter->get_public_timeline();  
    $this->statusbar->pop(1);  
    $this->statusbar->push(1, 'last updated ' . date('Y-m-d H:i') .  
        ' ' . count($list) . ' new tweets');  
    $store = $this->treeview->get_model();  
    $store->clear();  
    foreach ($list as $object) {  
        $store->append(array(null,  
            $object->user->profile_image_url, $object->user->name,  
            $object->user->id, $object->text, $object->favorited,  
            $object->created_at, $object->id));  
    }  
    return true;  
}
```

Create Statusbar and Toolbar



- Dialogs are cool
- Dialogs are usually modal, but not always
- Dialogs can be very general or very specific
- You can put anything inside one



```

class Php_Gtk_Twitter_Login_Dialog extends GtkDialog {
    protected $emailentry;
    protected $passwordentry;

    public function __construct($parent){
        parent::__construct('Login to Twitter', $parent, Gtk::DIALOG_MODAL,
            array(
                Gtk::STOCK_OK, Gtk::RESPONSE_OK,
                Gtk::STOCK_CANCEL, Gtk::RESPONSE_CANCEL));

        $table = new GtkTable();
        $email = new GtkLabel('Email:');
        $table->attach($email, 0, 1, 0, 1);
        $password = new GtkLabel('Password:');
        $table->attach($password, 0, 1, 1, 2);
        $this->emailentry = new GtkEntry();
        $table->attach($this->emailentry, 1, 2, 0, 1);
        $this->passwordentry = new GtkEntry();
        $table->attach($this->passwordentry, 1, 2, 1, 2);
        $this->passwordentry->set_visibility(false);
        $this->vbox->add($table);
        $this->errorlabel = new GtkLabel();
        $this->vbox->add($this->errorlabel);
        $this->show_all();
    }

    public function check_login($twitter){
        $this->errorlabel->set_text("");
        $email = $this->emailentry->get_text();
        $password = $this->passwordentry->get_text();
        if (empty($password) || empty($password)){
            $this->errorlabel->set_markup(
                '<span color="red">Name and Password must be entered</span>');
            return false;
        }
        if ($twitter->login($email, $password)) {
            return true;
        } else {
            $this->errorlabel->set_markup(
                '<span color="red">Authentication Error</span>');
            return false;
        }
    }
}
    
