

Welcome!

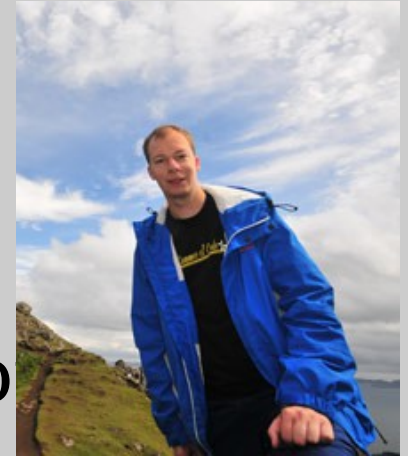
# PHP Extensions - What and Why

ZendCon - Santa Clara, US - Oct 17th, 2011

Derick Rethans - [derick@derickrethans.nl](mailto:derick@derickrethans.nl) - twitter:  
@derickr  
<http://joind.in/3776>

## Derick Rethans

- Dutchman living in London
- PHP development
- Freelancer doing PHP internals development  
ie.: writing extensions for a living
- Anderskor's Camouflage
- Author of Xdebug
- Author of the `mcrypt`, `input_filter`, `dbus`, `translit` and `date/time` extensions
- Contributor to the Apache Zeta Components Incubator project (formerly eZ Components)



# Extension on a slide

```
helloworld.c (~/dev/php/hello-world) - VIM - Terminal
PHP_ARG_ENABLE(helloworld, whether to enable helloworld support,
  --enable-helloworld      Enable helloworld support)

if test "$PHP_helloworld" != "no"; then
  PHP_NEW_EXTENSION(helloworld, helloworld.c, $ext_shared)
fi
config.m4
2,1 All
#ifndef PHP_HELLOWORLD_H
#define PHP_HELLOWORLD_H

#include "php.h"

extern zend_module_entry helloworld_module_entry;
#define phpext_helloworld_ptr &helloworld_module_entry

#ifdef PHP_WIN32
#define PHP_HELLOWORLD_API __declspec(dllexport)
#else
#define PHP_HELLOWORLD_API
#endif

#ifdef ZTS
#include "TSRM.h"
#endif

PHP_FUNCTION(helloworld);

#endif
~
~
~
~
~
~
php_helloworld.h 17,1 All
#endif HAVE_CONFIG_H
#include "config.h"
#endif

#include "php.h"
#include "php_helloworld.h"
#include "ext/standard/info.h"

#ifdef COMPILE_DL_HELLOWORLD
ZEND_GET_MODULE(helloworld)
#endif

PHP_FUNCTION(helloworld)
{
    php_printf("Hello World!\n");
}

zend_function_entry helloworld_functions[] = {
    PHP_FE(helloworld, NULL)
    {NULL, NULL, NULL}
};

zend_module_entry helloworld_module_entry = {
    STANDARD_MODULE_HEADER,
    "helloworld",
    helloworld_functions,
    NULL, NULL, NULL, NULL,
    "0.0.1",
    STANDARD_MODULE_PROPERTIES
};
helloworld.c 30,2 All
```

# Config.m4: configuration

- Hooks into PHP's configure system
- Both for shared and static extensions
- Also checks for libraries, functions, and system/OS dependent features

## Minimal config.m4:

```
PHP_ARG_ENABLE(helloworld, whether to enable helloworld support,  
[ --enable-helloworld          Enable helloworld support])  
  
if test "$PHP_helloworld" != "no"; then  
    PHP_NEW_EXTENSION(helloworld, helloworld.c, $ext_shared)  
fi
```

# Config.m4: ext/date

```
sininclude(ext/date/lib/timelib.m4)
sininclude(lib/timelib.m4)

PHP_DATE_CFLAGS="-I@ext_builddir@/lib"
timelib_sources="lib/astro.c lib/dow.c lib/parse_date.c lib/parse_tz.c
                lib/timelib.c lib/tm2unixtime.c lib/unixtime2tm.c lib/parse_iso_intervals.c lib/interval.c"

PHP_NEW_EXTENSION(date, php_date.c $timelib_sources, no,, $PHP_DATE_CFLAGS)

PHP_ADD_BUILD_DIR([$ext_builddir/lib], 1)
PHP_ADD_INCLUDE([$ext_builddir/lib])
PHP_ADD_INCLUDE([$ext_srcdir/lib])

PHP_INSTALL_HEADERS([ext/date], [php_date.h lib/timelib.h lib/timelib_structs.h lib/timelib_config.h])

cat > $ext_builddir/lib/timelib_config.h <<EOF
#ifdef PHP_WIN32
# include "config.w32.h"
#else
# include <php_config.h>
#endif
EOF
```

# Config.m4: ext/fileinfo

```
PHP_ARG_ENABLE(fileinfo, for fileinfo support,
[ --disable-fileinfo      Disable fileinfo support], yes)

if test "$PHP_FILEINFO" != "no"; then

    libmagic_sources=" \
        libmagic/apprentice.c libmagic/apptype.c libmagic/ascmagic.c \
        libmagic/cdf.c libmagic/cdf_time.c libmagic/compress.c \
        libmagic/encoding.c libmagic/fsmagic.c libmagic/funcs.c \
        libmagic/is_tar.c libmagic/magic.c libmagic/print.c \
        libmagic/readcdf.c libmagic/readelf.c libmagic/softmagic.c"

    PHP_NEW_EXTENSION(fileinfo, fileinfo.c $libmagic_sources, $ext_shared,,-
I@ext_srcdir@/libmagic)
    PHP_ADD_BUILD_DIR($ext_builddir/libmagic)

    AC_CHECK_FUNCS([utimes strndup])

    PHP_ADD_MAKEFILE_FRAGMENT
fi
```

# Config.m4: ext/openssl

```
PHP_ARG_WITH(openssl, for OpenSSL support,
[ --with-openssl[=DIR]    Include OpenSSL support (requires OpenSSL >= 0.9.6)])

PHP_ARG_WITH(kerberos, for Kerberos support,
[ --with-kerberos[=DIR]    OPENSSSL: Include Kerberos support], no, no)

if test "$PHP_OPENSSL" != "no"; then
    PHP_NEW_EXTENSION(openssl, openssl.c xp_ssl.c, $ext_shared)
    PHP_SUBST(OPENSSSL_SHARED_LIBADD)

    if test "$PHP_KERBEROS" != "no"; then
        PHP_SETUP_KERBEROS(OPENSSSL_SHARED_LIBADD)
    fi

    AC_CHECK_LIB(ssl, DSA_get_default_method, AC_DEFINE(HAVE_DSA_DEFAULT_METHOD, 1, [OpenSSL 0.9.7 or later]))
    AC_CHECK_LIB(crypto, X509_free, AC_DEFINE(HAVE_DSA_DEFAULT_METHOD, 1, [OpenSSL 0.9.7 or later]))

    PHP_SETUP_OPENSSL(OPENSSSL_SHARED_LIBADD,
    [
        AC_DEFINE(HAVE_OPENSSL_EXT,1,[ ])
    ], [
        AC_MSG_ERROR([OpenSSL check failed. Please check config.log for more information.])
    ])
fi
```

# Config.m4: pecl/dbus

```
PHP_ARG_ENABLE(dbus, whether to enable dbus functions,
[ --enable-dbus[=DIR]          Enable dbus support], yes)

if test -z "$PHP_LIBXML_DIR"; then
    PHP_ARG_WITH(libxml-dir, libxml2 install dir,
[ --with-libxml-dir=DIR        DBUS: libxml2 install prefix], no, no)
fi

if test "$PHP_DBUS" != "no"; then
    if test "$PHP_LIBXML" = "no"; then
        AC_MSG_ERROR([DBus extension requires LIBXML extension, add --enable-libxml])
    fi

    AC_MSG_CHECKING(for pkg-config)
    if test ! -f "$PKG_CONFIG"; then
        PKG_CONFIG=`which pkg-config`
    fi
    if test -f "$PKG_CONFIG"; then
        AC_MSG_RESULT(found)

        AC_MSG_CHECKING(for dbus)
        if $PKG_CONFIG --exists dbus-1; then
            AC_MSG_RESULT(found)
            LDFLAGS="$LDFLAGS ` $PKG_CONFIG --libs dbus-1 ` "
            CFLAGS="$CFLAGS ` $PKG_CONFIG --cflags dbus-1 ` "
            PHP_SETUP_LIBXML(DBUS_SHARED_LIBADD, [
                AC_DEFINE(HAVE_DBUS, 1, [whether dbus exists in the system])
                PHP_NEW_EXTENSION(dbus, dbus.c introspect.c, $ext_shared)
                PHP_EVAL_LIBLINE($LDFLAGS,DBUS_SHARED_LIBADD)
                PHP_SUBST(DBUS_SHARED_LIBADD)
            ], [
                AC_MSG_ERROR([xml2-config not found. Please check your libxml2 installation.])
            ])
        else
            AC_MSG_RESULT(not found)
            AC_MSG_ERROR(The DBUS-1 package was not detected on your system)
        fi
    else
        AC_MSG_RESULT(not found)
        AC_MSG_ERROR(pkg-config is not found)
    fi
fi
```

