

Embedding PHP

\$embed->empower->enrich();

Sara Golemon
University of California – Berkeley

pollita@php.net

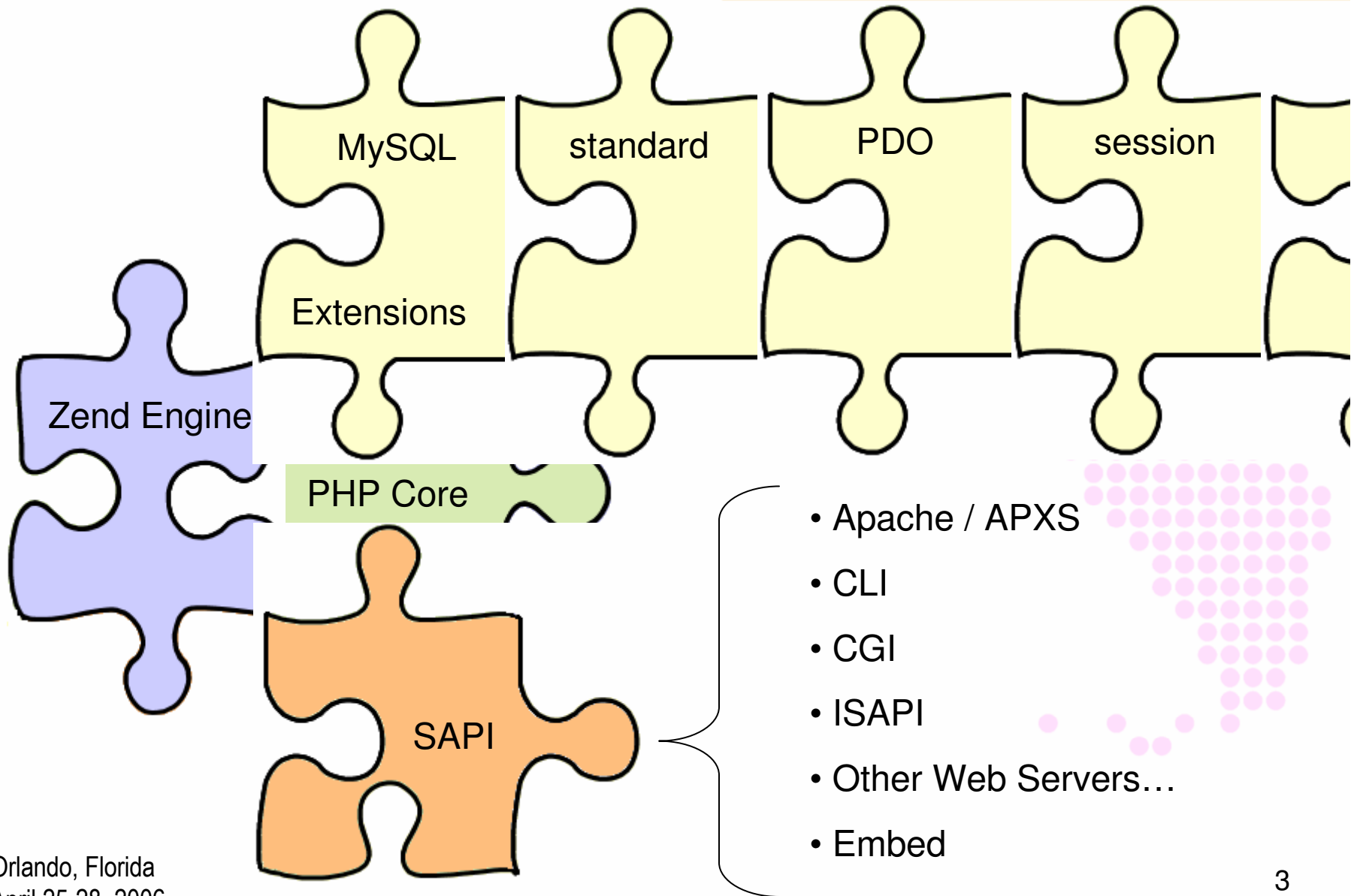


Introduction

- About This Talk
 - Embedding the PHP interpreter into 3rd party applications.
- About You
 - You have an application which could benefit from scriptibility.
 - You have some familiarity with the PHP API.
 - You like asking questions and aren't shy about it.
- About Me
 - PHP Internals & PECL Developer
 - It's my first time speaking, BE NICE!

