


Fonctionnement de PHP

-

Internals et extensions

Hello, I'm Julien Pauli :)

- @julienpauli 
 - <http://julien-pauli.developpez.com>
- IRC Freenode, EFNet
 - @jpauli
- Software architect – PHP Guru
- Working at Comuto
 - <http://www.covoiturage.fr>
- OSS Contributor
 - PHP Contributor – <http://doc.php.net/fr>
 - jpauli@php.net

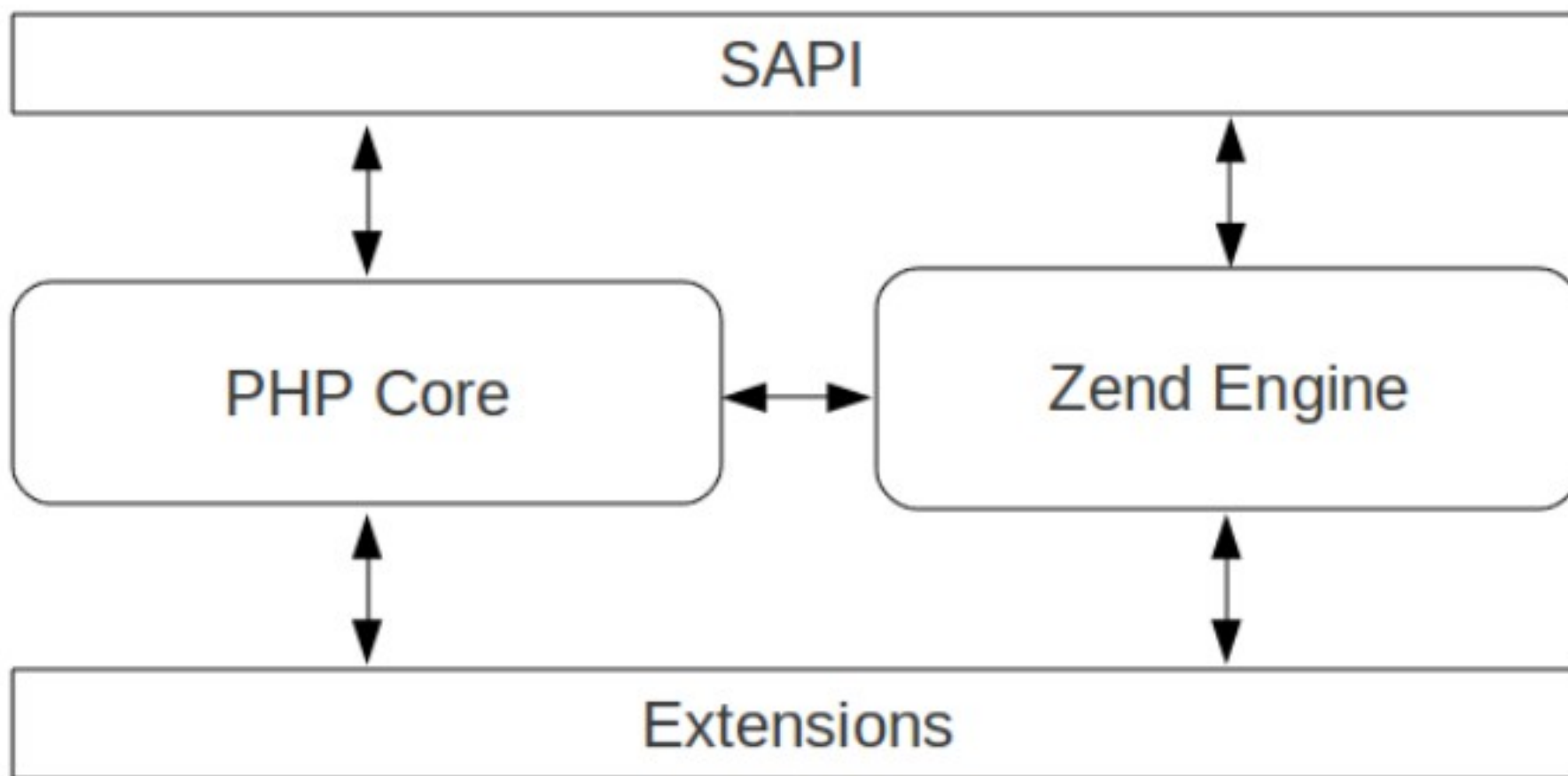


PHP, internals & extensions

- PHP vu de l'intérieur
 - Les différents organes
 - L'architecture interne
- Présentation du système d'extensions
 - Les types d'extensions
 - Le mécanisme de compilation et de chargement
- Exemples d'extensions
 - pecl
 - safe-made
- Conclusion

