

Winbinding Your Code

Win32 GUI Development with
PHP and Winbinder

Aaron Wormus

Getting the Slides

- The most recent version of this slide and the accompanying source code from:

<http://www.wormus.com/slides/winbinder>

Questions?

Just shout!

Introductions

- Aaron Wormus
- Freelance Consultant
- Works with SMEs in implementing CMS/CRM systems and integrating them into their existing workflow.
- My clients use Windows

Who are you?

- PHP Programmers?
- Do you use Windows?
- Have you used other GUI toolkits (TK/GTK/wxWidgets)?
- Familiar with Event Based Programming?
- Full Fledged Win32 Programmers?
- Any GUI or Web designers here?
- Ever used AJAX?

This talk is a ...

- Technology Preview / Overview
- Code Tutorial
- Case Study

Table of Contents

- History of Winbinder
- Why Winbinder?
- Code Basics
- Windows and Controls
- Winbinder Framework
- Distributing your Code
- Design Considerations
- Q&A

What is Winbinder?

- Winbinder is a Framework
- Building a Windows Application requires more than just a GUI Toolkit
- Winbinder solves very specific problems
- While Winbinder is stable code, the project is still young
- Winbinder interfaces will probably change
- You have been warned

History: To Fill a Gap

- Winbinder was conceived as a Rapid Development environment for C programmers to prototype Applications.
- Initial Requirements:
 - Simple (does not force you to do OO coding)
 - C-like
 - Interpreted
 - Lightweight
 - Native Windows “Look and Feel”

It does a lot more than that!

- What using PHP adds
 - LOTS of PHP coders and potential developers
 - Simple access to Bundled PHP extensions
 - Easy access to PECL and thousands of third-party PHP libraries.
 - High performance (for an interpreted language)
 - Can easily create OO wrapper for procedural code
 - PHP allows a (relatively) small footprint and is easily installed

Why Winbinder?

- An affordance is a property of an object, or a feature of the immediate environment, that indicates how to interface with that object or feature.
- Have you ever used an application which used non-standard controls?
- Most users do not appreciate innovative user interfaces.
- As a technophile this isn't always easy to understand

Why Winbinder?

- Client side vs. Server side
 - Low latency
 - Software can interact with other client side applications
 - Direct interface with the filesystem and Windows API
 - There are fewer Security issues (javascript limits etc.) which limit web based applications
 - Have you ever hit backspace at the wrong time when filling out a complex web form?

Other Options: PHP-GTK

Pros

- Stable and documented
- Lots of Widgets available
- Cross platform
- Good object oriented design
- Easy to create user Interfaces with GLADE

Cons

- Stable version is PHP4 and GTK1
- Widgets do not have a Windows look/feel
- Verbose code
- Does not supply Window specific functions

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

Other Options: PHP-TK

Pros

- TK is a stable toolkit
- Syntax familiar to TK programmers
- Object oriented design

Cons

- Not available on Windows
- Requires TK to be installed
- Widgets do not have a Windows look/feel
- Not PHP-Like code

Project URL: <http://php-tk.sourceforge.net/>

Other Options: wxPHP

Pros

- WxWidgets look great
- Cross platform

Cons

- Under early Development
- Requires external libraries to be installed
- No working release

Project URL: <http://wxphp.sourceforge.net/>

Other Options: Ajax

Pros

- Cross platform
- Central Server Based Code
- Cool

Cons

- Usability Issues
- Web isn't the most stable platform for many programs
- Javascript Limitations
- No access to Windows API

It's Apples and Oranges!