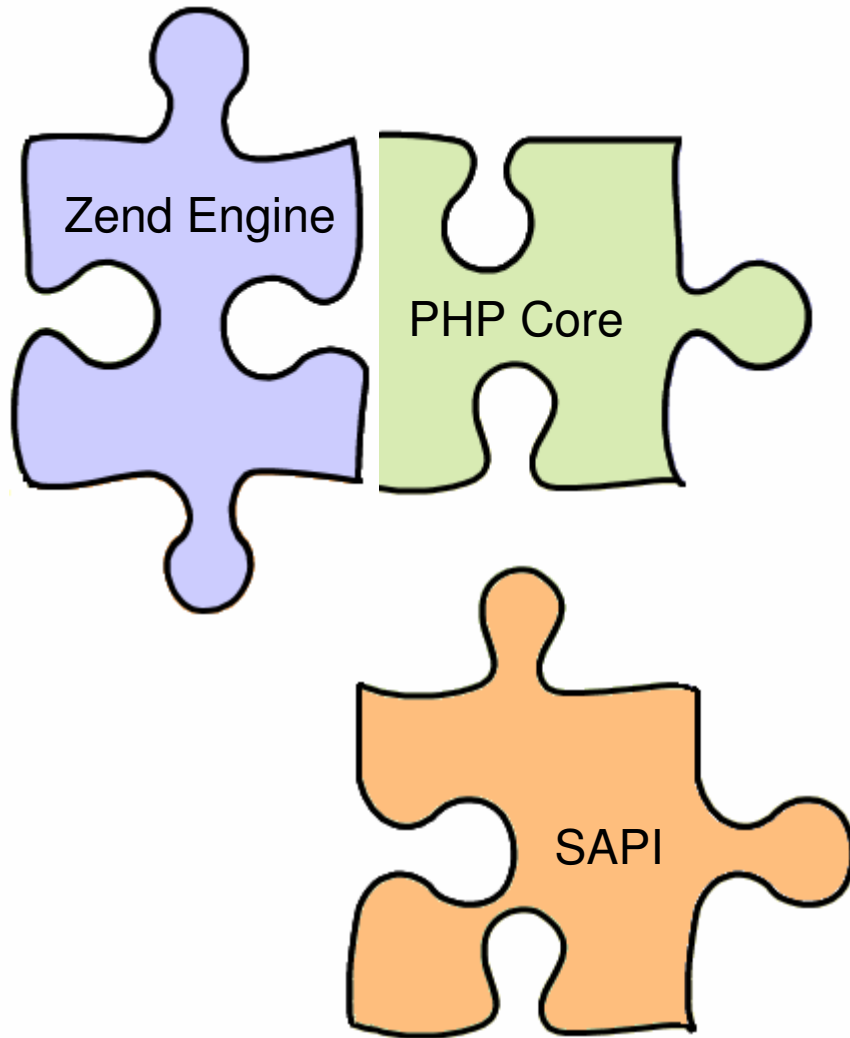


PHP Architecture














How this division looks in Windows



Name	Size
 php5ts.dll	4,309 KB

Name	Size
 php-win.exe	29 KB
 php-cgi.exe	53 KB
 php.exe	29 KB
 php5nsapi.dll	29 KB
 php5isapi.dll	29 KB
 php5embed.lib	612 KB
 php5apache_hooks.dll	53 KB
 php5apache.dll	37 KB
 php5apache2.dll	37 KB



Taking the role of the SAPI

A. Make system calls to PHP executables

- Slow: Every syscall requires PHP to be loaded, initialized, activated, deactivated, shutdown, and unloaded.
- Nonpersistent: Each call to PHP is a new request with a fresh symbol table and no state information.
- Noninteractive: No access to symbol tables, limited error handling, no ability to expose host application to PHP.