PHP : les organes

- 820.000 lignes de code C
 - Conçues pour représenter des couches qui glissent l'une sur l'autre



PHP : SAPI

- Server Application Interface
 - Point d'entrée vers le moteur PHP
- CLI, CGI, FastCGI, APXS(Apache), Built-in-Server
- Buts de la SAPI :
 - gérer les en-têtes de la requête d'entrée ;
 - gérer les données de la requête d'entrée : POST, COOKIE ;
 - écrire sur la sortie, la SAPI est peu liée au système de tampon de PHP (ob) : elle écrit dans la couche basse,
 - et permet de la vider (flush) ;
 - démarrer/Arrêter le ZendEngine et le PHPCore.

sapi_module_struct

```
struct _sapi_module_struct {
    char *name;
    char *pretty_name;

    int (*startup)(struct _sapi_module_struct *sapi_module); // fonction de démarrage
    int (*shutdown)(struct _sapi_module_struct *sapi_module); // fonction de de fermeture

    int (*activate)(TSRMLS_D); // fonction d'activation (environnement de requête)
    int (*deactivate)(TSRMLS_D); // fonction de désactivation (environnement de requête)

    int (*ub_write)(const char *str, unsigned int str_length TSRMLS_DC);
    void (*flush)(void *server_context);
    struct stat *(*get_stat)(TSRMLS_D);
    char *(*getenv)(char *name, size_t name_len TSRMLS_DC);

    void (*sapi_error)(int type, const char *error_msg, ...);

    int (*header_handler)(sapi_header_struct *sapi_header, sapi_header_op_enum op,
                          sapi_headers_struct *sapi_headers TSRMLS_DC);
    int (*send_headers)(sapi_headers_struct *sapi_headers TSRMLS_DC);
    void (*send_header)(sapi_header_struct *sapi_header, void *server_context TSRMLS_DC);

    int (*read_post)(char *buffer, uint count_bytes TSRMLS_DC);
    char *(*read_cookies)(TSRMLS_D);
```

Exemple de SAPI Apache2

- http://lxr.php.net/xref/PHP_5_4/sapi/apache2handler/sapi_apache2.c

```
zend_first_try {  
    // ...  
    zend_file_handle zfd;  
  
    zfd.type = ZEND_HANDLE_FILENAME;  
    zfd.filename = (char *) r->filename;  
    zfd.free_filename = 0;  
    zfd.opened_path = NULL;  
  
    if (!parent_req) {  
        php_execute_script(&zfd TSRMLS_CC);  
    } else {  
        zend_execute_scripts(ZEND_INCLUDE TSRMLS_CC, NULL, 1, &zfd);  
    }  
    apr_table_set(r->notes, "mod_php_memory_usage",  
        apr_psprintf(ctx->r->pool, "%zu", zend_memory_peak_usage(1 TSRMLS_CC)));  
}  
} zend_end_try();
```

PHP Core

- Un genre de fourre-tout
 - gestion du réseau via une API de flux (streams) ;
 - algorithme de tri (MergeSort) ;
 - lecture de la ligne de commandes (getopt()) ;
 - open_basedir, safe_mode et parsing de chemins (fopen) ;
 - buffer de sortie (ob_) ;
 - ticks ;
 - logos (php et zend) ;
 - fonctions de lecture des données de la requête (passées à la SAPI comme fonctions par défaut) ;
 - variables globales, parsing d'url, outils de traitement des chaînes, gestion des erreurs ;
 - "Tout ce qui n'est pas extension"