```

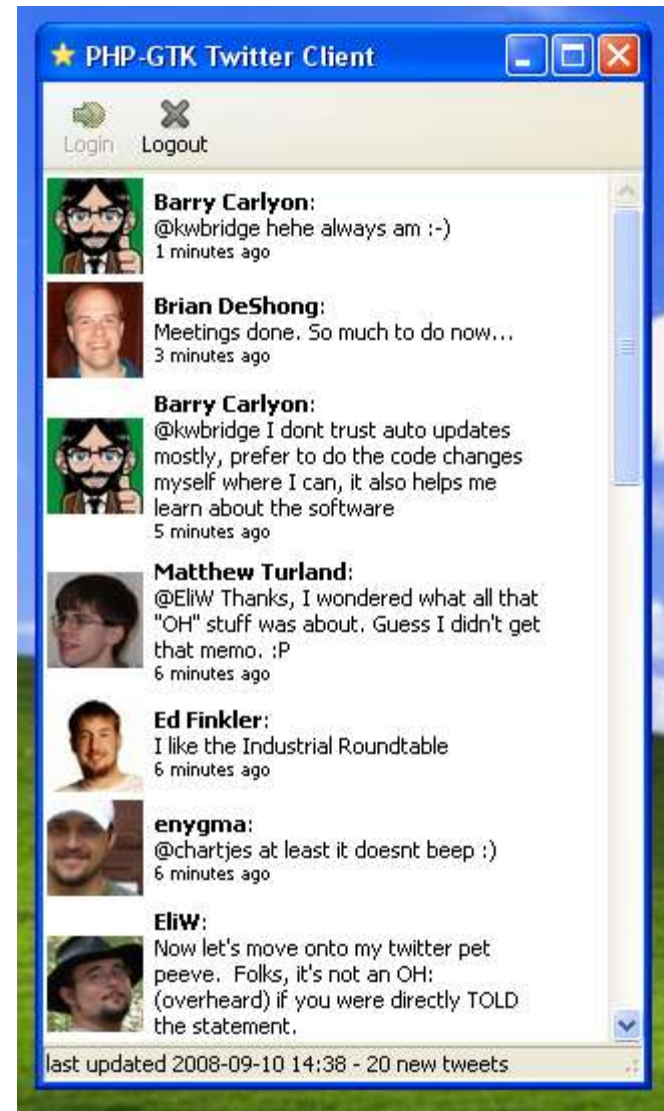


Callbacks

```
public function login(){
    if (!empty($this->load_images_timeout)){
        Gtk::timeout_remove($this->load_images_timeout);
        $readd = true;
    }
    Gtk::timeout_remove($this->public_timeline_timeout);
    $login = new Php_Gtk_Twitter_Login_Dialog($this);
    while ($response = $login->run()){
        if ($response == GTK::RESPONSE_CANCEL ||
            $response == GTK::RESPONSE_DELETE_EVENT){
            if (isset($readd)){
                $this->load_images_timeout =
                    Gtk::timeout_add(500,array($this,'pic_queue'));
            }
            $this->public_timeline_timeout =
                Gtk::timeout_add(61000,
                    array($this,'update_public_timeline'));
            $login->destroy();
            break;
        } elseif ($response == GTK::RESPONSE_OK){
            if ($login->check_login($this->twitter)){
                $this->logoutbutton->set_sensitive(true);
                $this->loginbutton->set_sensitive(false);
                $login->destroy();
                $this->public_timeline_timeout = Gtk::timeout_add(
                    61000,array($this,'update_timeline'));
                $this->load_images_timeout = Gtk::timeout_add(500,
                    array($this,'pic_queue'));
                $this->treeview->get_model()->clear();
                $this->pic_queue = array();
                $this->pic_cached = array();
                $this->update_timeline();
                break;
            }
        }
    }
}

public function logout(){
    $this->twitter->logout();
    $this->logoutbutton->set_sensitive(false);
    $this->loginbutton->set_sensitive(true);
    $this->public_timeline_timeout = Gtk::timeout_add(61000,
        array($this,'update_public_timeline')); // every 60 seconds
    $this->pic_queue = array();
    $this->pic_cached = array();
    $this->update_public_timeline();
}
```

After Login



- GTKEntry
- Basic Data Entry – activates on return, can set maximum length allowed
- Simple label for messages – could use a dialog or other method of informing the user



Packing it in

```

$ vbox = new GtkVBox();
$ this->add($ vbox);
$ vbox->pack_start($ tb, false, false);
$ scrolled = new GtkScrolledWindow();
$ scrolled->set_policy(Gtk::POLICY_NEVER, Gtk::POLICY_ALWAYS);
$ vbox->pack_start($ scrolled);
$ vbox->pack_start(new GtkLabel('What are you doing?'), false, false);
$ vbox->pack_start($ this->updateentry, false, false);
$ vbox->pack_start($ this->entrystatus, false, false);
$ vbox->pack_start($ this->statusbar, false, false);

```

Creating the Entries

```

// Create an update area
$ this->updateentry = new GtkEntry();
$ this->updateentry->set_max_length(140);
$ this->updateentry->set_sensitive(false);
$ this->updateentry->connect('activate',
    array($ this, 'send_update'));
$ this->entrystatus = new GtkLabel();

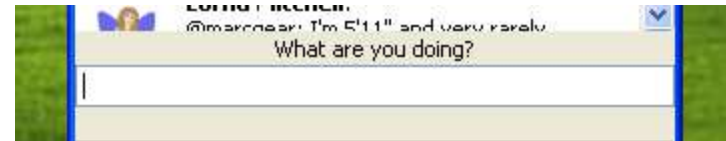
```

The Callback on Activate

```

public function send_update($entry) {
    if ($this->twitter->send($entry->get_text())) {
        $this->entrystatus->set_text('Message Sent');
        $this->update_timeline();
        $this->updateentry->set_text("");
    } else {
        $this->entrystatus->set_markup('
<span color="red">Error Sending Message - Try Again</span>');
    }
}

```