B. Write your own SAPI implementation

- Most powerful option for integrating an application with PHP.
- Several easy mistakes to make.
- More work than necessary for most applications.

C. Link against the Embed SAPI.

- Requires minimal programming to “get started”.
- Can be built on gradually to work towards a full-featured SAPI.
- Hides the messy ZTS bits from applications which don’t care about threading.



Installing SAPI/embed

- On Windows
 - Already built (php5embed.lib) but needs source code (for headers) as well.
- On Linux/BSD/*nix
 - *./configure --other-options *
--enable-embed=shared
or static
 - *make all install*
 - */usr/local/lib/libphp5.so*
 - */usr/local/include/php/**



Simplest Usage

sample.c

```
#include <php_embed.h>

int main (int argc, char *argv[])
{
    PHP_EMBED_START_BLOCK(argc, argv)
    zend_eval_string("echo 'Hello World';",
        NULL, "Embedded Code" TSRMLS_CC);
    PHP_EMBED_END_BLOCK()
    return 0;
}
```

```
CFLAGS=-O2 -Wall \
    -I/usr/local/include/php \
    -I/usr/local/include/php/Zend \
    -I/usr/local/include/php/main \
    -I/usr/local/include/php/TSRM \
    -I/usr/local/include/php/sapi/embed
LDLFLAGS=-L/usr/local/lib -lphp5
CC=gcc
```

```
sample: sample.c
    $(CC) sample.c -o sample \
        $(CFLAGS) $(LDLFLAGS)
```

- php_embed.h includes all other standard PHP headers.
- zend_eval_string() works similar to userspace eval() statement.
- “Embedded Code” label is used as the filename in error messages



Collecting return values

```
#include <php_embed.h>
#include <stdio.h>

int main (int argc, char *argv[])
{
    PHP_EMBED_START_BLOCK(argc, argv)
    zval retval;

    zend_eval_string("return 1 + 1;", &retval,
        "Code returning a value" TSRMLS_CC);
    convert_to_string(&retval);
    printf("The script code returned: %s\n", Z_STRVAL(retval));
    zval_dtor(&retval);
    PHP_EMBED_END_BLOCK()
}
```



Using PHP's output mechanism

```
#include <php_embed.h>

int main (int argc, char *argv[])
{
    PHP_EMBED_START_BLOCK(argc, argv)
    zval retval;

    zend_eval_string("return 1 + 1;", &retval,
        "Code returning a value" TSRMLS_CC);
    convert_to_string(&retval);
    php_printf("The script code returned: %s\n", Z_STRVAL(retval));
    zval_dtor(&retval);
    PHP_EMBED_END_BLOCK()
}
```



Setting Variables

```
#include <php_embed.h>

Int main (int argc, char *argv[])
{
    PHP_EMBED_START_BLOCK(argc, argv)
    zval *tmpvar;

    /* Populate tmpvar with a string */
    MAKE_STD_ZVAL(tmpvar);
    ZVAL_STRING(tmpvar, "My Scriptable Hello World", 1);

    /* Set tmpvar to a variable in the global scope */
    ZEND_SET_SYMBOL(&EG(symbol_table), "application", tmpvar);

    /* Execute script code */
    zend_eval_string("echo 'The name of the application is: ' . $application;",
        NULL, "Hardcoded script" TSRMLS_CC);
    PHP_EMBED_END_BLOCK()

    return 0;
}
```



