



Consuming Web Services With PHP-GTK 2

By Scott Mattocks

November 2, 2006 – San Jose, CA

The Online Marketplace for On-Site Technology ServicesSM



OnForceSM
THE POWER OF ON

What's ON the Agenda

- **What is PHP-GTK 2?**
- **Why create a desktop application?**
- **Providing a GUI for a web service**

What is PHP-GTK 2

- **A PHP 5 wrapper around Gtk+2.0**
- **PHP 5**
 - Scripting language normally associated with web development
 - Quick and easy application development
 - Improved Object Handling
- **Gtk+2.0**
 - The Gimp ToolKit
 - C library for for displaying UI elements on screen
 - Used to create standalone desktop applications
 - Improved text handling and internationalization
 - Model/View architecture to separate data from display
- **PHP 5 + Gtk+2.0 = quick and easy development of desktop apps**

Why Create a Desktop Application?

- **Performance**
- **Flexibility**
- **Scalability**
- **Branding**

Why Create a Desktop Application? - Performance

● End User Performance

- Enables higher productivity
 - Narrows user focus to particular tasks
 - Processes can be streamed-lined for experienced users
 - Allows user to work offline
- Not necessary to wait for server response for every action
 - Batching of data for upload
 - Ability to work while processing background tasks
- More powerful UI components
 - Sorting of lists and trees is built in
 - Easy to include search and auto-complete elements
 - ComboBox = cross between text input and select drop down

Why Create a Desktop Application? - Performance

● System Performance

- Offload work to client
 - All UI is client side - No need to process or output any HTML
 - Data can be pre-screened to reduce invalid submission (which require a second request)
 - Raw data can be passed back and manipulated on client
 - Batched requests can be sent during off-peak hours
- Offload data to client
 - Direct access to user's computer for data storage
 - State maintained client side – no need for session
- Offload users!
 - Users don't have to be constantly connected to server

Why Create a Desktop Application? - Scalability

- **Reduced server load == higher critical mass**
 - Server load of 10 desktop users < Server load of 10 website users
 - Can add more users before you need to add more servers

- **Surprisingly small additional code footprint**
 - Data/Display separation allows reuse of code and data
 - Cross-platform == reduced code base
 - Stock icons built in – no need to create/distribute images

Why Create a Desktop Application? - Flexibility

● Customizable

- Features can be added/removed/relocated to suit user's preference
- User's theme determines default settings
 - Icons and or text for toolbars and menus.
 - Color scheme and font sizes
- Allows multiple presentations of same data to suit user's needs

● Time management

- User can work when and where they want – no need to be online.
- Multi-tasking – user can work while background tasks process

● Internationalization

- Stock text displays based on users locale
- All text UTF-8

Why Create a Desktop Application? - Branding

● Customization

- Application can be made to fit corporate color scheme
- Shape of widgets can be shaped
 - All buttons can be corporate logo

● Desktop Presence

- Icon on users desktop, start menu, task bar, etc...
- More options == more chances to create loyal customer

Providing a GUI for a Web Service

- **Goals**

- Simplify the user experience
- Streamline and direct task flow
- Empower the user with ability to work with data on their terms
- Make the user feel more comfortable working with data
- Increase value add for users
- Branding

Connecting to the Web Service

- Connecting is easy – Use PHP 5's SoapClient class!

```
try {  
    $this->soapClient = new SoapClient(self::WSDL);  
} catch (SoapFault $sf) {  
    $this->displayError($sf->getMessage());  
}  
  
...  
$details = $this->soapClient->get_workorder_details(...);
```

Building the GUI

- **Step 1 – Provide a way to collect information from the user for the different API calls**
- **Step 2 – Provide a way to display the return information from the different API calls**
- **Step 3 – Make it look cool**

Collecting Input

- Create input forms just like HTML

- Textarea
- Text input
- Check boxes
- Radio buttons
- Select drop downs
- File chooser
- Submit/Reset buttons

The screenshot shows a web browser window titled "OnForce Client Desktop v1.0.0". The browser has a menu bar with "File", "View", and "Help". Below the menu bar is a toolbar with icons for back, home, search, and a plus sign. A progress bar is visible below the toolbar. The main content area displays a form with the following fields:

- *Category: Consumer Electronics (dropdown menu)
- *Short Description: This is a demo. (text input)
- *Problem Description: Fix something. I don't care what, just find something and fix it. Even if you have to break something. (textarea)
- *Spend Limit: \$ 150 (text input)
- Flat Fee: Flat Fee: (checkbox)
- Tech Supply Parts: Tech Supply Parts: (checkbox)
- OS: -- Select -- (dropdown menu)
- Manufacturer: manuf003 (text input)
- Model: model003 (text input)
- Serial Number: (text input)

At the bottom of the form, there are "Previous" and "Next" buttons. A note at the bottom right states: "* denotes required field".

Collecting Input – A Little Differently

- **Some input elements have no HTML equivalent**
 - SpinButton – Text input with up/down arrows for selecting number from range
 - Sliders – Dragable bars for selecting from a numeric range
 - ComboBox – Cross between select drop down and text input
 - Used to provide suggestions and allow free form input
 - Input can be restricted to suggested values
 - Color chooser – Dialog for visually selecting a color value
 - Calendar – Widget for selecting dates

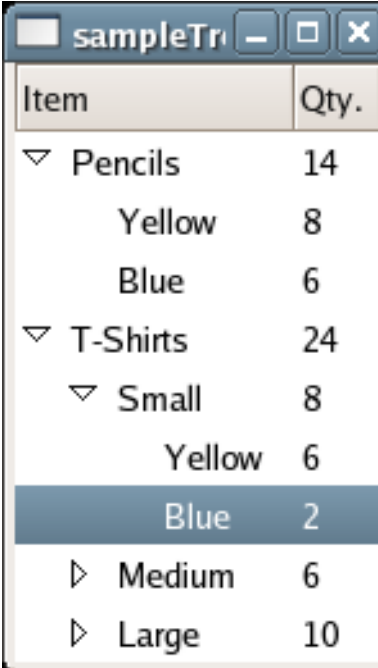
Displaying Output

- **Everything is flexible!**

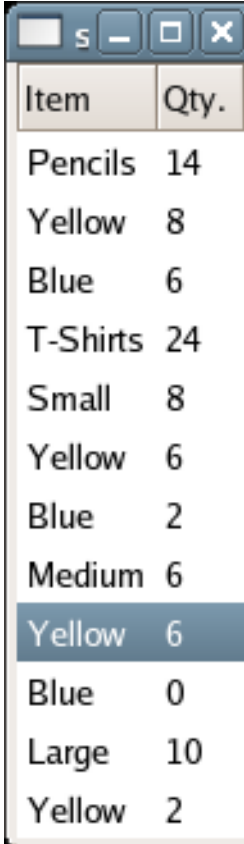
- Even labels (the simplest output method) can be customized
 - Left-to-right or right-to-left
 - at an angle
 - in different size, colors, weights or fonts
 - with different backgrounds
- Multi-line text, trees and lists can be formatted without changing underlying data
 - Different parts of application can display same data with different formatting
 - Visible area can be scrollable to allow large data to fit in a small space

Displaying Complex Output

- Tree view provides collapsible, searchable, sortable display for trees, lists and tabular data
- Data management and display logic separated into different classes
 - GtkTreeView, GtkTreeViewColumn, GtkCellRenderer, GtkSelection used for display
 - GtkTreeStore, GtkListStore, GtkTreeIter, GtkTreePath (actually just an array) used for data management



Item	Qty.
▼ Pencils	14
Yellow	8
Blue	6
▼ T-Shirts	24
▼ Small	8
Yellow	6
Blue	2
▶ Medium	6
▶ Large	10



Item	Qty.
Pencils	14
Yellow	8
Blue	6
T-Shirts	24
Small	8
Yellow	6
Blue	2
Medium	6
Yellow	6
Blue	0
Large	10
Yellow	2

Interacting with the User

- **HTML pages are request driven**

- User makes request, data is returned, repeat
- Only user can initiate action
- Examples: GET - click a link, POST - submit a form

- **PHP-GTK apps are event driven**

- Event occurs, action is taken, action may trigger other events
- User or application can trigger events
- Examples: click a button, change text in a text box, select a new element from a list, destroy a widget, close a window, etc