```

class Php_Gtk_Twitter_Api {
    protected $login = false;
    protected $username;
    protected $password;

    protected $cached_public;
    protected $cached_public_timestamp = 0;

    protected $lastid = 0;

    protected $headers = array
        (, 'X-Twitter-Client: PHP-GTK Twitter Client',
          'X-Twitter-Client-Version: 0.1.0-dev',
          'X-Twitter-Client-URL: http://elizabethmariesmith.com');

    public function login($username, $password) {
        $this->username = $username;
        $this->password = $password;
        $worked = $this->process('account/verify_credentials.json');
        if ($worked && $worked->authorized == true) {
            $this->login = true;
            return true;
        }
        return false;
    }

    public function logout() {
        $this->username = null;
        $this->password = null;
        $this->login = false;
        $this->process('account/end_session', 0, 'POST');
    }

    public function get_public_timeline() {
        if ($this->cached_public_timestamp < time()) {
            $this->cached_public =
                json_decode(file_get_contents(
                    'http://twitter.com/statuses/public_timeline.json'));
            $this->cached_public_timestamp = time() + 60;
            // caches every 60 seconds
        }
        return $this->cached_public;
    }
}

```

```

    public function get_timeline() {
        if ($this->login && $this->can_call()) {
            if (empty($this->lastid)) {
                $data = $this->process('statuses/friends_timeline.json');
            } else {
                $data = $this->process('statuses/friends_timeline.json', $this->lasttime,
                    'GET', array('since_id' => $this->lastid));
            }
            if ($data) {
                $this->lastid = $data[0]->id;
            }
            $this->lasttime = time();
            return $data;
        }
    }

    public function send($message) {
        if ($this->login && $this->can_call()) {
            $data = $this->process('statuses/update.json', 0, 'POST', array('status' => $message));
            if ($data) {
                return true;
            }
            return false;
        }
    }

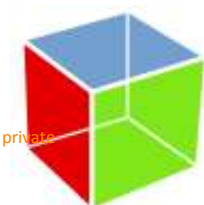
    protected function can_call() {
        if (!$this->login) {
            return false;
        }
        $worked = $this->process('account/rate_limit_status.json');
        return ($worked->remaining_hits > 1);
    }

    protected function process($url, $date = 0, $type = 'GET', $data = null) {
        // add caching header
        $this->headers[0] = 'If-Modified-Since: ' . date(DATE_RFC822, $date);

        $options = array(
            'http' => array(
                'method' => $type,
                'header' => $this->headers
            )
        );
        if (!is_null($data)) {
            $options['http']['content'] = http_build_query($data);
        }
        $context = stream_context_create($options);
        if ($this->username && $this->password) {
            $base = 'http://' . urlencode($this->username) . ':' . urlencode($this->password)
                . '@twitter.com/';
        } else {
            $base = 'http://twitter.com/';
        }
        set_error_handler(array($this, 'swallow_error'));
        $string = file_get_contents($base . $url, false, $context);
        restore_error_handler();
        return json_decode($string);
    }

    public function swallow_error($errno, $errstr) { // this should be treated as private
}

```



```

class Php_Gtk_Twitter_Login_Dialog extends GtkDialog {
    protected $emailentry;
    protected $passwordentry;

    public function __construct($parent){
        parent::__construct('Login to Twitter',
            $parent, Gtk::DIALOG_MODAL,
            array (Gtk::STOCK_OK, Gtk::RESPONSE_OK,
                Gtk::STOCK_CANCEL, Gtk::RESPONSE_CANCEL));
        $table = new GtkTable();
        $email = new GtkLabel('Email:');
        $table->attach($email, 0, 1, 0, 1);
        $password = new GtkLabel('Password:');
        $table->attach($password, 0, 1, 1, 2);
        $this->emailentry = new GtkEntry();
        $table->attach($this->emailentry, 1, 2, 0, 1);
        $this->passwordentry = new GtkEntry();
        $table->attach($this->passwordentry, 1, 2, 1, 2);
        $this->passwordentry->set_visibility(false);
        $this->vbox->add($table);
        $this->errorlabel = new GtkLabel();
        $this->vbox->add($this->errorlabel);
        $this->show_all();
    }

    public function check_login($twitter){
        $this->errorlabel->set_text("");
        $email = $this->emailentry->get_text();
        $password = $this->passwordentry->get_text();
        if (empty($password) || empty($password)){
            $this->errorlabel->set_markup(
                '<span color="red">Name and Password must be entered</span>');
            return false;
        }
        if ($twitter->login($email, $password)) {
            return true;
        } else {
            $this->errorlabel->set_markup(
                '<span color="red">Authentication Error</span>');
            return false;
        }
    }
}

```

```

class Php_Gtk_Twitter_Icon extends GtkStatusIcon {
    protected $alive;
    protected $lockout;

    public function __construct(){
        parent::__construct();