Error Handling

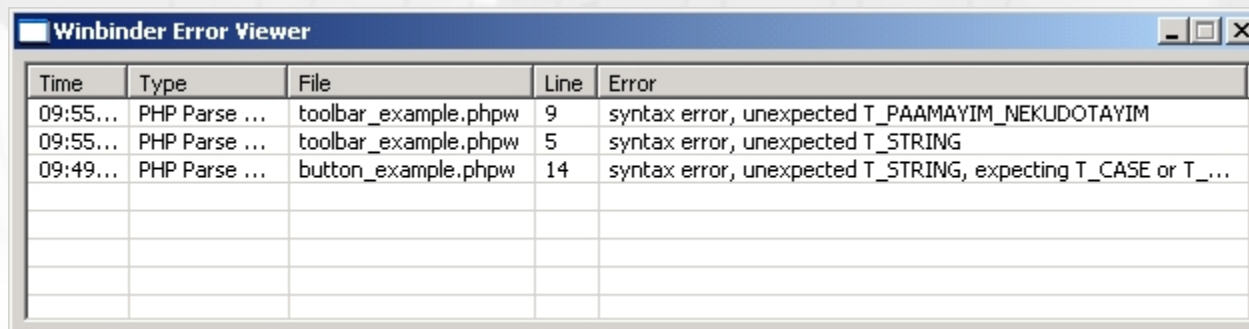
- Unlike the CLI SAPI php-win.exe does not echo error messages to STDOUT
- If a parse error occurs you will get no errors
- This is frustrating
- But quickly fixed with a few php.ini changes

```
display_errors = On  
log_errors = On  
error_log = php_errors.txt  
error_reporting = E_ALL & ~E_NOTICE
```