PHPCore Globales

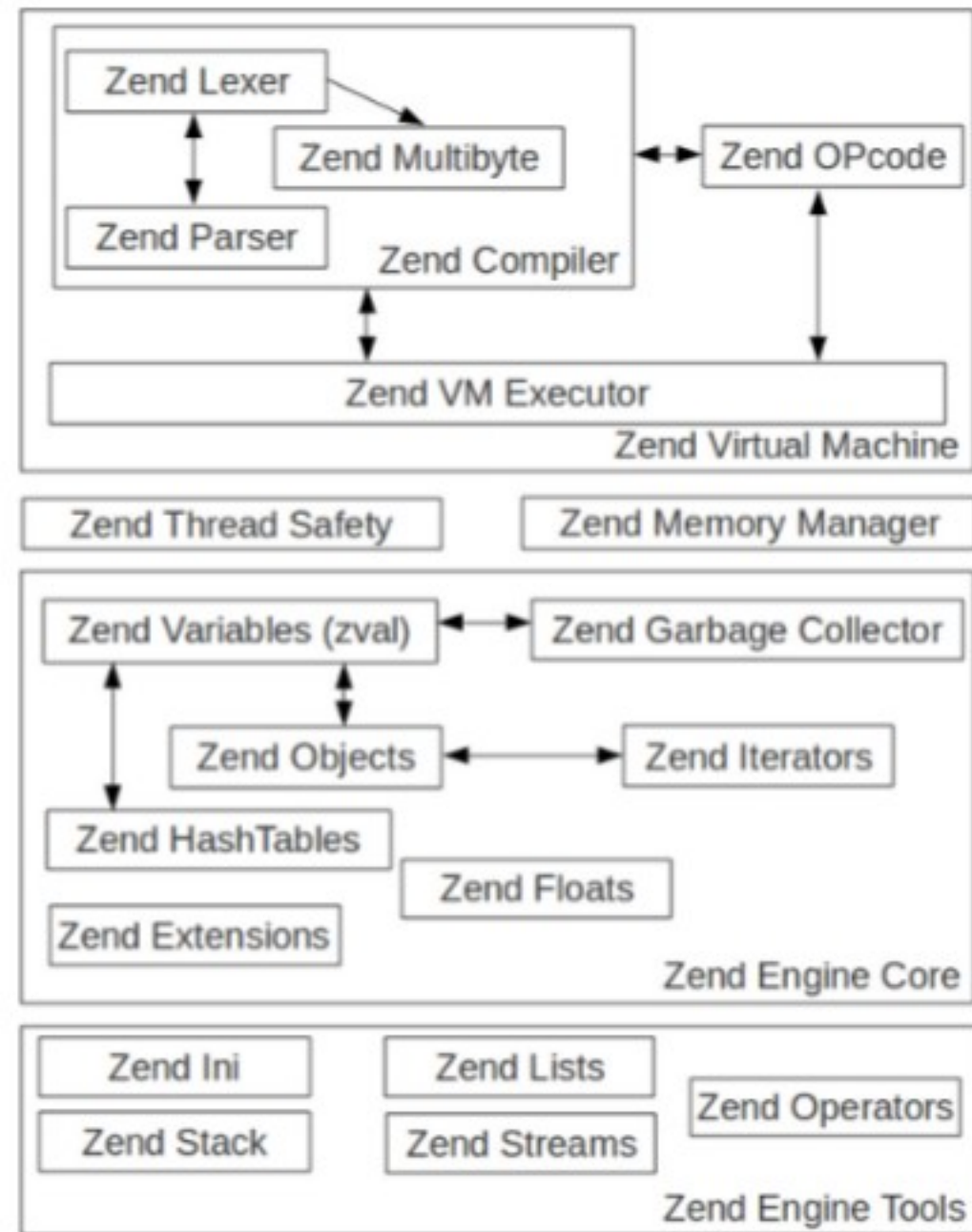
- Globales accessibles de partout
 - macro PG()

```
if (PG(register_argc_argv)) {
    php_build_argv(SG(request_info).query_string,
        PG(http_globals)[TRACK_VARS_SERVER]);
}
```

```
struct _php_core_globals {
    zend_bool magic_quotes_gpc;
    zend_bool magic_quotes_runtime;
    zend_bool magic_quotes_sybase;
    zend_bool safe_mode;
    zend_bool allow_call_time_pass_reference;
    zend_bool implicit_flush;
    long output_buffering;
    char *safe_mode_include_dir;
    zend_bool safe_mode_gid;
    zend_bool sql_safe_mode;
    zend_bool enable_dl;
    char *output_handler;
    char *unserialize_callback_func;
    long serialize_precision;
    char *safe_mode_exec_dir;
    long memory_limit;
    long max_input_time;
    ... ..
}
```