        $this->set_from_stock(Gtk::STOCK_ABOUT);
        $this->set_tooltip('PHP-GTK Twitter Client');
        while (Gtk::events_pending() || Gdk::events_pending()){
            Gtk::main_iteration_do(true);
        }
        $this->is_ready();
        return;
    }

    public function is_ready(){
        $this->alive = true;
        if ($this->lockout < 5 && !$this->is_embedded()){
            Gtk::timeout_add(750, array($this, 'is_ready'));
            ++$this->lockout;
            $this->alive = false;
        } else if (!$this->is_embedded()){
            die ("Error: Unable to create Tray Icon.
                Please insure that your system's tray is enabled.\n");
            $this->alive = false;
        }
        return;
    }

    public function activate_window($icon, $window){
        if ($window->is_visible()){
            $window->hide();
        } else {
            $window->deiconify();
            $window->show();
        }
    }
}

```



Final Code, all 492 lines

```

class Php_Gtk_Twitter_Client extends GtkWindow {
    protected $statusicon;
    protected $twitter;
    protected $treeview;
    protected $statusbar;

    protected $temp;
    protected $pic_queue = array();
    protected $pic_cached = array();

    protected $public_timeline_timeout;
    protected $load_images_timeout;

    public function __construct() {
        parent::__construct();
        $this->set_icon($this->render_icon(Gtk::STOCK_ABOUT, Gtk::ICON_SIZE_DIALOG));
        $this->set_size_request(300, 500);
        $this->set_title('PHP-GTK Twitter Client');
        $this->connect_simple('destroy', array('Gtk', 'main_quit'));

        $this->statusicon = new Php_Gtk_Twitter_Icon;
        $this->statusicon->connect('activate', array($this->statusicon,
            'activate_window'), $this);
        $this->set_skip_taskbar_hint(true);
        $this->connect('window-state-event', array($this, 'minimize_to_tray'));

        $this->temp = sys_get_temp_dir() . 'php-gtk-twitter-api-cache\\';
        if (!file_exists($this->temp)) {
            mkdir($this->temp, null, true);
        }

        $this->twitter = new Php_Gtk_Twitter_Api;
        $this->statusbar = new GtkStatusBar();

        // Create a toolbar with login button
        $tb = new GtkToolbar();
        $tb->set_show_arrow(false);
        $this->loginbutton = GtkToolButton::new_from_stock(Gtk::STOCK_JUMP_TO);
        $this->loginbutton->set_label('Login');
        $this->loginbutton->connect_simple('clicked', array($this, 'login'));
        $tb->insert($this->loginbutton, -1);

        // logout button, hide it
        $this->logoutbutton = GtkToolButton::new_from_stock(Gtk::STOCK_CLOSE);
        $this->logoutbutton->set_label('Logout');
        $this->logoutbutton->connect_simple('clicked', array($this, 'logout'));
        $tb->insert($this->logoutbutton, -1);
        $this->logoutbutton->set_sensitive(false);

        // Create an update area
        $this->updateentry = new GtkEntry();
        $this->updateentry->set_max_length(140);
        $this->updateentry->set_sensitive(false);
        $this->updateentry->connect('activate', array($this, 'send_update'));
        $this->entrystatus = new GtkLabel();

        // User image pixbuf, user image string, user name, user id, text, favorited, created_at, id
        $store = new GtkListStore(GdkPixbuf::gtype, GObject::TYPE_STRING, GObject::TYPE_STRING,
            GObject::TYPE_LONG, GObject::TYPE_STRING, GObject::TYPE_BOOLEAN,
            GObject::TYPE_STRING,
            GObject::TYPE_LONG);
        $store->set_sort_column_id(7, Gtk::SORT_DESCENDING);

        $list = $this->twitter->get_public_timeline();
        $this->statusbar->push(1, 'last updated ' . date('Y-m-d H:i') . ' - ' . count($list) . ' new tweets');
    }

```

```

foreach($list as $object) {
    $store->append(array(null, $object->user->profile_image_url,
        $object->user->name, $object->user->id,
        $object->text, $object->favorited, $object->created_at,
        $object->id));
}

```

```

$this->public_timeline_timeout = Gtk::timeout_add(61000,,
    array($this, 'update_public_timeline')); // every 60 seconds

```