- Do this before starting

My Winbinder Error Viewer

- To ease development I created a log viewer

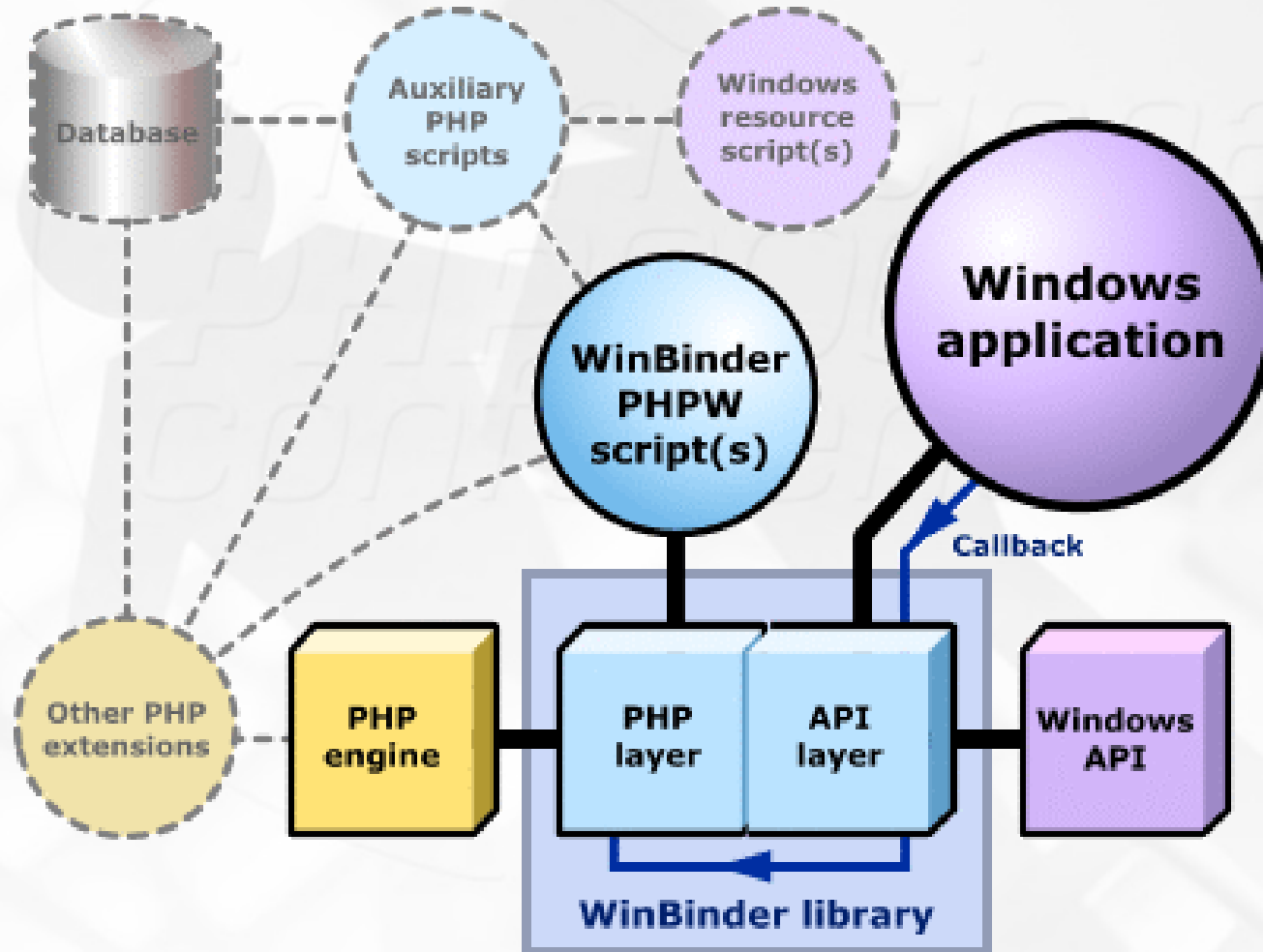


The screenshot shows a window titled "Winbinder Error Viewer" with a table containing error logs. The table has five columns: Time, Type, File, Line, and Error. The data rows are as follows:

Time	Type	File	Line	Error
09:55...	PHP Parse ...	toolbar_example.phpw	9	syntax error, unexpected T_PAAMAYIM_NEKUDOTAYIM
09:55...	PHP Parse ...	toolbar_example.phpw	5	syntax error, unexpected T_STRING
09:49...	PHP Parse ...	button_example.phpw	14	syntax error, unexpected T_STRING, expecting T_CASE or T_...

- Before distributing an application be sure to have a good error handling scheme in place

Quick Overview



PHP Winbinder Files

- `winbinder.php`
 - File to be included in all winbinder scripts
- `wb_resouces.inc.php`
 - Adds functions or parsing RC files
- `wb_generic.inc.php`
 - Ini parsing, and other convenience functions
- `wb_windows.inc.php`
 - Heavy lifting, creates controls, set/get values, creates standard dialog boxes

*international
PHP2005
conference*

Let's see some code!

Hello, World!

```
include "inc/winbinder.php";
define(BUTTON, 101);

$mainwin = wb_create_window(NULL, AppWindow, "Button Example", 220, 90);
wb_create_control($mainwin, PushButton, "Push Me", 70, 20, 80, 22, BUTTON);
wb_set_handler($mainwin, "process_main");
wb_main_loop();

function process_main($window, $event_id) {
    switch($event_id) {
        case BUTTON:
            wb_message_box($window, "You Pushed Me!");
            break;

        case IDCLOSE:
            wb_destroy_window($window);
            break;
    }
}
```

OO Interface

```
$w["main"] = new WbAppWindow("Clock", 170, 80);  
$w["currTime"] = $w["main"]->addControl('WbLabel', 5, 5, 150, 20, "time");  
$w["gauge"] = $w["main"]->addControl('WbGauge', 5, 25, 150, 20);
```

```
$timer = new WbTimer($w["main"]);  
$timer->action = "hit_timer";  
$timer->interval = 1000;  
$timer->start();
```

```
Wb::mainLoop();
```

```
function hit_timer(){  
    global $w;  
    $w["currTime"]->text = "Time: " . date("H:i:s");  
    $w["main"]->text = "[" . date("H:i:s") . "];"  
    $w["gauge"]->value = (date("s")*100)/59;  
}
```

Creating Controls

- Controls are created with the `wb_create_control()` function

```
wb_create_control($parent, // Parent Class  
$ctrl, // Control Name  
$param, // Parameter, Window Caption or Label  
$xpos, // X position of the control  
$ypos, // Y position of the control  
$width, // Control Width  
$height, // Control Height  
$event, // Event this Control Triggers  
$style, // Style parameter  
$param2, // Additional Parameters  
$tab // Tab position );
```

- These parameters vary depending on which control you are creating

Creating Controls

- `wb_create_control` is one of the PHP functions provided in the winbinder library.
- The controls which do not follow this pattern are:
 - Accel
 - Toolbar
 - Menu
 - Hyperlink
- When in the doubt, use the source!