Zend Engine

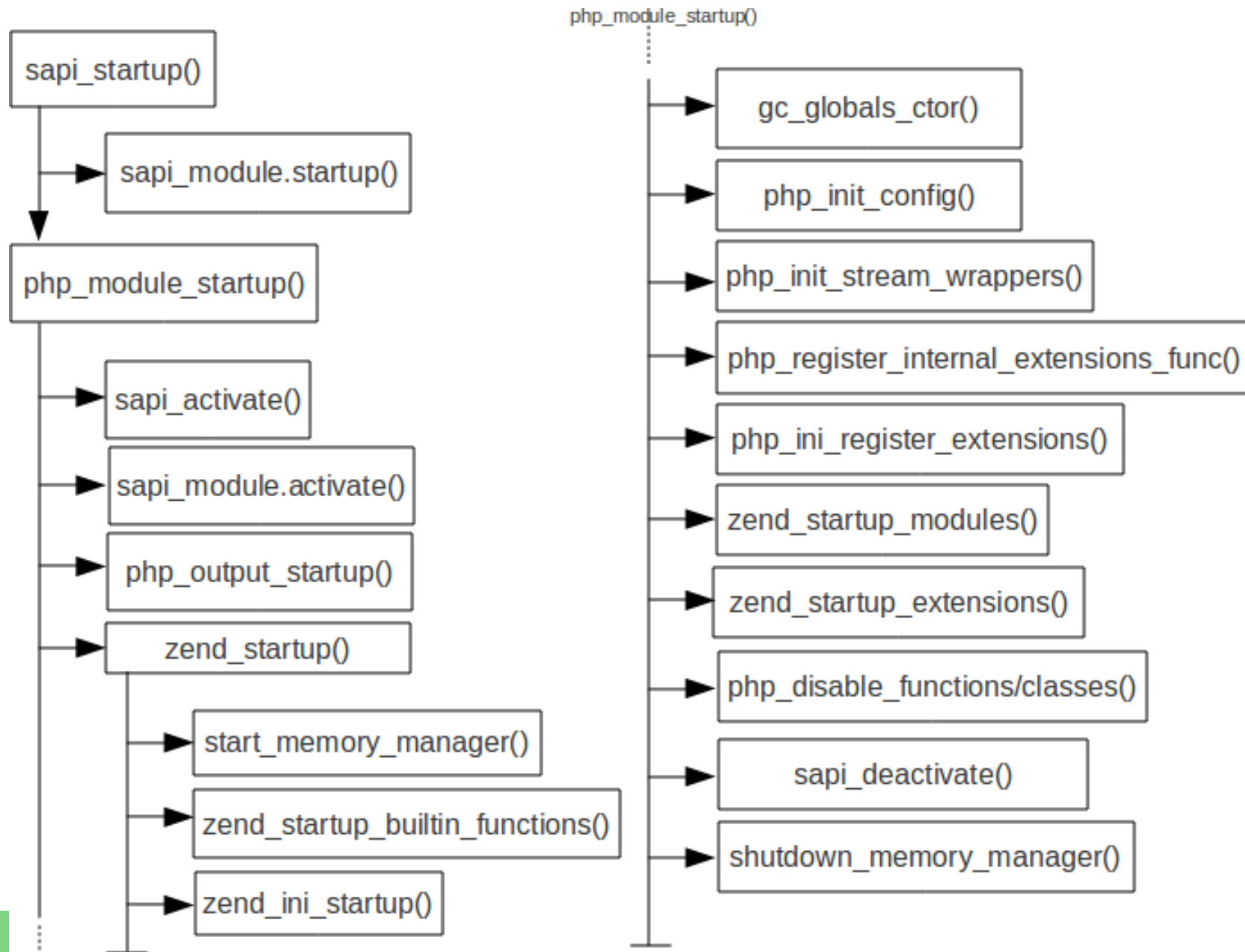
- Core
- Tools
- VirtualMachine
- Thread Safety
- Memory Manager