# Config.m4: xdebug

```
PHP_ARG_ENABLE(xdebug, whether to enable eXtended debugging support,
[ --enable-xdebug          Enable Xdebug support])

if test "$PHP_XDEBUG" != "no"; then
    dn1 We need to set optimization to 0 because my GCC otherwise optimizes too
    dn1 much out.
    CFLAGS=`echo $CFLAGS | sed 's/O2/O0/'`

    AC_MSG_CHECKING([Check for supported PHP versions])
    PHP_XDEBUG_FOUND_VERSION=`${PHP_CONFIG} --version`
    PHP_XDEBUG_FOUND_VERNUM=`echo "${PHP_XDEBUG_FOUND_VERSION}" | $AWK 'BEGIN { FS = "."; } { printf "%d", ([\$]1 * 100 + [\$]2)
* 100 + [\$]3;}'`
    if test "$PHP_XDEBUG_FOUND_VERNUM" -lt "50400"; then
        AC_MSG_RESULT([supported ($PHP_XDEBUG_FOUND_VERSION)])
    else
        AC_MSG_ERROR([not supported. Need a PHP version < 5.4.0 (found $PHP_XDEBUG_FOUND_VERSION)])
    fi

    AC_DEFINE(HAVE_XDEBUG,1,[ ])

    dn1 Check for new current_execute_data field in zend_executor_globals
    old_CPPFLAGS=$CPPFLAGS
    CPPFLAGS="$INCLUDES $CPPFLAGS"

    AC_TRY_COMPILE([
#include <zend_compile.h>
#include <zend_globals.h>
], [static struct _zend_executor_globals zeg; zend_execute_data *zed = zeg.current_execute_data],
    [AC_DEFINE(HAVE_EXECUTE_DATA_PTR, 1, [ ])]
    )
    AC_CHECK_FUNCS(gettimeofday)

    PHP_CHECK_LIBRARY(m, cos, [ PHP_ADD_LIBRARY(m,, XDEBUG_SHARED_LIBADD) ])

    CPPFLAGS=$old_CPPFLAGS

    PHP_NEW_EXTENSION(xdebug, xdebug.c xdebug_code_coverage.c xdebug_com.c xdebug_compat.c xdebug_handler_dbgp.c
xdebug_handlers.c xdebug_llist.c xdebug_hash.c xdebug_private.c x
    PHP_SUBST(XDEBUG_SHARED_LIBADD)
    PHP_ADD_MAKEFILE_FRAGMENT
fi
```

# Config.w32

```
ARG_WITH("mcrypt", "mcrypt support", "no");

if (PHP_MCRYPT != "no") {

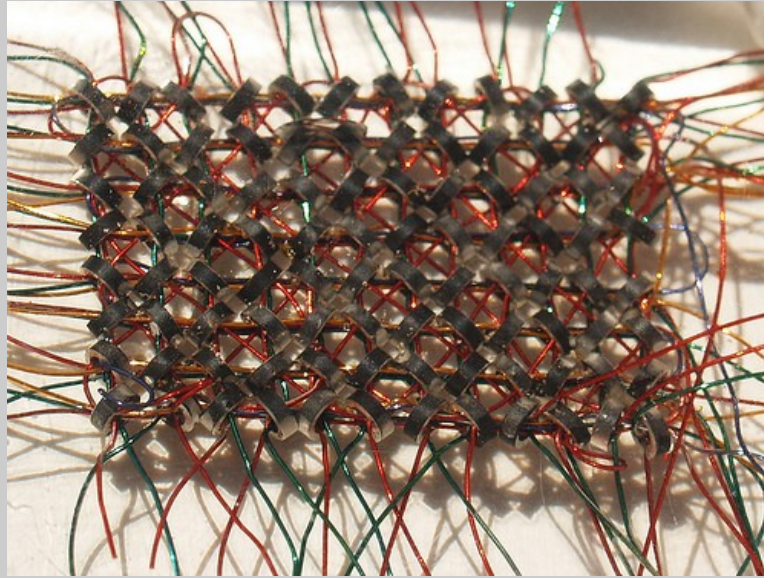
    if (CHECK_HEADER_ADD_INCLUDE('mcrypt.h', 'CFLAGS_MCRYPT') &&
        CHECK_LIB('libmcrypt_a.lib;libmcrypt.lib', 'mcrypt') &&
        CHECK_LIB('Advapi32.lib', 'mcrypt')
        ) {

        EXTENSION('mcrypt', 'mcrypt.c mcrypt_filter.c', false);
        AC_DEFINE('HAVE_LIBMCRYPT', 1);
        AC_DEFINE('HAVE_LIBMCRYPT24', 1);
    } else {
        WARNING("mcrypt not enabled; libraries and headers not found");
    }
}
```

# What?



- Extensions add functionality to PHP
- Extensions replace functionality in PHP
- PHP's extension mechanism is easy
- PHP's extension mechanism is (too) powerful



ext/standard

- Contains all the default PHP functions
- Can not be disabled

Other core extensions are:

- ext/date
- ext/ereg
- ext/pcre
- ext/reflection



Everything in ext/

- Are compiled into PHP
- Can most of the time be compiled as shared extensions as well ← distributions like to do that (too much)



Everything not in the core distribution

- Are often not compiled into PHP, but instead are compiled as shared objects
- Sometimes make their way into the core distribution, and sometimes they are moved away



## PECL: PHP Extension C Library

- <http://pecl.php.net>
- A whole load of extensions for various interesting (and odd things)
- Are installed with the PECL tool
- Used to be called "Siberia"
- Don't really have to be hosted in PHP's SVN repository, on the PECL website



# A quick overview...

## Internal Extensions

### Core Extensions

ext/standard  
ext/date  
ext/ereg  
ext/pcre  
ext/reflection  
ext/spl

### Bundled Extensions

ext/bcmath  
ext/bz2  
ext/calendar  
ext/com\_dotnet  
ext/ctype  
ext/curl  
ext/dba  
ext/dom  
ext/enchant  
ext/exif  
ext/fileinfo  
ext/filter  
ext/ftp  
...

## External Extensions

### PECL Extensions

pecl/activestscript  
pecl/adt  
pecl/amfext  
pecl/amqp  
pecl/apache\_accessor  
pecl/apc  
pecl/apd  
pecl/apm  
...  
pecl/yaz  
pecl/yp  
pecl/zeroconf  
pecl/zip  
pecl/zlib\_filter

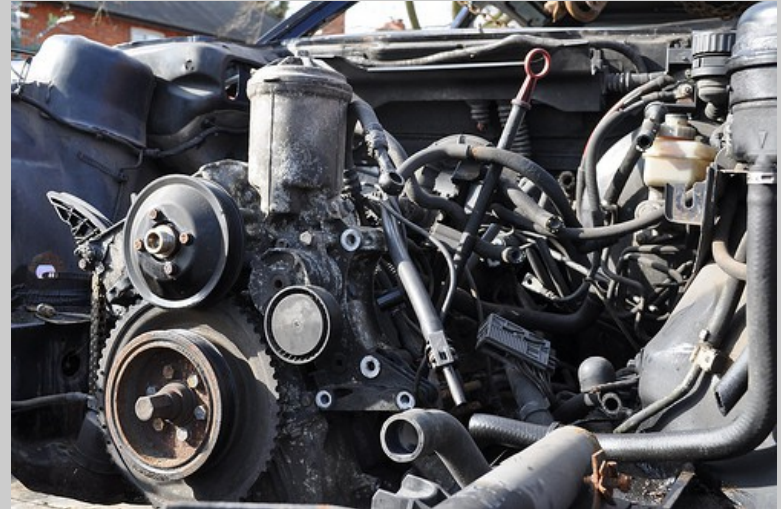
### Other Extensions

github.com/facebook/xhprof  
github.com/facebook/xhp  
github.com/fhoenig/Kellner  
github.com/fujimoto/php-fuse  
github.com/mkoppanen/php-zmq  
github.com/mgdm/php-pango  
...