Executing Scripts (The Short Way)

```
#include <php_embed.h>

int main(int argc, char *argv[])
{
    PHP_EMBED_START_BLOCK(argc, argv)
    zend_eval_string("include 'test.php';",
        NULL, "Test script includer" TSRMLS_CC);
    PHP_EMBED_END_BLOCK()

    return 0;
}
```





Executing Scripts (The Long Way)

```
#include <php_embed.h>

int main(int argc, char *argv[])
{
    zend_file_handle script;

    script.type = ZEND_HANDLE_FP;
    script.filename = "test.php";
    script.opened_path = NULL;
    script.free_filename = 0;
    script.handle.fp = fopen(script.filename, "rb");

    PHP_EMBED_START_BLOCK(argc,argv)
        zend_execute_scripts(ZEND_REQUIRE TSRMLS_CC, NULL, 1, &script);
    PHP_EMBED_END_BLOCK()
    return 0;
}
```





Executing Scripts

```
void myapp_execute_script(char *filename TSRMLS_DC)
{
    zend_file_handle script;

    script.type = ZEND_HANDLE_FP;
    script.filename = filename;
    script.opened_path = NULL;
    script.free_filename = 0;
    script.handle.fp = fopen(script.filename, "rb");

    zend_execute_scripts(ZEND_REQUIRE TSRMLS_CC,
                        NULL, 1, &script);
}
```





Altering INI Entries

```
#include <php_embed.h>

int main(int argc, char *argv[])
{
    PHP_EMBED_START_BLOCK(argc, argv)
    zend_alter_ini_entry(
        "display_errors", sizeof("display_errors"),
        "0", sizeof("0") - 1,
        PHP_INI_SYSTEM, PHP_INI_STAGE_RUNTIME);
    zend_alter_ini_entry(
        "log_errors", sizeof("log_errors"),
        "1", sizeof("1") - 1,
        PHP_INI_SYSTEM, PHP_INI_STAGE_RUNTIME);
    myapp_execute_script("test.php" TSRMLS_CC);
    PHP_EMBED_END_BLOCK()
    return 0;
}
```



Capturing Output

- By default, echo/print go to stdout
- Error messages go to stderr
(when log_errors is enabled)
- Both of these targets are usually not where you want output to go.
- These can be redirected by overriding sapi/Embed's default handlers:
 - php_embed_module.ub_write
 - php_embed_module.log_message
 - php_embed_module.sapi_error



Capturing echo/print

```
#include <php_embed.h>

int myapp_php_ub_write(const char *str, unsigned int str_length TSRMLS_DC)
{
    myapp_write_to_window(str);

    return str_length;
}

int main(int argc, char *argv[])
{
    php_embed_module.ub_write = myapp_php_ub_write;
    PHP_EMBED_START_BLOCK(argc, argv)
        zend_eval_string("echo 'Hello World';", NULL,
            "Captured output example" TSRMLS_CC);
    PHP_EMBED_END_BLOCK()

    return 0;
}
```





Buffering Output

```
#include <php_embed.h>

int myapp_php_ub_write(const char *str, unsigned int str_length TSRMLS_DC) { }
void myapp_php_flush(void *server_context) { }

int main(int argc, char *argv[])
{
    php_embed_module.ub_write = myapp_php_ub_write;
    php_embed_module.flush = myapp_php_flush;
    ...
}
```

- ub_write method may store output in its own internal buffers as desired
- Useful for line-buffering output, or for accumulating all output before display/use.
- flush() method called at the of script processing to indicate that output should be performed.



Capturing Errors

```
#include <php_embed.h>

void myapp_php_log_message(char *message) { }
void myapp_php_sapi_error(int type, const char *fmt, ...) { }

int main(int argc, char *argv[])
{
    php_embed_module.log_message = myapp_php_log_message;
    php_embed_module.sapi_error = myapp_php_sapi_error;
    ...
}
```

- Same concept as normal output handler, but with different prototypes.
- Nearly all errors will wind up at *log_message**
- Only a few specialized errors (file upload problems) will be routed to *sapi_error*.