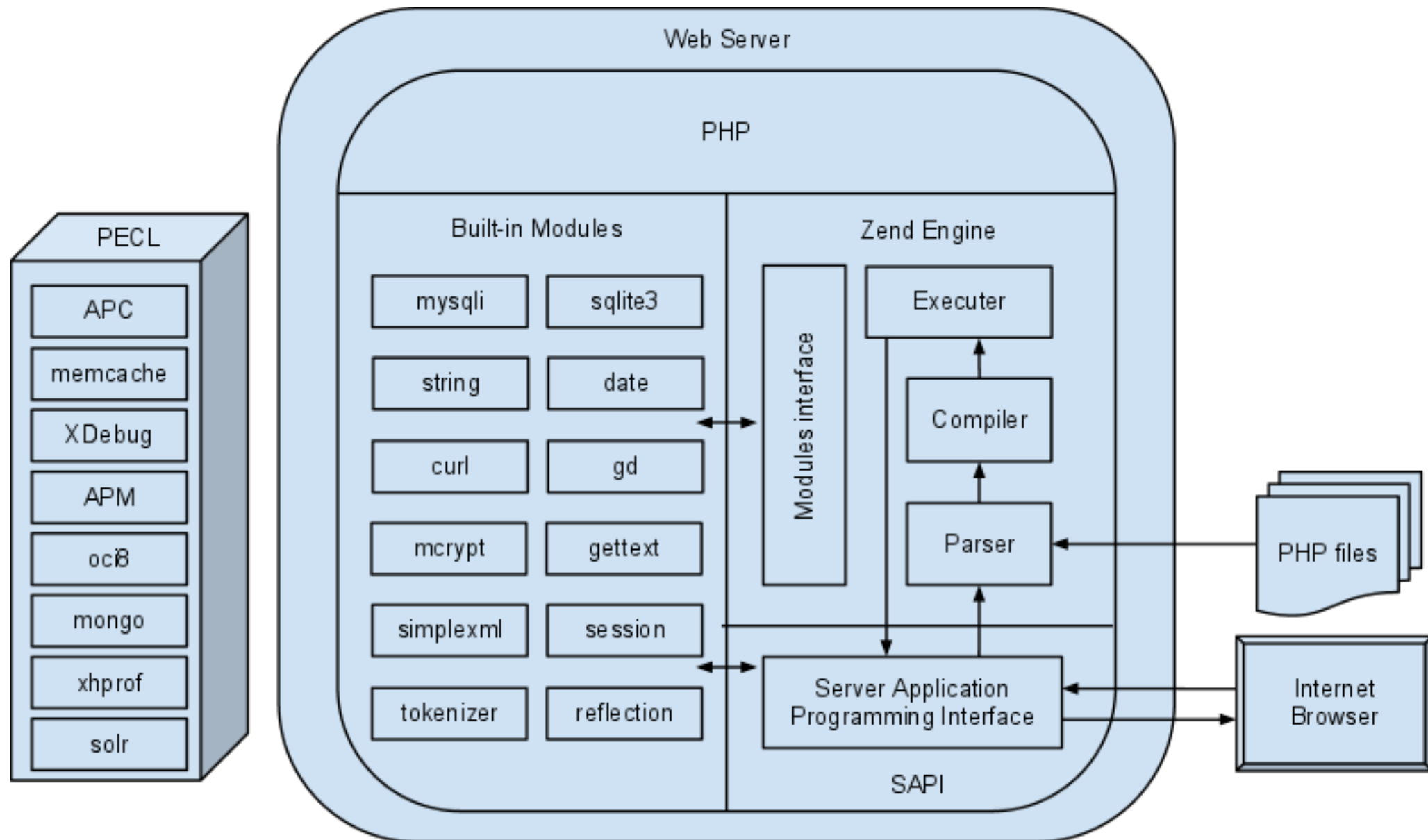
Zend Engine

- compilateur : analyseurs lexical/syntaxique (re2c/bison), syntaxe PHP et syntaxe INI
- exécuteur : la partie la plus importante, la machine virtuelle de PHP
- gestionnaire de mémoire : Zend Memory Manager
- définitions des concepts de "classe" et de "fonction" pour PHP
- définitions et gestion des variables PHP : zval, références et types PHP
- une partie de la gestion de la concurrence des threads (Thread Safety : TS)
- zend_gc : le ramasse-miettes (garbage collector) de PHP
- gestion avancée du type flottant de C
- moteur de gestion des erreurs et des exceptions
- moteur d'extensions
- hashtables (array PHP), algorithmes de tri, objets PHP et itérateurs

Life Cycle



Extensions



Une extension PHP, pourquoi ?