# Traditional parts

- Module init/shutdown
- Request init/shutdown
- Functions
- Configuration (INI) settings

```
zend_module_entry xdebug_module_entry = {  
    STANDARD_MODULE_HEADER,  
    "xdebug",  
    xdebug_functions,  
    PHP_MINIT(xdebug),  
    PHP_MSHUTDOWN(xdebug),  
    PHP_RINIT(xdebug),  
    NULL, // PHP_RSHUTDOWN not in use  
    PHP_MINFO(xdebug),  
    XDEBUG_VERSION,  
#if (PHP_MAJOR_VERSION == 5 && PHP_MINOR_VERSION >= 2) || PHP_MAJOR_VERSION >= 6  
    NO_MODULE_GLOBALS,  
#endif  
    ZEND_MODULE_POST_ZEND_DEACTIVATE_N(xdebug),  
    STANDARD_MODULE_PROPERTIES_EX  
};
```



# Module initialisation and shutdown

```
PHP_MINIT_FUNCTION(xdebug)
{
    zend_extension dummy_ext;

    ZEND_INIT_MODULE_GLOBALS(xdebug, php_xdebug_init_globals, php_xdebug_shutdown_globals);
    REGISTER_INI_ENTRIES();

    ...

    /* Redirect compile and execute functions to our own */
    old_compile_file = zend_compile_file;
    zend_compile_file = xdebug_compile_file;
}

PHP_MSHUTDOWN_FUNCTION(xdebug)
{
    if (XG(profiler_aggregate)) {
        xdebug_profiler_output_aggr_data(NULL TSRMLS_CC);
    }

    /* Reset compile, execute and error callbacks */
    zend_compile_file = old_compile_file;
}

...
```

# .ini entries

```
PHP_INI_BEGIN()  
    STD_PHP_INI_ENTRY("xdebug.manual_url",          "http://www.php.net", PHP_INI_ALL,    OnUpdateString,  
        manual_url,          zend_xdebug_globals, xdebug_globals)  
  
    STD_PHP_INI_ENTRY("xdebug.max_nesting_level", "100",          PHP_INI_ALL,    OnUpdateLong,  
        max_nesting_level, zend_xdebug_globals, xdebug_globals)  
  
    STD_PHP_INI_BOOLEAN("xdebug.overload_var_dump", "1", PHP_INI_SYSTEM|PHP_INI_PERDIR, OnUpdateBool,  
        overload_var_dump, zend_xdebug_globals, xdebug_globals)  
  
    STD_PHP_INI_BOOLEAN("xdebug.show_exception_trace", "0",          PHP_INI_ALL,    OnUpdateBool,  
        show_ex_trace,      zend_xdebug_globals, xdebug_globals)
```

- name, default value
- where is it active
- what to do when it updates
- internal globals struct field
- globals struct type
- global struct name

# Request initialisation and shutdown

```
PHP_RINIT_FUNCTION(xdebug)
{
    zend_function *orig;
    char *idekey;
    zval **dummy;

    /* get xdebug ini entries from the environment also */
    XG(ide_key) = NULL;
    xdebug_env_config();
    idekey = zend_ini_string("xdebug.idekey", sizeof("xdebug.idekey"), 0);

    XG(no_exec)          = 0;
    XG(level)            = 0;
    XG(do_trace)         = 0;
    XG(coverage_enable) = 0;
    XG(do_code_coverage) = 0;
    XG(code_coverage)   = xdebug_hash_alloc(32, xdebug_coverage_file_dtor);
    XG(stack)            = xdebug_llist_alloc(xdebug_stack_element_dtor);

    ...
}

PHP_RSHUTDOWN_FUNCTION(date)
{
    if (DATEG(timezone)) {
        efree(DATEG(timezone));
    }
    DATEG(timezone) = NULL;
    if (DATEG(tzcache)) {
        zend_hash_destroy(DATEG(tzcache));
        FREE_HASHTABLE(DATEG(tzcache));
        DATEG(tzcache) = NULL;
    }

    ...
}
```

# Request post-deactivation

```
ZEND_MODULE_POST_ZEND_DEACTIVATE_D(xdebug)
{
    zend_function *orig;
    TSRMLS_FETCH();

    if (XG(remote_enabled)) {
        XG(context).handler->remote_deinit(&(XG(context)));
        xdebug_close_socket(XG(context).socket);
    }
    if (XG(context).program_name) {
        xdfree(XG(context).program_name);
    }
    ...
}
```

# Shutdown order

- 1. Call all possible shutdown functions registered with `register_shutdown_function()`
- 2. Call all possible `__destruct()` functions
- 3. Flush all output buffers
- 4. Send the set HTTP headers (note: This must be done AFTER `php_output_discard_all()` / `php_output_end_all()` !!)
- 5. Reset `max_execution_time` (no longer executing php code after response sent)
- 6. Call all extensions RSHUTDOWN functions
- 7. Destroy super-globals
- 7.5 free last error information
- 7. Shutdown scanner/executor/compiler and restore ini entries
- 8. Call all extensions post-RSHUTDOWN functions
- 9. SAPI related shutdown (free stuff)
- 10. Destroy stream hashes
- 11. Free Willy (here be crashes) -shuts down memory manager
- 12. Reset `max_execution_time`

## Function declaration (in php\_xdebug.h):

```
PHP_FUNCTION(xdebug_start_error_collection);
```

## Function definition (in xdebug.c):

```
PHP_FUNCTION(xdebug_start_error_collection)
{
    if (XG(do_collect_errors) == 1) {
        php_error(E_NOTICE, "Error collection was already started");
    }
    XG(do_collect_errors) = 1;
}
```

## Function entry array (in xdebug.c):

```
zend_function_entry xdebug_functions[] = {
    ...
    PHP_FE(xdebug_start_error_collection,      NULL)
    ...
    {NULL, NULL, NULL}
};
```

## Definition (in `php_xdebug.h`):

```
ZEND_BEGIN_MODULE_GLOBALS (xdebug)
    int          status;
    int          reason;

    ...
    zend_bool   do_scream;
ZEND_END_MODULE_GLOBALS (xdebug)
```

## Using:

```
if (XG(remote_enabled) && XG(context).handler->register_eval_id && fse->function.type == XFUNC_EVAL) {
    eval_id = XG(context).handler->register_eval_id(&(XG(context)), fse);
}
```

## In other `.c` files:

```
ZEND_EXTERN_MODULE_GLOBALS (xdebug)
```

# Modern parts



- Classes and methods
- Argument description
- Dependencies on other extensions

# Class registration

## in MINIT:

```
PHP_MINIT_FUNCTION(quickhash)
{
    qh_register_class_intset(TSRMLS_C);
}
```

## In qh\_intset.h:

```
typedef struct _php_qh_intset_obj php_qh_intset_obj;
```

```
struct _php_qh_intset_obj {
    zend_object      std;
    qhi              *hash;
};
```

```
void qh_register_class_intset(TSRMLS_D);
PHPAPI zend_class_entry *php_qh_get_intset_ce(void);
```

## In qh\_intset.c:

```
void qh_register_class_intset(TSRMLS_D)
{
    zend_class_entry ce_intset;

    INIT_CLASS_ENTRY(ce_intset, "QuickHashIntSet", qh_funcs_intset);
    ce_intset.create_object = qh_object_new_intset;
    qh_ce_intset = zend_register_internal_class_ex(&ce_intset, NULL, NULL, TSRMLS_CC);
    memcpy(&qh_object_handlers_intset, zend_get_std_object_handlers(), sizeof(zend_object_handlers));
}
```

# Class registration (2)

## In `qh_intset.c`:

```
static inline zend_object_value qh_object_new_intset_ex(zend_class_entry *class_type, php_qh_intset_obj **ptr TSRMLS_DC)
{
    php_qh_intset_obj *intern;
    zend_object_value retval;
    zval *tmp;

    intern = emalloc(sizeof(php_qh_intset_obj));
    memset(intern, 0, sizeof(php_qh_intset_obj));
    if (ptr) {
        *ptr = intern;
    }

    zend_object_std_init(&intern->std, class_type TSRMLS_CC);
#ifdef PHP_MINOR_VERSION > 3
    object_properties_init(&intern->std, class_type);
#else
    zend_hash_copy(intern->std.properties, &class_type->default_properties,
        (copy_ctor_func_t) zval_add_ref, (void *) &tmp, sizeof(zval *));
#endif

    retval.handle = zend_objects_store_put(intern, (zend_objects_store_dtor_t) zend_objects_destroy_object,
        (zend_objects_free_object_storage_t) qh_object_free_storage_intset, NULL TSRMLS_CC);
    retval.handlers = &qh_object_handlers_intset;

    return retval;
}

static zend_object_value qh_object_new_intset(zend_class_entry *class_type TSRMLS_DC)
{
    return qh_object_new_intset_ex(class_type, NULL TSRMLS_CC);
}

static void qh_object_free_storage_intset(void *object TSRMLS_DC)
{
    php_qh_intset_obj *intern = (php_qh_intset_obj *) object;

    if (intern->hash) {
        qho *tmp_options = intern->hash->options;

        qhi_free(intern->hash);
        qho_free(tmp_options);
    }

    zend_object_std_dtor(&intern->std TSRMLS_CC);
    efree(object);
}
```

- TSRM: TSRMLS\_C, TSRMLS\_CC, TSRMLS\_D, TSRMLS\_DC, TSRMLS\_FETCH()
- PHP\_MINOR\_VERSION, PHP\_VERSION\_ID
- memory allocation: emalloc/efree/estrdup vs malloc/free/strdup
- `#if COMPILE_DL_XDEBUG`  
`ZEND_GET_MODULE(xdebug) #endif`

# Class methods

## In `qh_intset.h`:

```
PHP_METHOD(QuickHashIntSet, __construct);
PHP_METHOD(QuickHashIntSet, getSize);
PHP_METHOD(QuickHashIntSet, add);
PHP_METHOD(QuickHashIntSet, exists);
PHP_METHOD(QuickHashIntSet, delete);
...
```