* when log_errors is enabled and no error_log file is named



Capturing Unformatted Errors

```
#include <php_embed.h>

void *myapp_php_error_cb(int type, const char *error_filename, const uint error_lineno,
                        const char *format, va_list args)
{
    char buffer[2048];
    int len;

    if (type != E_ERROR && type != E_USER_ERROR && type != E_CORE_ERROR &&
        type != E_PARSE && type != E_COMPILE_ERROR) {
        return;
    }
    len = snprintf(buffer, 2048, "Error on line %d: ", error_lineno);
    vsnprintf(buffer + len, (2048 - len), format, args);
    myapp_show_error(buffer);

    zend_bailout(); /* Like a userspace "exit;" - IMPORTANT - Fatal Errors, must BE fatal */
}

int main(int argc, char *argv[])
{
    PHP_EMBED_START_BLOCK(argc, argv)
    zend_error_cb = myapp_php_error_cb;
    zend_eval_string("This->is a parse error;", NULL, "Error Capturing example" TSRMLS_CC);
    PHP_EMBED_END_BLOCK()
    return 0;
}
```

Abstracting PHP in your application

This won't build:

```
#include <php_embed.h>

void myapp_init(argc, argv)
{
    /* Startup and Initialization related to application
    PHP_EMBED_START_BLOCK(argc, argv)
}

void myapp_done()
{
    PHP_EMBED_END_BLOCK()
    /* Shutdown and cleanup related to application
}

int main(int argc, char *argv[])
{
    myapp_init(argc, argv);
    myapp_done();

    return 0;
}
```

```
{
    TSRMLS_D;
    php_embed_init(argc, argv TSRMLS_CC);
    EG(bailout_set) = 0;
    zend_try {

```

```
} zend_catch {
} zend_end_try();
php_embed_shutdown(TSRMLS_C);
}
```



Abstracting PHP in your application

This will:

```
#include <php_embed.h>

#ifdef ZTS
void ***tsrm_ls;
#endif

void myapp_init(argc, argv)
{
    /* Startup and Initialization
       related to application */
    php_embed_init(argc, argv TSRMLS_CC);
    EG(bailout_set) = 0;
}

void myapp_done()
{
    php_embed_shutdown(TSRMLS_C);
    /* Shutdown and cleanup
       related to application */
}
```

- Keeps startup and shutdown code in one spot and “PHP clutter” to a minimum.
- Allows the possibility of providing runtime-optional PHP scripting support via dlopen() or similar mechanism.

```
int main(int argc, char *argv[])
{
    myapp_init(argc, argv);
    myapp_done();

    return 0;
}
```



Dealing with critical errors

- Errors can happen anywhere in a PHP script execution:
 - Out of memory
 - Time limit exceeded
 - `trigger_error("message", E_USER_ERROR);`
 - Scripting error
 - Syntax/Parse Error
 - Call to undefined function/class/method

How Zend Jumps Out of the Way

```
#define zend_first_try EG(bailout_set) = 0; zend_try
```

```
zend_try {
```

```
} zend_catch {
```

```
} zend_end_try();
```

```
{  
    jmp_buf orig_bailout;  
    zend_bool orig_bailout_set=EG(bailout_set);  
  
    EG(bailout_set)=1;  
    memcpy(&orig_bailout, &EG(bailout),  
          sizeof(jmp_buf));  
    if (setjmp(EG(bailout)) == 0) {  
        /* Normal Code goes here */  
    } else {  
        /* zend_bailout(); sends processing here */  
    }  
    memcpy(&EG(bailout), &orig_bailout,  
          sizeof(jmp_buf));  
    EG(bailout_set) = orig_bailout_set;  
}
```