- Code C
 - Rapide, très rapide
- Permet de rajouter à PHP :
 - Des fonctions
 - Des classes
 - Des constantes
 - Des paramètres dans php.ini
 - Des gestionnaires de flux
 - Des variables, globales (burk)
- Permet aussi de lier PHP à une lib du système
 - ext/simpleXML et ext/dom lient PHP à libxml
 - ext/gd lie PHP à libpng et libjpeg

PHP est lié à l'OS via ses extensions

```
julien@jpauli:~$ ldd $(which php)
linux-vdso.so.1 => (0x00007fff5abff000)
libcrypt.so.1 => /lib/x86_64-linux-gnu/libcrypt.so.1 (0x00007f8398338000)
libz.so.1 => /lib/x86_64-linux-gnu/libz.so.1 (0x00007f8398120000)
libexslt.so.0 => /usr/lib/libexslt.so.0 (0x00007f8397f0a000)
libtidy-0.99.so.0 => /usr/lib/libtidy-0.99.so.0 (0x00007f8397cad000)
libedit.so.2 => /usr/lib/libedit.so.2 (0x00007f8397a86000)
libncurses.so.5 => /lib/libncurses.so.5 (0x00007f8397841000)
librt.so.1 => /lib/x86_64-linux-gnu/librt.so.1 (0x00007f8397639000)
libmcrypt.so.4 => /usr/lib/libmcrypt.so.4 (0x00007f8397406000)
libltdl.so.7 => /usr/lib/libltdl.so.7 (0x00007f83971fb000)
libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f8396ff7000)
libldap_r-2.4.so.2 => /usr/lib/libldap_r-2.4.so.2 (0x00007f8396dac000)
liblber-2.4.so.2 => /usr/lib/liblber-2.4.so.2 (0x00007f8396b9d000)
libpng12.so.0 => /usr/lib/x86_64-linux-gnu/libpng12.so.0 (0x00007f8396976000)
libbz2.so.1.0 => /lib/libbz2.so.1.0 (0x00007f8396766000)
libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007f83964e0000)
libnsl.so.1 => /lib/x86_64-linux-gnu/libnsl.so.1 (0x00007f83962c6000)
libssl.so.0.9.8 => /lib/libssl.so.0.9.8 (0x00007f8396073000)
libcrypto.so.0.9.8 => /lib/libcrypto.so.0.9.8 (0x00007f8395ce3000)
libcurl-gnutls.so.4 => /usr/lib/libcurl-gnutls.so.4 (0x00007f8395a8f000)
libicui18n.so.44 => /usr/lib/libicui18n.so.44 (0x00007f83956d1000)
libicuuc.so.44 => /usr/lib/libicuuc.so.44 (0x00007f8395388000)
libicudata.so.44 => /usr/lib/libicudata.so.44 (0x00007f8394348000)
```

...

Tout est extension

- Dans PHP, tout ce qui n'est pas Core et ZendEngine est une extension.
 - Une extension n'est pas forcément un .so (.dll)
 - Une extension peut être compilée et liée "statiquement", il n'y a donc pas de .so
 - Une extension ajoute des fonctionnalités à PHP
- Vocabulaire : une extension s'appelle ext/{something}
- Le moteur est tellement extensible, que tout est extension :
 - strings, array, numbers ...
 - regexp, dates
 - SPL ...

ext/standard

standard

Dynamic Library Support	enabled
Path to sendmail	/usr/sbin/sendmail -t -i

Directive	Local Value	Master Value
assert.active	1	1
assert.bail	0	0
assert.callback	<i>no value</i>	<i>no value</i>
assert.quiet_eval	0	0
assert.warning	1	1
auto_detect_line_endings	0	0
default_socket_timeout	60	60
from	<i>no value</i>	<i>no value</i>
safe_mode_allowed_env_vars	PHP_	PHP_
safe_mode_protected_env_vars	LD_LIBRARY_PATH	LD_LIBRARY_PATH
url_rewriter.tags	a=href,area=href,frame=src,input=src,form=fakeentry	a=href,area=href,frame=src,input=src,form=fakeentry
user_agent	<i>no value</i>	<i>no value</i>

- fonctions array_ ; fonctions str_ ; sérialisation PHP
- tout ce qui concerne les urls, les caractères dans les chaines... ;
- transtypages des variables, bases numériques, locale ;
- crypt, base64, md5, assert, crc32, phpinfo... ;
- gestion des fichiers et du système de fichiers ;
- fonctions relatives au réseau, flux.

ext/core

Core

PHP Version	5.3.8
-------------	-------

Directive	Local Value	Master Value
allow_call_time_pass_reference	Off	Off
allow_url_fopen	On	On
allow_url_include	Off	Off
always_populate_raw_post_data	Off	Off
arg_separator.input	&	&
arg_separator.output	&	&
asp_tags	Off	Off
auto_append_file	<i>no value</i>	<i>no value</i>
auto_globals_jit	On	On