## In `qh_intset.c`:

```
zend_function_entry qh_funcs_intset[] = {
    PHP_ME(QuickHashIntSet, __construct,    arginfo_qh_intset_construct,    ZEND_ACC_CTOR|ZEND_ACC_PUBLIC)
    PHP_ME(QuickHashIntSet, getSize,        arginfo_qh_intset_get_size,    ZEND_ACC_PUBLIC)
    PHP_ME(QuickHashIntSet, add,            arginfo_qh_intset_add,        ZEND_ACC_PUBLIC)
    PHP_ME(QuickHashIntSet, exists,        arginfo_qh_intset_exists,    ZEND_ACC_PUBLIC)
    PHP_ME(QuickHashIntSet, delete,        arginfo_qh_intset_exists,    ZEND_ACC_PUBLIC)
    ...
    {NULL, NULL, NULL}
};

PHP_METHOD(QuickHashIntSet, getSize)
{
    ...
}
```

# Function/Method duality

## In ext/date/php\_date.h:

```
PHP_FUNCTION(date_timezone_get);
```

## In ext/date/php\_date.c:

```
const zend_function_entry date_functions[] = {
    PHP_FE(date_timezone_get, arginfo_date_timezone_get)
    PHP_FE_END
};

const zend_function_entry date_funcs_date[] = {
    PHP_ME_MAPPING(createFromFormat, date_timezone_get, arginfo_date_timezone_get, ZEND_ACC_PUBLIC)
    PHP_FE_END
};

PHP_FUNCTION(date_timezone_get)
{
    zval          *object;
    php_date_obj  *dateobj;
    php_timezone_obj *tzobj;

    if (zend_parse_method_parameters(ZEND_NUM_ARGS() TSRMLS_CC, getThis(), "O", &object, date_ce_date) == FAILURE) {
        RETURN_FALSE;
    }
    dateobj = (php_date_obj *) zend_object_store_get_object(object TSRMLS_CC);
    ...
}
```

## Definition:

```
ZEND_BEGIN_ARG_INFO_EX(arginfo_array_multisort, ZEND_SEND_PREFER_REF, 0, 1)
    ZEND_ARG_INFO(ZEND_SEND_PREFER_REF, arr1) /* ARRAY_INFO(0, arg1, 0) */
    ZEND_ARG_INFO(ZEND_SEND_PREFER_REF, SORT_ASC_or_SORT_DESC)
    ZEND_ARG_INFO(ZEND_SEND_PREFER_REF, SORT_REGULAR_or_SORT_NUMERIC_or_SORT_STRING)
    ZEND_ARG_INFO(ZEND_SEND_PREFER_REF, arr2)
    ZEND_ARG_INFO(ZEND_SEND_PREFER_REF, SORT_ASC_or_SORT_DESC)
    ZEND_ARG_INFO(ZEND_SEND_PREFER_REF, SORT_REGULAR_or_SORT_NUMERIC_or_SORT_STRING)
ZEND_END_ARG_INFO()
```

## In the function list:

```
PHP_FE(array_multisort, arginfo_array_multisort)
```

## CLI output:

```
Function [ <internal:standard> function array_multisort ] {
  - Parameters [6] {
    Parameter #0 [ <required> &$arr1 ]
    Parameter #1 [ <optional> &${SORT_ASC_or_SORT_DESC} ]
    Parameter #2 [ <optional> &${SORT_REGULAR_or_SORT_NUMERIC_or_SORT_STRING} ]
    Parameter #3 [ <optional> &$arr2 ]
    Parameter #4 [ <optional> &${SORT_ASC_or_SORT_DESC} ]
    Parameter #5 [ <optional> &${SORT_REGULAR_or_SORT_NUMERIC_or_SORT_STRING} ]
  }
}
```

# Extension Dependencies

## In dbus/config.m4:

```
if test "$PHP_DBUS" != "no"; then
    if test "$PHP_LIBXML" = "no"; then
        AC_MSG_ERROR([DBUS extension requires LIBXML extension, add --enable-libxml])
    fi
fi
```

## In dbus/dbus.c:

```
static const zend_module_dep dbus_deps[] = {
    ZEND_MOD_REQUIRED("libxml")
    ZEND_MOD_END
};

zend_module_entry dbus_module_entry = {
    STANDARD_MODULE_HEADER_EX,
    NULL,
    dbus_deps,
    "dbus",          /* extension name */
    ...
};
```

There is also `ZEND_MOD_CONFLICTS` and `ZEND_MOD_OPTIONAL`; as well as `ZEND_MOD_REQUIRED_EX` and `ZEND_MOD_OPTIONAL_EX`

# Questions

?

# Variables

## The Zed-val

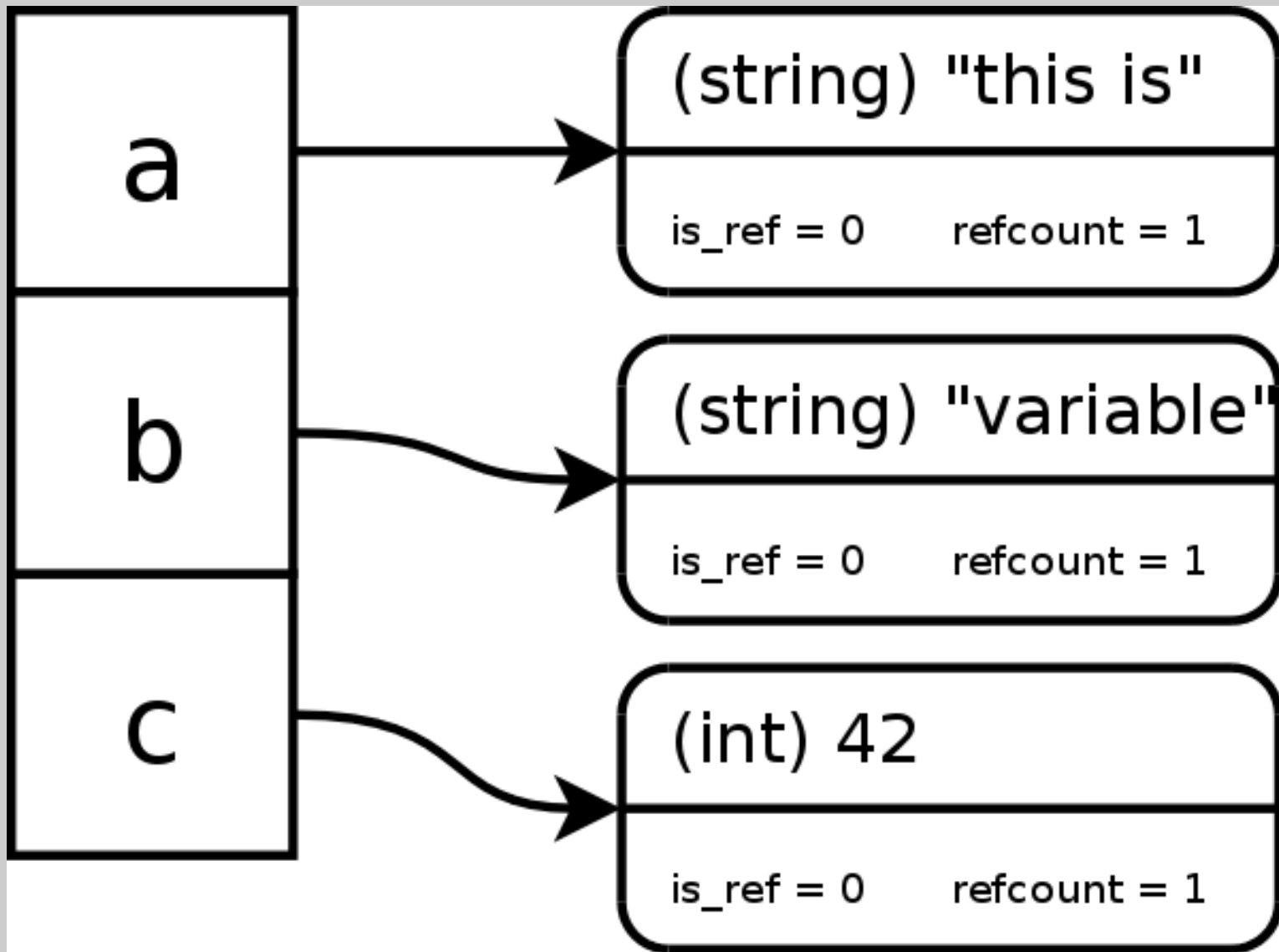
```
typedef union _zvalue_value {
    long lval;                /* long value */
    double dval;             /* double value */
    struct {
        char *val;
        int len;
    } str;
    HashTable *ht;           /* hash table value */
    zend_object_value obj;
} zvalue_value;
```

```
struct _zval_struct {
    /* Variable information */
    zvalue_value value;      /* value */
    zend_uint refcount__gc;
    zend_uchar type;        /* active type */
    zend_uchar is_ref__gc;
};
```

```
#define IS_NULL      0
#define IS_LONG     1
#define IS_DOUBLE   2
#define IS_BOOL     3
#define IS_ARRAY    4
#define IS_OBJECT   5
#define IS_STRING   6
#define IS_RESOURCE 7
#define IS_CONSTANT 8
#define IS_CONSTANT_ARRAY 9
```