```

$vbbox = new GtkVBox();
$this->add($vbbox);
$vbbox->pack_start($tb, false, false);
$scrolled = new GtkScrolledWindow();
$scrolled->set_policy(Gtk::POLICY_NEVER, Gtk::POLICY_ALWAYS);
$vbbox->pack_start($scrolled);
$this->treeview = new GtkTreeView($store);
$scrolled->add($this->treeview);
$this->treeview->set_property('headers-visible', false);
$this->treeview->set_rules_hint(true);
$vbbox->pack_start(new GtkLabel('What are you doing?'), false, false);
$vbbox->pack_start($this->updateentry, false, false);
$vbbox->pack_start($this->entrystatus, false, false);
$vbbox->pack_start($this->statusbar, false, false);

```

```

$picture_renderer = new GtkCellRendererPixbuf();
$picture_column = new GtkTreeViewColumn('Picture',
    $picture_renderer, 'pixbuf', 0);
$picture_column->set_cell_data_func($picture_renderer,
    array($this, 'show_user'));
$this->treeview->append_column($picture_column);

```

```

$message_renderer = new GtkCellRendererText();
$message_renderer->set_property('wrap-mode', Gtk::WRAP_WORD);
$message_renderer->set_property('wrap-width', 200);
$message_renderer->set_property('width', 10);

```

```

$message_column = new GtkTreeViewColumn('Message',
    $message_renderer);
$message_column->set_cell_data_func($message_renderer,
    array($this, 'message_markup'));
$this->treeview->append_column($message_column);

```

```

$this->treeview->set_resize_mode(Gtk::RESIZE_IMMEDIATE);

```



```

publicfunction show_user($column, $cell, $store, $position) {
    $pic = $store->get_value($position, 1);
    $name = $this->temp . md5($pic);
    if (isset($this->pic_queue[$name])){
        return;
    } elseif (isset($this->pic_cached[$name])){
        $store = $this->treeview->get_model();
        if (is_null($store->get_value($position, 0))){
            $pixbuf = GdkPixbuf::new_from_file($name . '.jpg');
            $store->set($position, 0, $pixbuf);
            $cell->set_property('pixbuf', $pixbuf);
        }
        return;
    }
    $this->pic_queue[$name] = array('name' => $name, 'url' => $pic,
        'pos' => $position, 'cell' => $cell);
    if (empty($this->load_images_timeout)){
        $this->load_images_timeout = Gtk::timeout_add(500,
            array($this, 'pic_queue'));
    }
}

public function pic_queue(){
    $pic = array_shift($this->pic_queue);
    if (empty($pic)){
        $this->load_images_timeout = null;
        return true;
    }
    if (!file_exists($pic['name'])){
        file_put_contents($pic['name'] . '.jpg',
            file_get_contents($pic['url']));
        $this->pic_cached[$pic['name']] = $pic['url'];
    }
    return true; // keep the timeout going
}

public function message_markup($column, $cell, $store,
    $position) {
    $user = utf8_decode($store->get_value($position, 2));
    $message = utf8_decode($store->get_value($position, 4));
    $time = $this->distance($store->get_value($position, 6));

    $message = htmlspecialchars_decode($message, ENT_QUOTES);
    $message = str_replace(array('@' . $user, '&nbsp;', '&T'), array(
        '<span foreground="#FF6633">@'
        . $user . '</span>', '', '&amp;'), $message);
    $cell->set_property('markup',
        "<b>$user</b>:\n$message\n<small>$time</small>");
}

```

```

protected function distance($from) {
    $minutes = round(abs(time() - strtotime($from)) / 60);

    switch(true) {
        case ($minutes == 0):
            return 'less than 1 minute ago';
        case ($minutes < 1):
            return '1 minute ago';
        case ($minutes <= 55):
            return $minutes . ' minutes ago';
        case ($minutes <= 65):
            return 'about 1 hour ago';
        case ($minutes <= 1439):
            return 'about ' . round((float) $minutes / 60.0) . ' hours';
        case ($minutes <= 2879):
            return '1 day ago';
        default:
            return 'about ' . round((float) $minutes / 1440) . ' days ago';
    }
}

public function minimize_to_tray($window, $event) {
    if ($event->changed_mask == Gdk::WINDOW_STATE_ICONIFIED &&
        $event->new_window_state & Gdk::WINDOW_STATE_ICONIFIED) {
        $window->hide();
    }
    return true; // stop bubbling
}

public function update_public_timeline(){
    $this->pic_queue = array();
    $list = $this->twitter->get_public_timeline();
    $this->statusbar->pop(1);
    $this->statusbar->push(1, 'last updated ' .
        date('Y-m-d H:i') . ' - ' . count($list) . ' new tweets!');
    $store = $this->treeview->get_model();
    $store->clear();
    foreach ($list as $object) {
        $store->append(array(null, $object->user->profile_image_url,
            $object->user->name, $object->user->id, $object->text,
            $object->favorited, $object->created_at,
            $object->id));
    }
    return true;
}

```



Final Code, all 492 lines