- `func_get_args()`, `gc_enable()`, `memory_get_usage()`, `zend_version()`

Extensions obligatoires

- Au minimum, PHP doit disposer de
 - core
 - date
 - ereg
 - pcre
 - reflection
 - SPL
 - standard
- Tout le reste est optionnel
- Un PHP "minimal" consomme très très peu de mémoire

2 types d'extensions

- Extensions
 - Ok, on connaît
 - extension = myext.so (utilise extension_dir)
- Zend Extensions
 - Plus puissantes
 - Peuvent remplacer des fonctions du ZendEngine
 - non compatibles avec dl()
 - zend_extension = full/path/to/myext.so
- Mais les 2 se compilent de la même manière
 - C'est du code C, point...
 - Il a simplement des rôles différents
 - Et une profondeur d'interaction dans PHP différente

Structure d'une extension PHP

```

struct _zend_module_entry {
    unsigned short size;
    unsigned int zend_api;
    unsigned char zend_debug;
    unsigned char zts;
    const struct _zend_ini_entry *ini_entry;
    const struct _zend_module_dep *deps;
    const char *name;
    const struct _zend_function_entry *functions;
    int (*module_startup_func)(INIT_FUNC_ARGS);
    int (*module_shutdown_func)(SHUTDOWN_FUNC_ARGS);
    int (*request_startup_func)(INIT_FUNC_ARGS);
    int (*request_shutdown_func)(SHUTDOWN_FUNC_ARGS);
    void (*info_func)(ZEND_MODULE_INFO_FUNC_ARGS);
    const char *version;
    size_t globals_size;
    void* globals_ptr;
    void (*globals_ctor)(void *global TSRMLS_DC);
    void (*globals_dtor)(void *global TSRMLS_DC);
    int (*post_deactivate_func)(void);
    int module_started;
    unsigned char type;
    void *handle;
    int module_number;
    char *build_id;
};
  
```

Structure d'une Zend Extension

```
struct _zend_extension {  
    char *name;  
    char *version;  
    char *author;  
    char *URL;  
    char *copyright;  
    startup_func_t startup;  
    shutdown_func_t shutdown;  
    activate_func_t activate;  
    deactivate_func_t deactivate;  
    message_handler_func_t message_handler;  
    op_array_handler_func_t op_array_handler;  
    statement_handler_func_t statement_handler;  
    fcall_begin_handler_func_t fcall_begin_handler;  
    fcall_end_handler_func_t fcall_end_handler;  
    op_array_ctor_func_t op_array_ctor;  
    op_array_dtor_func_t op_array_dtor;  
    int (*api_no_check)(int api_no);  
    int (*build_id_check)(const char* build_id);  
    void *reserved3;  
    void *reserved4;  
    void *reserved5;  
    void *reserved6;  
    void *reserved7;  
    void *reserved8;  
    DL_HANDLE handle;  
    int resource_number;  
};
```

Introspecter une extension

- php --ri
- php --re
- php -m

```
julien@jpauli:~$ php --re json
Extension [ <persistent> extension #19 json version 1.2.1 ] {

  - Constants [12] {
    Constant [ integer JSON_HEX_TAG ] { 1 }
    Constant [ integer JSON_HEX_AMP ] { 2 }
    Constant [ integer JSON_HEX_APOS ] { 4 }
    ...
  }
  - Functions {
    Function [ <internal:json> function json_encode ] {
      - Parameters [2] {
        Parameter #0 [ <required> $value ]
        Parameter #1 [ <optional> $options ]
      }
    }
    Function [ <internal:json> function json_decode ] {
      - Parameters [3] {
        Parameter #0 [ <required> $json ]
        Parameter #1 [ <optional> $assoc ]
        Parameter #2 [ <optional> $depth ]
      }
    }
  }
}
```

Introspecter une extension

```

ReflectionExtension implements Reflector {

    /* Properties */
    public $ReflectionExtension->name ;

    /* Methods */
    final private void ReflectionExtension::__clone ( void )
    ReflectionExtension::__construct ( string $name )
    public static string ReflectionExtension::export ( string $name [, string $return = false ] )
    public array ReflectionExtension::getClasses ( void )
    public array ReflectionExtension::getClassNames ( void )
    public array ReflectionExtension::getConstants ( void )
    public array ReflectionExtension::getDependencies ( void )
    public array ReflectionExtension::getFunctions ( void )
    public array ReflectionExtension::getINIEntries ( void )
    public string ReflectionExtension::getName ( void )
    public string ReflectionExtension::getVersion ( void )
    public void ReflectionExtension::info ( void )
    public void ReflectionExtension::isPersistent ( void )
    public void ReflectionExtension::isTemporary ( void )
    public string ReflectionExtension::__toString ( void )
}