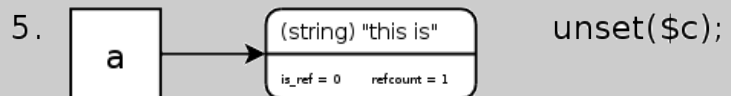
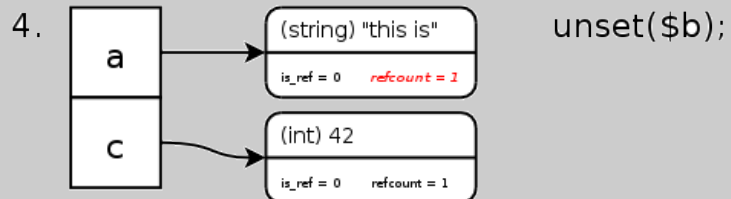
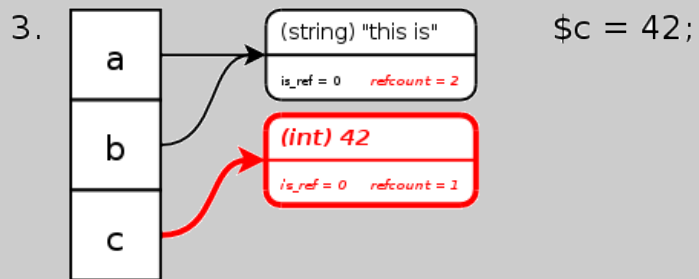
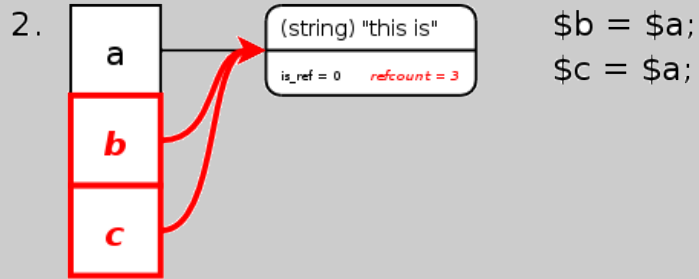
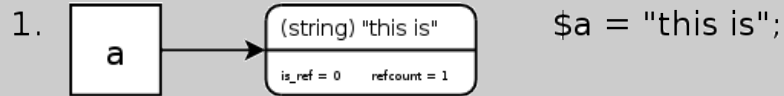
# Variables

## The Symbol Table



# Variables

## Introducing Refcounting

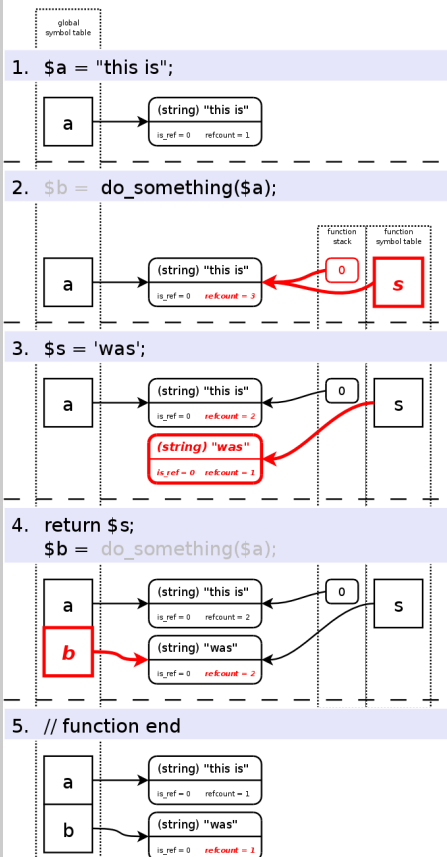


# Variables

## RefCounting and Passing Variables to User Defined Functions

```
<?php
function do_something($s)
{
    $s = 'was';
    return $s;
}

$a = 'this is';
$b = do_something($a);
?>
```

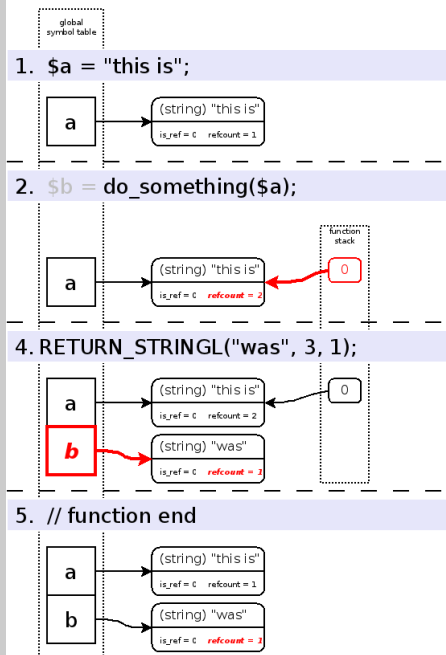


# Variables

## RefCounting and Passing Variables to User Defined Functions

```
PHP_FUNCTION(do_something)
{
    RETURN_STRINGL("was", 3, 1);
}

<?php
$a = 'this is';
$b = do_something($a);
?>
```

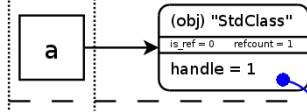


# Variables

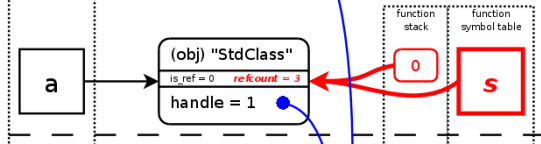
## Objects in PHP 5

```
<?php
function definition( $s )
{
    $s->prop = "baz";
}
$s = new StdClass;
$s->prop = "bar";
definition( $s );
?>
```

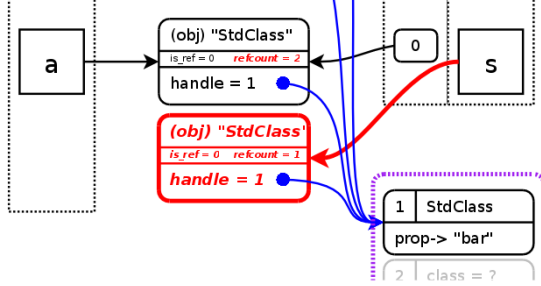
1. `$s = new StdClass; $s->prop = "bar";`



2. `definition( $s );`



3. `$s->prop = "baz";`



# Argument parsing

```
int zend_parse_parameters(int num_args TSRMLS_DC, char *type_spec, ...);
```

## Examples:

```
zend_bool clear = 0;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "|b", &clear) == FAILURE) {
    return;
}
```

```
char *prefix = NULL;
int prefix_len;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "|s", &prefix, &prefix_len) == FAILURE) {
```

```
char *fname = NULL;
int fname_len = 0;
long options = XG(trace_options);
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "|sl", &fname, &fname_len, &options) == FAILURE) {
    zval *array;
```

```
long sort_type = PHP_SORT_REGULAR;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "a|l", &array, &sort_type) == FAILURE) {
    zval *var_array, *prefix = NULL;
```

```
long extract_type = EXTR_OVERWRITE;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "a|lz/", &var_array, &extract_type, &prefix) == FAILURE) {
```

# Argument parsing

- a - array (zval\*)
- A - array or object (zval \*)
- b - boolean (zend\_bool)
- C - class (zend\_class\_entry\*)
- d - double (double)
- f - function or array containing php method call info (returned as zend\_fcall\_info and zend\_fcall\_info\_cache)
- h - array (returned as HashTable\*)
- H - array or HASH\_OF(object) (returned as HashTable\*)
- l - long (long)
- L - long, limits out-of-range numbers to LONG\_MAX/LONG\_MIN (long)
- o - object of any type (zval\*)
- O - object of specific type given by class entry (zval\*, zend\_class\_entry)
- p - valid path (string without null bytes in the middle) and its length (char\*, int)
- r - resource (zval\*)
- s - string (with possible null bytes) and its length (char\*, int)
- z - the actual zval (zval\*)
- Z - the actual zval (zval\*\*)
- \* - variable arguments list (0 or more)
- + - variable arguments list (1 or more)
- | - indicates that the remaining parameters are optional, they should be initialized to default values by the extension since they will not be touched by the parsing function if they are not passed to it.
- / - use SEPARATE\_ZVAL\_IF\_NOT\_REF() on the parameter it follows
- ! - the parameter it follows can be of specified type or NULL (applies to all specifiers except for 'b', 'l', and 'd'). If NULL is passed, the results pointer is set to NULL as well.