PushButton

- Creating a regular button

```
define(BUTTON, 101);  
wb_create_control($parent,  
                PushButton,  
                "Push Me",  
                70, 20, 80, 22,  
                BUTTON);
```



- In this case we are using the third parameter to define the label of the button

Accel

- Accel control creates keyboard shortcuts

```
$parameters = array(array(IDCLOSE, "Alt+X"));  
wb_create_control($parent,  
                 Accel,  
                 $parameters);
```

- IDCLOSE is the name of the event which the accelerator triggers

Menu

- Creates a Menu for your window

```
$parameters = array(&Exit,  
                    array(IDCLOSE, "E&xit\tAlt+X", "", "", "Alt+X");  
wb_create_control($parent,  
                  Menu,  
                  $parameters);
```

- The fourth and fifth parameters are for "Hint" and "Icon"

Toolbar

- Creates a toolbar with image buttons

```
$parameters = array(array(ID_OPEN,  
                          NULL,  
                          "Open a file",  
                          1),  
                    null, // Toolbar separator  
                    array(IDCLOSE,  
                          NULL,  
                          "Exit this application",  
                          14));  
  
wb_create_control($parent,  
                ToolBar,  
                $parameters,  
                0, 0, 16, 15, 0, 0, // width/height refer to the icons  
                "toolbar.bmp");
```

ListView 1/2

- Creates a ListView control

```
define("ID_ITEMLIST", 101);
```

```
$list = wb_create_control($mainwin, ListView, "", 5, 5, 500, 200,  
    ID_ITEMLIST, WBC_VISIBLE | WBC_ENABLED | WBC_SORT |  
    WBC_LINES);
```

```
wb_set_text($list, array(  
    array("Username",      100),  
    array("Fullname",      180),  
    array("Display Name", 140),  
    array("Status",       60),  
));
```

```
// ...
```


Creating Windows

- Winbinder provides simple ways to create different styles of Windows.
- Windows are created with the function `wb_create_windows` which takes similar parameters to `wb_create_control`.

```
$mainwin = wb_create_window(NULL,  
                             ResizableWindow,  
                             "RSS Reader",  
                             WBC_CENTER,  
                             WBC_CENTER,  
                             410, 400,  
                             WBC_NOTIFY, WBC_RESIZE);
```

Triggers and Positions

- WBC_NOTIFY creates triggers for events which the window triggers.
- WBC_CENTER centers the Window in your screen. This can also be done manually:

```
$workarea = explode(" ", wb_get_system_info("workarea"));  
$xwidth = $workarea[2];  
$ywidth = $workarea[3];
```

- You get the idea. The clock example shows this in detail.

Window Styles (1/2)