Bailout Conditions

```
php_printf("Hello World");  
zend_bailout();  
php_printf("I am never reached!");
```

```
php_printf("Hello World");  
php_error_docref(NULL TSRMLS_CC, E_ERROR, "foo");  
php_printf("I am never reached!");
```

```
php_printf("Hello World");  
zend_eval_string("$parse?ERROR;broken...", NULL,  
    "Deliberately broken example" TSRMLS_CC);  
php_printf("I am never reached!");
```

```
php_printf("Hello World");  
ptr = emalloc(INI_GET("memory_limit") * 2);  
php_printf("I am never reached!");
```



Catching Errors Gracefully

```
int myapp_execute_script(char *filename TSRMLS_DC)
{
    zend_file_handle script;
    int ret = SUCCESS;

    script.type = ZEND_HANDLE_FP;
    script.filename = filename;
    script.opened_path = NULL;
    script.free_filename = 0;
    script.handle.fp = fopen(script.filename, "rb");

    zend_try {
        zend_execute_scripts(ZEND_REQUIRE TSRMLS_CC, NULL, 1, &script);
    } zend_catch {
        ret = FAILURE;
    } zend_end_try();

    return ret;
}
```



Extending and Embedding at Once

- Generic scripting allows the user to do plenty with the system environment, but doesn't give ready access to your application.
- By extending at the same time, your application's internal functionality can be exposed giving more power to the end user.



Simple Example

```
PHP_FUNCTION(myapp_popup)
{
    char *message;
    int message_len;

    if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "s",
                             &message, &message_len) == FAILURE) {
        return;
    }
    RETURN_BOOL(myapp_display_popup(message, message_len));
}
```

```
zend_function_entry myapp_php_functions[] = {
    PHP_FE(myapp_popup, NULL)
    { NULL, NULL, NULL }
};
```

```
zend_module_entry myapp_php_module = {
    STANDARD_MODULE_HEADER,
    "myapp",
    myapp_php_functions,
    NULL, NULL, NULL, NULL,
    "1.0", /* version */
    STANDARD_MODULE_PROPERTIES
};
```

```
void myapp_init(void)
{
    static char *argv[] = { "myapp", NULL };

    php_embed_module.ub_write = myapp_php_ub_write;
    php_embed_module.flush = myapp_php_flush;
    php_embed_init(1, argv TSRMLS_CC);

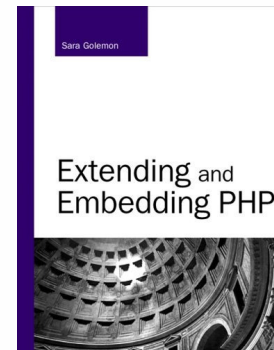
    zend_startup_module(&myapp_php_module);

    zend_error_cb = myapp_php_error_cb;
    zend_alter_ini_entry(
        "max_execution_time", sizeof("max_execution_time"),
        "0", sizeof("0") - 1,
        PHP_INI_SYSTEM, PHP_INI_STAGE_RUNTIME);
    EG(bailout_set) = 0;
}
```

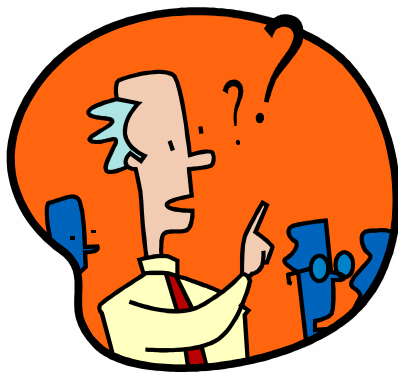
Where to go from here

- Existing projects using sapi/Embed
 - PHP-IRSSI `cvs -d:pserver:cvsread@cvs.php.net:/repository co embed/php-irssi`
- Implement your own SAPI
 - Use embed as a starting point for building your own SAPI.
 - Start by using sapi/embed
 - Gradually replace SAPI struct methods
 - Eventually drop embed and link into PHP with your own implementation
- Internet Resources
 - PHP Mailing lists (pecl-dev@lists.php.net)
 - Websites (<http://www.zend.com/php/internals>)
- “How PHP Ticks” (Tomorrow at 2:15, Room 3)
- Books
 - Extending and Embedding PHP

(by me, yes this is a shameless plug)



Questions?