# Returning Values

## Scalar:

```
#define RETVAL_BOOL(b)                ZVAL_BOOL(return_value, b)
#define RETVAL_NULL()                ZVAL_NULL(return_value)
#define RETVAL_LONG(l)               ZVAL_LONG(return_value, l)
#define RETVAL_DOUBLE(d)            ZVAL_DOUBLE(return_value, d)
#define RETVAL_STRING(s, duplicate)  ZVAL_STRING(return_value, s, duplicate)
#define RETVAL_STRINGL(s, l, duplicate) ZVAL_STRINGL(return_value, s, l, duplicate)
#define RETVAL_EMPTY_STRING()        ZVAL_EMPTY_STRING(return_value)
#define RETVAL_ZVAL(zv, copy, dtor)  ZVAL_ZVAL(return_value, zv, copy, dtor)
#define RETVAL_FALSE                 ZVAL_BOOL(return_value, 0)
#define RETVAL_TRUE                   ZVAL_BOOL(return_value, 1)
```

+ RETURN\_\* versions of all of them.

```
PHP_METHOD(DbusObjectPath, getData)
{
    zval *object;
    php_dbus_object_path_obj *object_path_obj;

    if (FAILURE == zend_parse_method_parameters(ZEND_NUM_ARGS() TSRMLS_CC, getThis(),
        "O", &object, dbus_ce_dbus_object_path)) {
        RETURN_FALSE;
    }
    object_path_obj = (php_dbus_object_path_obj *) zend_object_store_get_object(object
    TSRMLS_CC);

    RETURN_STRING(object_path_obj->path, 1);
}
```

# Returning Values

## Arrays:

```
array_init(return_value);
```

```
PHP_METHOD(DbusSet, getData)
{
    int i;
    zval *object;
    php_dbus_set_obj *set_obj;

    if (FAILURE == zend_parse_method_parameters(ZEND_NUM_ARGS() TSRMLS_CC, getThis(),
        "O", &object, dbus_ce_dbus_set)) {
        RETURN_FALSE;
    }
    set_obj = (php_dbus_set_obj *) zend_object_store_get_object(object TSRMLS_CC);
    array_init(return_value);
    for (i = 0; i < set_obj->element_count; i++) {
        Z_ADDREF_P(set_obj->elements[i]);
        add_next_index_zval(return_value, set_obj->elements[i]);
    }
}
```

# Returning Values

## Objects:

```
PHP_FUNCTION(date_create)
{
    zval          *timezone_object = NULL;
    char          *time_str = NULL;
    int           time_str_len = 0;

    if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "|sO!", &time_str, &time_str_len, &timezone_object,
date_ce_timezone) == FAILURE) {
        RETURN_FALSE;
    }

    php_date_instantiate(date_ce_date, return_value TSRMLS_CC);
    if (!php_date_initialize(zend_object_store_get_object(return_value TSRMLS_CC), time_str, time_str_len, NULL,
timezone_object, 0 TSRMLS_CC)) {
        RETURN_FALSE;
    }
}
```

```
PHPAPI zval *php_date_instantiate(zend_class_entry *pce, zval *object TSRMLS_DC)
{
    Z_TYPE_P(object) = IS_OBJECT;
    object_init_ex(object, pce);
    Z_SET_REFCOUNT_P(object, 1);
    Z_UNSET_ISREF_P(object);
    return object;
}
```

# Excercise: helloworld

- Download from github: `git clone git://github.com/derickr/helloworld.git`
- `cd helloworld`
- `phpize && ./configure && make`
- `sudo make install`
- `php -dextension=helloworld.so --re helloworld`

## Output:

```
Extension [ <persistent> extension #61 helloworld version 0.0.1 ] {  
  
    - Functions {  
        Function [ <internal:helloworld> function helloworld ] {  
            }  
        }  
    }  
}
```

- The slide deck is at <http://derickrethans.nl/talks/phpexts-zendcon.pdf>
- Add a new function `helloearth()` that returns "Hello Earth!" (slide 23, 42).
- Add a new function `hello()` that accepts a string argument, and returns "Hello «argument»!" (slide 40/41).
- Add the argument info information to the `hello()` function (slide 31).
- Add a new function that multiplies a float and an integer and optionally also a boolean (slide 40/41).

# Debugging with gdb and valgrind

- `gdb --args php ....`
  - `run`
  - `bt`
  - `frame framenr`
  - `p varname`

- `valgrind php ....`

To make it easier:

- Make sure PHP is compiled in debug mode
- Disable the Zend memory manager: export `USE_ZEND_ALLOC=0`
- Disable shared extension unloading: export `ZEND_DONT_UNLOAD_MODULES=1`

# Questions

?



## Normal extensions

- Module and request init and shutdown; and post module deinitialisation
- `phpinfo()`

## Zend extensions

- statement calls
- all sorts of internal handlers: activate/deactivate functions, `op_array` handler, `op_array` constructor and destructor

# phpinfo() hook

```
PHP_MININFO_FUNCTION(xdebug)
{
    xdebug_remote_handler_info *ptr = xdebug_handlers_get();

    php_info_print_table_start();
    php_info_print_table_header(2, "xdebug support", "enabled");
    php_info_print_table_row(2, "Version", XDEBUG_VERSION);
    php_info_print_table_end();
    ...

    DISPLAY_INI_ENTRIES();
}
```

# Overloading



- Error handler: Xdebug, SOAP
- Exception throwing handler
- Compiler and executor: APC and other caches, funcall, Xdebug
- Setting of headers
- Opcodes: Xdebug, operator

# Overloading the error handler

## In MINIT:

```
/* Replace error handler callback with our own */  
xdebug_old_error_cb = zend_error_cb;  
xdebug_new_error_cb = xdebug_error_cb;
```

## In MSHUTDOWN:

```
zend_error_cb = xdebug_old_error_cb;
```

## xdebug\_error\_cb:

```
/* Error callback for formatting stack traces */  
void xdebug_error_cb(int type, const char *error_filename, const uint error_lineno, const char *format, va_list args)  
{  
    ...  
}
```

# Overloading the exception handler

## In RINIT:

```
/* Hack: We check for a soap header here, if that's existing, we don't use
 * Xdebug's error handler to keep soap fault from fucking up. */
if (XG(default_enable) && zend_hash_find(Z_ARRVAL_P(PG(http_globals)[TRACK_VARS_SERVER]), "HTTP_SOAPACTION", 16,
(void**) &dummy) == FAILURE) {
    zend_error_cb = xdebug_new_error_cb;
    zend_throw_exception_hook = xdebug_throw_exception_hook;
}
```

## xdebug\_throw\_exception\_hook:

```
static void xdebug_throw_exception_hook(zval *exception TSRMLS_DC)
{
    zval *message, *file, *line, *xdebug_message_trace, *previous_exception;
```

# Overloading the compiler and executor

## In MINIT:

```
old_compile_file = zend_compile_file;
zend_compile_file = xdebug_compile_file;
xdebug_old_execute = zend_execute;
zend_execute = xdebug_execute;
xdebug_old_execute_internal = zend_execute_internal;
zend_execute_internal = xdebug_execute_internal;
```

## In MSHUTDOWN:

```
zend_compile_file = old_compile_file;
zend_execute = xdebug_old_execute;
zend_execute_internal = xdebug_old_execute_internal;
```

## xdebug\_compile\_file:

```
/* {{{ zend_op_array srm_compile_file (file_handle, type)
 * This function provides a hook for the execution of bananas */
zend_op_array *xdebug_compile_file(zend_file_handle *file_handle, int type TSRMLS_DC)
{
    zend_op_array *op_array;
    op_array = old_compile_file(file_handle, type TSRMLS_CC);
    ...
    return op_array;
}
/* }}} */
```

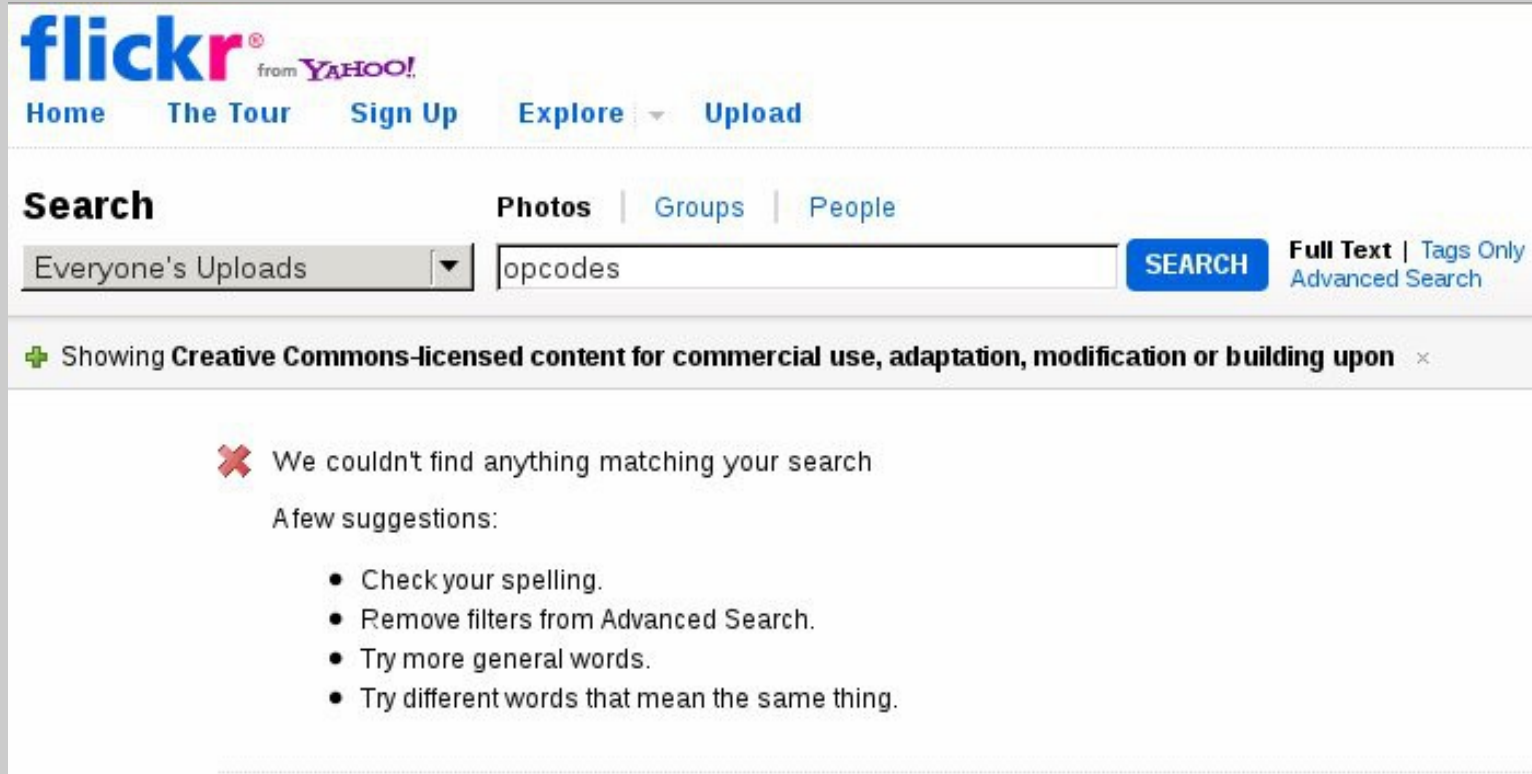
## xdebug\_execute:

```
void xdebug_execute(zend_op_array *op_array TSRMLS_DC)
{
    ...
    if (op_array && op_array->filename && strcmp("xdebug://debug-eval", op_array->filename)
        xdebug_old_execute(op_array TSRMLS_CC);
        return;
    }
    ...
}
```