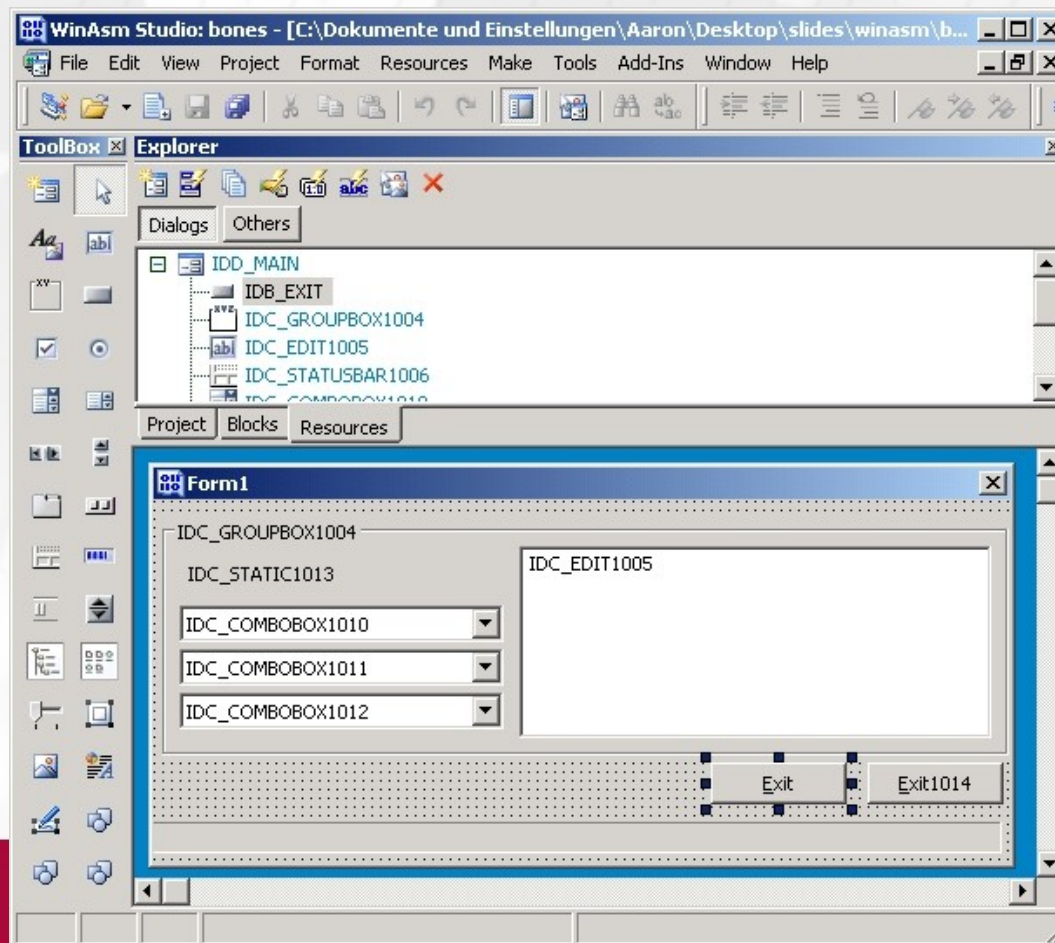
- Application Window Types
 - `AppWindow` - A fixed-size application window.
 - `ResizableWindow` - A normal application window with a resizable border.
 - `NakedWindow` - A fixed-size application window with no border and no title bar.
 - `PopupWindow` - A fixed-size application window that cannot be minimized.
- Additional Styles can be applied through the `Style` parameter.

Window Styles (2/2)

- Dialog Boxes
 - ModalDialog - A modal dialog box.
 - ModelessDialog - A modeless dialog box.
 - ToolDialog - Modeless dialog with a caption.
- Standard System Dialogs
 - wb_sys_dlg_color - Select Color dialog box
 - wb_sys_dlg_open - Open dialog box
 - wb_sys_dlg_path - Select Path dialog box
 - wb_sys_dlg_save - Save As dialog box

Building your GUI

- Winbinder supports the import of RC files from WinASM.