Interacting with the User – Signals & Events

- **When an event occurs, a signal is emitted**
 - Signal identifies a specific event on a specific widget
 - Signals can be emitted on user actions or system actions
 - User clicks button
 - Data arrives on watched I/O stream
 - Main loop check signal queue and handles requests for each signal emitted
- **Applications respond to signals via signal handlers**
 - Signal handler identifies action to be taken when specific signal is fired
 - Example: When main application window is destroyed, terminate main loop
 - Action taken by creating callback functions/methods
 - Example: `$window->connect_simple('destroy', array('Gtk', 'main_quit'));`

Customizing the Look & Feel – Makin' it Pretty

- **Display properties are similar to HTML**

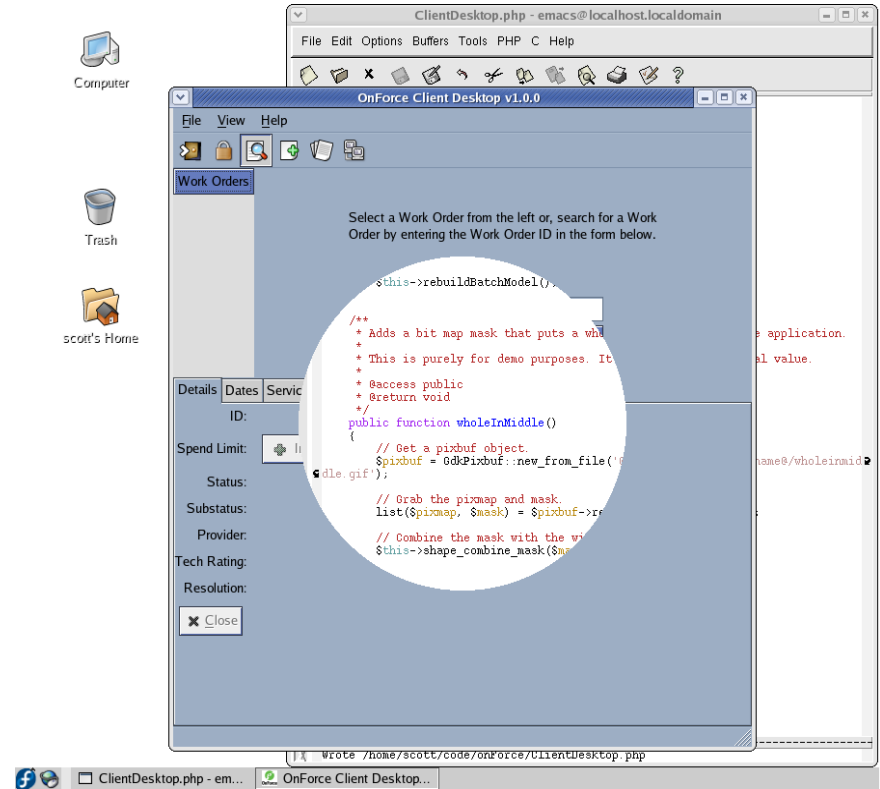
- Each widget has a GtkStyle object that defines colors, fonts, and sizes
- Appearance can be altered at run-time by altering GtkStyle object
- Appearance can be set at start up by parsing RC file
 - RC file is similar to CSS file
 - Defines styles and applies them to widgets based on rules
 - Rules can apply to specific widgets, widget classes, or widgets nested inside a specific container heirarchy

- **Stock Icons**

- Widely recognized icons make it easy for user to identify features
- Not a requirement
- Free to use any images you want, including animations

Customizing the Look & Feel – Changing Shapes

- **Widgets are not required to be rectangular**
 - All widgets (including main application window) can be re-shaped
 - Apply bitmask to widget, only non-transparent pixels will be shown
 - Create pony-shaped buttons
 - Get rid of normal rectangular window and make your app in the shape of your logo



More Information

- **PHP-GTK Website**
 - Documentation
 - Mailing Lists
 - IRC Channel
- **Gnope.org**
- **Crisscott.com**
- **PHP|Architect**
 - July & August issues
 - A/R/T
- **PHP-GTK 2 Cookbook**
 - <http://kksou.com/php-gtk2/>