## xdebug\_execute\_internal:

```
void xdebug_execute_internal(zend_execute_data *current_execute_data, int return_value_used TSRMLS_DC)
{
    zend_execute_data *edata = EG(current_execute_data);
    ...
    XG(level)++;
    if (XG(level) == XG(max_nesting_level)) {
        php_error(E_ERROR, "Maximum function nesting level of '%ld' reached, aborting!", XG(max_nesting_level));
    }
}
```

# Opcode overloading



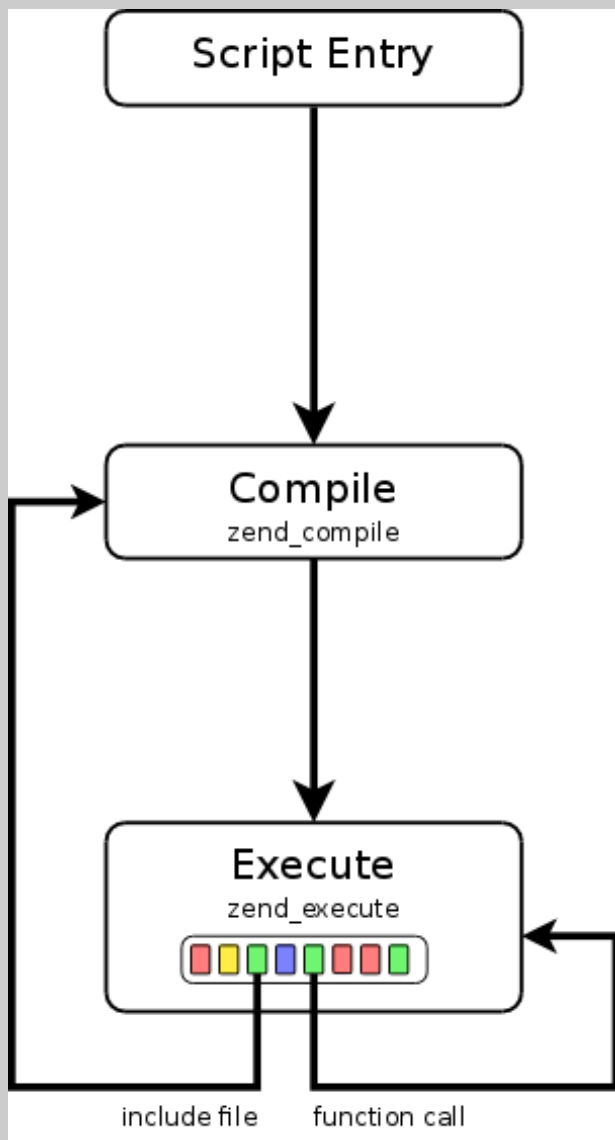
The screenshot shows the Flickr website's search interface. At the top left is the Flickr logo with the text "from YAHOO!". Navigation links include "Home", "The Tour", "Sign Up", "Explore", and "Upload". The search bar is set to "Everyone's Uploads" and contains the text "opcodes". A blue "SEARCH" button is to the right of the input field. Further right are links for "Full Text | Tags Only" and "Advanced Search". Below the search bar, a green plus icon indicates "Showing Creative Commons-licensed content for commercial use, adaptation, modification or building upon". The search results area shows a red "X" icon and the message "We couldn't find anything matching your search". Below this, it says "A few suggestions:" followed by a bulleted list:

- Check your spelling.
- Remove filters from Advanced Search.
- Try more general words.
- Try different words that mean the same thing.

Opcodes you say?

# Executing

In a diagram



# Compiling: Diagram

```
<?php
function normalizeColorArray($array)
{
    foreach (array_keys($array) as $key)
    {
        $array[$key] = (float) $array[$key]/255;
    }

    return $array;
}

function rgbToCMYK($rgbArray)
{
    $cya = 1 - min(1, max((float) $rgbArray['r'], 0));
    $mag = 1 - min(1, max((float) $rgbArray['g'], 0));
    $yel = 1 - min(1, max((float) $rgbArray['b'], 0));

    $min = min($cya, $mag, $yel);
    if (1 - $min == 0)
    {
        return array('c' => 1, 'm' => 1,
    }

    return array('c' => ($cya - $min) / (1 - $min),
                'm' => ($mag - $min) / (1 - $min),
                'y' => ($yel - $min) / (1 - $min),
                'k' => $min );
}

function rgbToCMYK2($r, $g, $b)
{
    return e2Nacht::rgbToCMYK(array('r' => $r, 'g' => $g, 'b' => $b));
}
?>
```