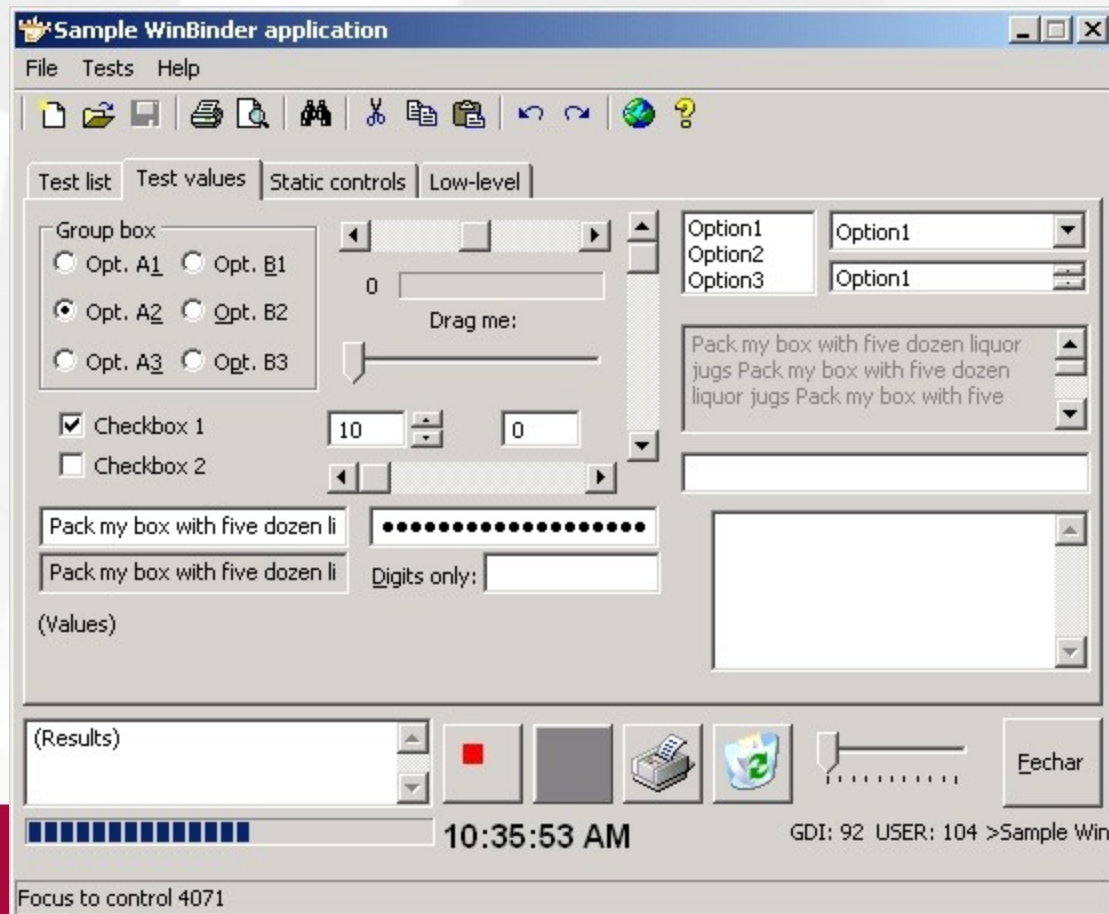


Building your GUI

- On-the-fly RC parsing offers an easy way to separate your code from display
- What you will see in Winbinder code
 - `eval(parse_rc(file_get_contents("calc.rc"), '$mainwin', null, 'PopupWindow'));`
- This will compile the calc.rc into PHP code and then run it.
- A better solution would be to use `file_put_contents`, and then include the file.

Building your GUI

- On-the-fly parsing allows you to create and maintain complex user interfaces easily.



The Framework – Databases

- Winbinder has built in database abstraction layer
- Four Goals
 - Isolate Application from Database type
 - Isolate Application from unique table names
 - No SQL knowledge needed
 - Usage of SQL is possible
- Currently SQLite and MySQL are supported

Opening a database

```
include "db_common.inc.php";

// Define your database type ... defaults to SQLite
define("WB_DATABASE", "SQLite");

// Define a table prefix for your Application
define("APPPREFIX", "myDB_");

define("PATH_DATA", "C:\winbinder\myApp\");
$database = "sample";

$dbhnd = db_open_database($database, PATH_DATA);

...

$result = db_close_database();
```

Using the Database

- Add records

- `$values = array("hut", " in the summer", 1);`
`$res = db_create_record("mytable", null, $fv);`