Orlando, Florida
April 25-28, 2006



sapi/embed

```
sapi_module_struct php_embed_module = {
    "embed", /* name */
    "PHP Embedded Library", /* pretty_name */
    php_embed_startup, /* startup */
    php_module_shutdown_wrapper, /* shutdown */
    NULL, /* activate */
    php_embed_deactivate, /* deactivate */
    php_embed_ub_write, /* ub_write */
    php_embed_flush, /* flush */
    NULL, /* get_stat */
    NULL, /* getenv */
    php_error, /* sapi_error */
    NULL, /* header_handler */
    NULL, /* send_headers */
    php_embed_send_header, /* send_header */
    NULL, /* read_post */
    php_embed_read_cookies, /* read_cookies */
    php_embed_register_variables, /* register_server_variables */
    php_embed_log_message, /* log_message */
    NULL, /* get_request_time */
    STANDARD_SAPI_MODULE_PROPERTIES
};
```





php_embed_ini()

```
int php_embed_init(int argc, char **argv TSRMLS_DC)
{
#ifdef ZTS
    zend_compiler_globals *compiler_globals;
    zend_executor_globals *executor_globals;
    php_core_globals *core_globals;
    sapi_globals_struct *sapi_globals;
    void ***tsrm_ls;
#endif
    signal(SIGPIPE, SIG_IGN);

#ifdef ZTS
    tsrm_startup(1, 1, 0, NULL);

    compiler_globals = ts_resource(compiler_globals_id);
    executor_globals = ts_resource(executor_globals_id);
    core_globals = ts_resource(core_globals_id);
    sapi_globals = ts_resource(sapi_globals_id);
    tsrm_ls = ts_resource(0);
    *ptsrm_ls = tsrm_ls;
#endif

    sapi_startup(&php_embed_module);
    if (php_embed_module.startup(&php_embed_module)
        ==FAILURE) {
        return FAILURE;
    }
}
```

```
if (argv) {
    php_embed_module.executable_location = argv[0];
}

SG(options) |= SAPI_OPTION_NO_CHDIR;
zend_alter_ini_entry("register_argc_argv", 19, "1", 1,
    PHP_INI_SYSTEM, PHP_INI_STAGE_ACTIVATE);
zend_alter_ini_entry("html_errors", 12, "0", 1,
    PHP_INI_SYSTEM, PHP_INI_STAGE_ACTIVATE);
zend_alter_ini_entry("implicit_flush", 15, "1", 1,
    PHP_INI_SYSTEM, PHP_INI_STAGE_ACTIVATE);
zend_alter_ini_entry("max_execution_time", 19, "0", 1,
    PHP_INI_SYSTEM, PHP_INI_STAGE_ACTIVATE);

SG(request_info).argc=argc;
SG(request_info).argv=argv;

if (php_request_startup(TSRMLS_C)==FAILURE) {
    php_module_shutdown(TSRMLS_C);
    return FAILURE;
}

SG(headers_sent) = 1;
SG(request_info).no_headers = 1;
php_register_variable("PHP_SELF", "-", NULL TSRMLS_CC);

return SUCCESS;
}
```



php_embed_shutdown()

```
void php_embed_shutdown(TSRMLS_D)
{
    php_request_shutdown((void *) 0);
    php_module_shutdown(TSRMLS_C);
    sapi_shutdown();
#ifdef ZTS
    tsmr_shutdown();
#endif
}
```

php_embed_init() was simplified for brevity, refer to CVS for the full version:

http://cvs.php.net/php-src/sapi/embed/php_embed.c