Parse

Compile  
zend\_compile



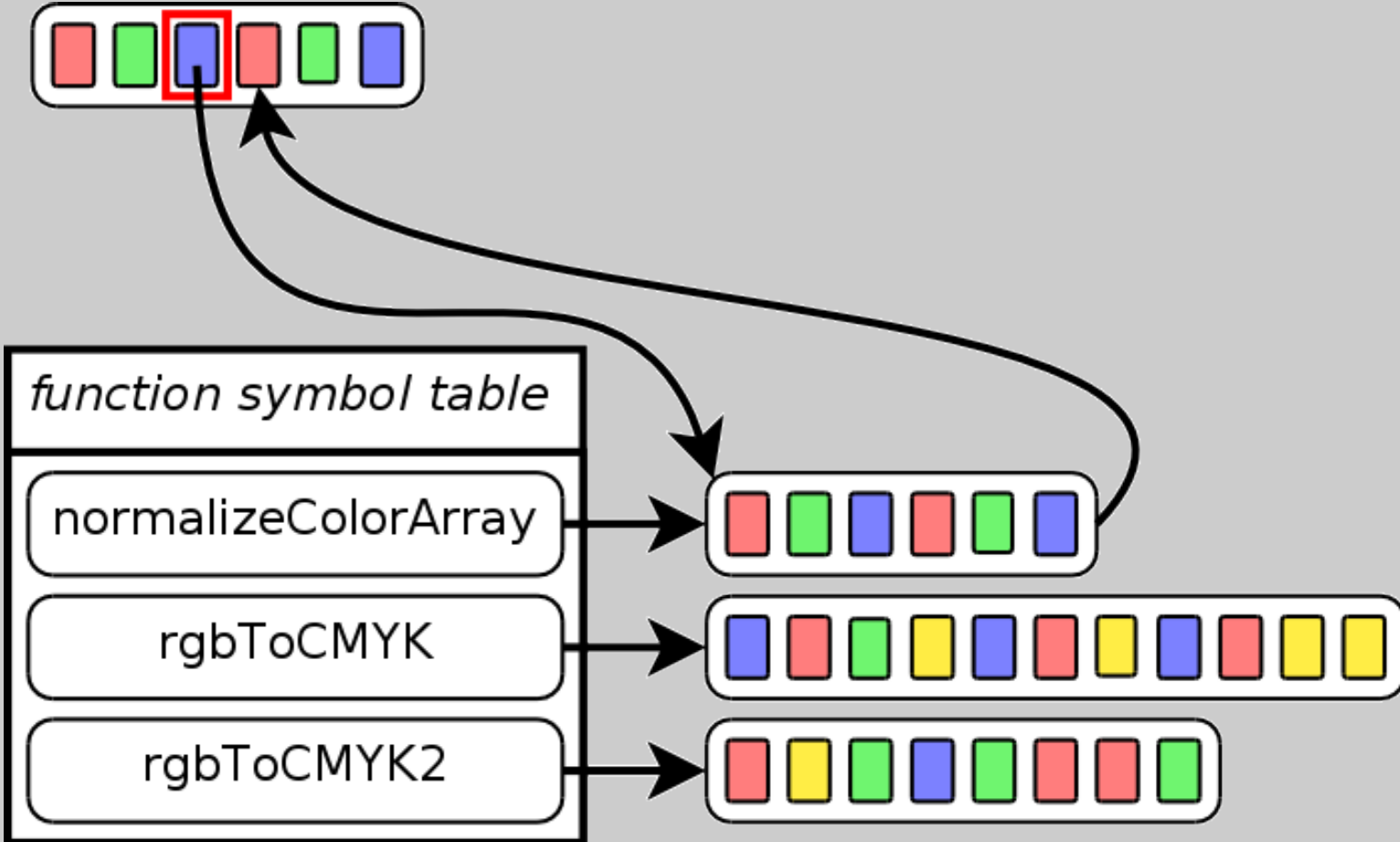
class symbol table

--

function symbol table

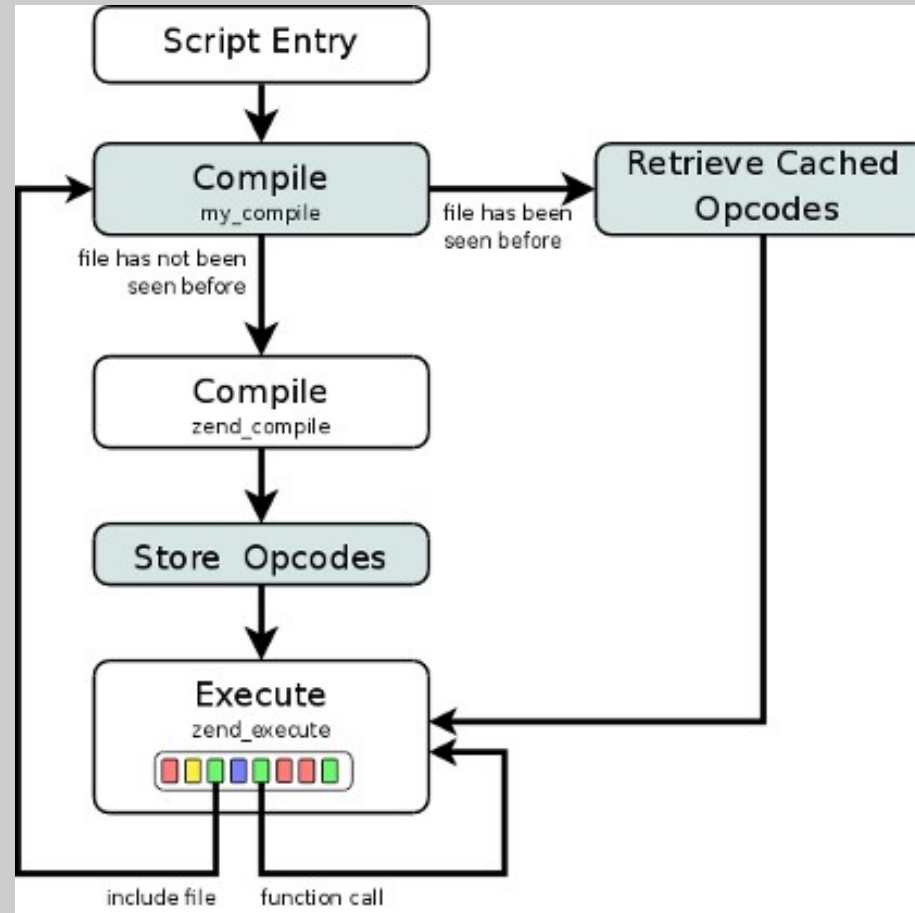
normalizeColorArray	→
rgbToCMYK	→
rgbToCMYK2	→

# Executing: Diagram



# Compiler Caches

## How it works



- In general, each source file is compiled once
- Compilation overhead becomes inconsequential
- Cache introduces its own overhead due to dynamic nature of includes

# Opcode overloading



- Scream: For making PHP ignore the @ operator
- Xdebug: For code coverage, and for making PHP ignore the @ operator
- Operator: Allows for operator overloading, but also demands that you hand over your first-born

# Overloading opcodes

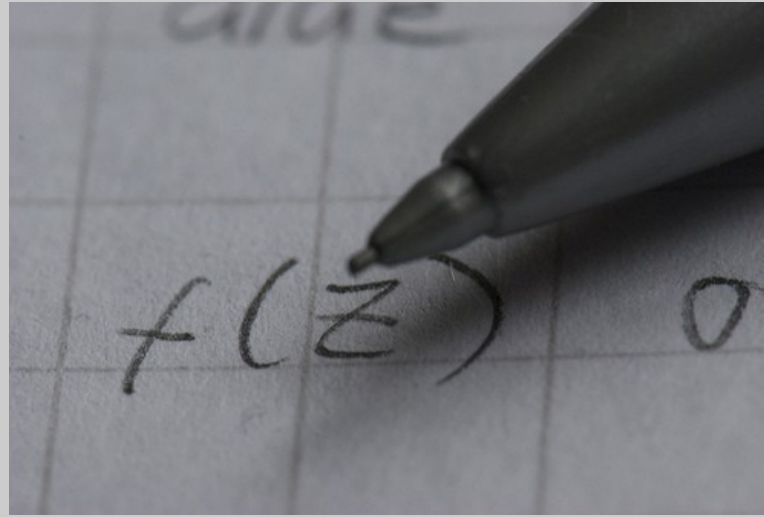
## In MINIT:

```
zend_set_user_opcode_handler(ZEND_BEGIN_SILENCE, xdebug_silence_handler);  
zend_set_user_opcode_handler(ZEND_END_SILENCE, xdebug_silence_handler);
```

## xdebug\_silence\_handler:

```
static int xdebug_silence_handler(ZEND_OPCODE_HANDLER_ARGS)  
{  
    if (XG(do_scream)) {  
        execute_data->opline++;  
        return ZEND_USER_OPCODE_CONTINUE;  
    }  
    return ZEND_USER_OPCODE_DISPATCH;  
}
```

# Replacing functions



- Replaces a function pointer in the symbol table with a new one
- You can also use it for methods
- Xdebug uses it to replace `var_dump` with `xdebug_var_dump`
- You can only replace internal functions (unless you're `runkit/classkit`)

# Overloading functions

## In RINIT:

```
zend_function *orig;

XG(var_dump_overloaded) = 0;
if (XG(overload_var_dump)) {
    zend_hash_find(EG(function_table), "var_dump", 9, (void **)&orig);
    XG(orig_var_dump_func) = orig->internal_function.handler;
    orig->internal_function.handler = zif_xdebug_var_dump;
    XG(var_dump_overloaded) = 1;
}
```

## In ZEND\_MODULE\_POST\_ZEND\_DEACTIVATE\_D:

```
zend_function *orig;

if (XG(var_dump_overloaded)) {
    zend_hash_find(EG(function_table), "var_dump", 9, (void **)&orig);
    orig->internal_function.handler = XG(orig_var_dump_func);
}
```

And `xdebug_var_dump()` is just defined as a normal function.



- Wrappers: Wrap around file reading and writing operations with a "protocol" (f.e.: `http://compress.bzip2`)
- Filters: On the fly encrypting/decrypting filter (f.e.: `mcrypt_filter`)



```
<?php  
$marmiteIsFor = 'trashcan';  
echo 'Hello World.';  
?>
```



```
<?php  
£marmiteIsFor = 'biscuits';  
announce 'Good morrow, fellow subjects of the Crown.';  
?>
```

- Sadly, this you can't do.
- Or can you?
- Facebook's xhp works as a pre-processor...

# Why extensions?

## Wrapping libraries



- General use libraries: re2
- Specific use libraries: cybermut, pam
- Libraries that you have written yourself, or your company
- Missing database support, or new webscale things like mongodb

# Why extensions?

They are faster than raw PHP code



- It's just too slow in normal PHP: ssh2, openssl
- Specific uses: QuickHash, StumbleCache

# Why extensions?

## Performance bottlenecks



- Your code is really slow, and need something quicker: twig

# Why extensions?

Impossible in just PHP



- Not everything is possible in pure PHP: apc, com, xdebug, xhprof

# Questions

?

Derick Rethans - [derick@derickrethans.nl](mailto:derick@derickrethans.nl) - twitter:  
@derickr

<http://derickrethans.nl/talks.html>

<http://joind.in/3776>

- These Slides: <http://derickrethans.nl/talks.php>
- VLD: <http://www.derickrethans.nl/vld.php>
- Xdebug: <http://xdebug.org>
- Included: <http://t3.dotgnu.info/blog/tags/included/>
- Extending and Embedding PHP ('Sara's Book'):  
<http://www.amazon.com/Extending-Embedding-PHP-Sara-Golemon/dp/067232704X>
- Tutorial on Zend's devzone:  
<http://devzone.zend.com/article/1021>