- Removing records

- `$idarray = array(1,3,5);`
`$result = db_delete_records("mytable",$idarray);`

- `db_query()` for SQL calls

- `$result = db_query("SELECT 'Tel./Fax' FROM mytable");`
`$tel_fx_nrs = db_fetch_array($result);`

- Functions for database maintenance are also included

Getting Data out of the Database

```
// Open database
db_open_database("sample.db");

// Read data from user whose identifier is 53
wb_set_text(wb_get_control($mainwin, ID_USER), db_get_data("user", 53));

// Fill up a list box with all user names
wb_set_text(wb_get_control($mainwin, ID_USERS),
            db_get_data("user", null, "name"));

// Display all users that are over 18 in a ListView
$itemlist = wb_get_control($window, ID_USERLIST);
$fields = array("id","name","phone","address");
$item_data = db_get_data("user", null, $fields, "age > 18");

wb_set_text($itemlist, $fields);
wb_create_items($itemlist, $item_data, true);
```

Other DB Options

- The Winbinder database only offers the most basic functionality
- If you want more advanced DB access and abstraction, look at PEAR::MDB2
- For creating larger projects a data persistence layer, like Propel or DB_Dataobjects from PEAR.

Registry Editing

- Reading and writing to the windows registry is supported
- ```
$wallpaper = wb_get_registry_key("HKCU",
 "Control Panel\\Desktop", "Wallpaper");
```
- Writing to the registry is possible, but dangerous
- ```
wb_set_registry_key("HKCU", "Software\\WinBinder", "String value", "1234");  
wb_set_registry_key("HKCU", "Software\\WinBinder", "Integer value", 1234);
```

Other Windows Functions

- `wb_exec()` spawns off a new process
 - Very useful, since PHP doesn't include this functionality
 - Can spawn a winbinder script, a regular windows application, or the program associated with a file.
- `wb_find_file()` searches and returns the path to a file
- `wb_play_sound()` plays a system sound

Packaging your Application

- Option 1: Create a self executing archive
 - IZArc and 7-Zip create self executing archives
 - Usually unpacks to a temp folder, but doesn't clean up after itself. Quickly wastes space.
- Option 2: Compile your PHP Script
 - PHC is a project that uses Bcompiler to create self contained executables.
 - Is not know to work with PHP5
- Option 3: Create an Installation Package
 - Inno Setup is free and creates nice packages.

Communicating with Windows

- Winbinder includes functions to communicate with any Windows library.
- Talking to the Kernel is cool.
- So is changing variable values by directly editing the memory space.
- Both of these things have the potential to crash a lot of things.
- You have been warned.

COM makes life easier

- COM is the best way to communicate with other Windows applications.

```
$com = new COM("SKYPEAPI.Access")  
    or die("can not create SKYPEAPILib.Access object");
```

```
$com->Connect();  
$data = $com->GetFriendList();
```

Thoughts on Program Design

- Procedural Event Based programming can quickly become unmanageable.
- When designing anything more complex than a simple script, look to design patterns for help.
- MVC was made for this.
- The free book "Dissecting a C# Application" is a good read and covers application design in depth.

Thoughts on Interface Design

- You can get away with things in web based design that are unacceptable in Client side programs.
- Designing simple User Interfaces is hard
- Use Usability guides, not your imagination

The Future

- Graphical Dialog Editor
- New OO Extension Library
- PHP Compiler and simple way to bundle your applications
- Advanced data integration (think Datagrid)

The project needs your Support!

Questions?

*international
PHP2005
conference*

References

- Winbinder: <http://winbinder.sf.net>
- PHC: <http://www.phpcompiler.org/>
- Izarc: <http://www.izarc.org/>
- 7-Zip: <http://www.7-zip.org/>
- Inno Setup: <http://www.jrsoftware.org/isinfo.php>
- WinASM: <http://www.winasm.net>
- PEAR: <http://pear.php.net>