```

publicfunction update_timeline(){
    $list = $this->twitter->get_timeline();
    $this->statusbar->pop(1);
    $this->statusbar->push(1, 'last updated ' . date('Y-m-d H:i') . ' - '
        . count($list) . ' new tweets');
    $store = $this->treeview->get_model();
    foreach ($list as $object) {
        $store->append(array(null, $object->user->profile_image_url,
            $object->user->name, $object->user->id, $object->text,
            $object->favorited, $object->created_at,
            $object->id));
    }
    return true;
}

public function login(){
    if (!empty($this->load_images_timeout)){
        Gtk::timeout_remove($this->load_images_timeout);
        $readd = true;
    }
    Gtk::timeout_remove($this->public_timeline_timeout);
    $login = new Php_Gtk_Twitter_Login_Dialog($this);
    while ($response = $login->run()) {
        if ($response == GTK::RESPONSE_CANCEL ||
            $response == GTK::RESPONSE_DELETE_EVENT) {
            if (isset($readd)) {
                $this->load_images_timeout = Gtk::timeout_add(500,
                    array($this, 'pic_queue'));
            }
            $this->public_timeline_timeout = Gtk::timeout_add(61000,
                array($this, 'update_public_timeline')); // every 60 seconds
            $login->destroy();
            break;
        } elseif ($response == GTK::RESPONSE_OK) {
            if ($login->check_login($this->twitter)) {
                $this->logoutbutton->set_sensitive(true);
                $this->loginbutton->set_sensitive(false);
                $login->destroy();
                $this->public_timeline_timeout = Gtk::timeout_add(61000,
                    array($this, 'update_public_timeline')); // every 60 seconds
                $this->load_images_timeout = Gtk::timeout_add(500,
                    array($this, 'pic_queue'));
                $this->treeview->get_model()->clear();
                $this->pic_queue = array();
                $this->pic_cached = array();
                $this->update_public_timeline();
                $this->updateentry->set_sensitive(true);
                break;
            }
        }
    }
}

public function logout(){
    $this->twitter->logout();
    $this->logoutbutton->set_sensitive(false);
    $this->loginbutton->set_sensitive(true);
    $this->public_timeline_timeout = Gtk::timeout_add(61000, array($this,
        'update_public_timeline')); // every 60 seconds
    $this->pic_queue = array();
    $this->pic_cached = array();
    $this->update_public_timeline();
    $this->updateentry->set_sensitive(false);
}

public function send_update($entry) {
    if ($this->twitter->send($entry->get_text())) {
        $this->entrystatus->set_text('Message Sent');
        $this->update_public_timeline();
        $this->updateentry->set_text("");
    } else {
        $this->entrystatus->set_markup('<span color="red">
            Error Sending Message - Try Again</span>');
    }
}

public function __destruct() {
    foreach (scandir($this->temp) as $filename) {
        if ($filename[0] == '.')
            continue;
        if (file_exists($this->temp . $filename)) {
            unlink($this->temp . $filename);
        }
    }
}

$window = new Php_Gtk_Twitter_Client;
$window->show_all();
Gtk::main();

```



- Slides
 - <http://callicore.net/php-gtk/php-on-the-desktop.pdf>
- Code
 - <http://callicore.net/php-gtk/php-gtk-twitter.zip>
- GTK docs
 - <http://gtk.org>
 - <http://pygtk.org>
 - <http://gtk.php.net/docs.php>
 - <http://leonpegg.com/php-gtk-doc/phpweb/>
 - <http://kksou.com>
 - <http://oops.opsat.net>
 - <http://php-gtk.eu>
- Me
 - <http://elizabethmariesmith.com>
 - <http://callicore.net>
 - <http://callicore.net/php-gtk>
 - auroraeosrose@php.net

THANKS