```


Lier une extension

- Statiquement
 - L'extension sera liée au binaire PHP
 - Pratique lorsqu'on l'utilise souvent
 - Alourdit l'empreinte mémoire de PHP en permanence
- Dynamiquement
 - L'extension sera sortie dans un .so
 - Chargé via PHP.ini à chaque démarrage de PHP
 - Permet de désactiver l'extension sans recompiler
 - Démarrage de PHP plus lent

Compiler une extension

```
'./configure' \  
'--with-apxs2=/usr/local/apache2/bin/apxs' \  
'--with-bz2' \  
'--with-config-file-scan-dir=/usr/local/etc' \  
'--with-curl' \  
'--with-gd' \  
'--with-gettext' \  
'--with-ldap' \  
'--with-mcrypt' \  
'--with-mysqli=mysqlnd' \  
'--with-mysql=mysqlnd' \  
'--with-pdo-mysql=mysqlnd' \  
'--with-openssl' \  
'--with-libedit' \  
'--with-tidy=shared' \  
'--with-xsl' \  
'--enable-bcmath' \  
'--enable-gd-native-ttf' \  
'--enable-intl' \  
'--enable-mbstring' \  
'--enable-pcntl' \  
'--enable-sockets' \  
'--enable-soap' \  
'--enable-sqlite-utf8' \  
'--enable-zip' \  
'--disable-cgi' \  

```

make && make install

Outils nécessaire pour compiler une ext

- Outils de compilation (GCC)
 - Paquet "build-essentials" sous Debian
- autoconf: 2.13 (2.59+ for PHP 5.4+)
- automake: 1.4+
- libtool: 1.4.x+ (except 1.4.2)

- PHP compilé en --debug
- Binaire "phpize"
- Script "php-config"
- script "ext_skel.sh"
 - <http://svn.php.net/viewvc/php/php-src/trunk/ext/skeleton/>

Processus de développement

- Outils utiles :
 - Un éditeur de code
 - gdb
 - valgrind

```
$ cd myext
$ edit README, CREDITS, RELEASE-0.1.0
$ edit config.m4 and config.w32
$ phpize
$ ./configure
$ make
$ php -n -dextension_dir=`pwd`/modules -dextension=myext.so myext.php
$ edit myext.c to correct errors
$ make
```

Tout est dans le config.m4

- Fichier utilisé par autoconf
 - Déclare les dépendances
 - Déclare les switch
 - Déclare les fichiers à compiler

```
dnl
dnl $Id: config.m4 282645 2009-06-23 13:09:34Z johannes $
dnl

PHP_ARG_ENABLE(json, whether to enable JavaScript Object Serialization support,
[ --disable-json          Disable JavaScript Object Serialization support], yes)

if test "$PHP_JSON" != "no"; then
    AC_DEFINE([HAVE_JSON], 1, [whether to enable JavaScript Object Serialization support])
    AC_HEADER_STDC

    PHP_NEW_EXTENSION(json, json.c utf8_to_utf16.c utf8_decode.c JSON_parser.c, $ext_shared)
    PHP_INSTALL_HEADERS([ext/json], [php_json.h])
    PHP_SUBST(JSON_SHARED_LIBADD)
fi
```

Extension minimale

```
#ifdef HAVE_CONFIG_H
#include "config.h"
#endif

#include "php.h"
#include "php_ini.h"
#include "ext/standard/info.h"
#include "php_myext.h"

zend_module_entry myext_module_entry = {
    STANDARD_MODULE_HEADER,
    "myext",
    NULL, /* Function entries */
    NULL, /* Module init */
    NULL, /* Module shutdown */
    NULL, /* Request init */
    NULL, /* Request shutdown */
    NULL, /* Module information */
    "0.1", /* Replace with version number for your extension */
    STANDARD_MODULE_PROPERTIES
};

#ifdef COMPILE_DL_MYEXT
ZEND_GET_MODULE(myext)
#endif
```

Exemple concret (ext/json)

```
#include "php.h"  
#include "php_ini.h"  
#include "ext/standard/info.h"  
#include "ext/standard/php_smart_str.h"  
#include "utf8_to_utf16.h"  
#include "JSON_parser.h"  
#include "php_json.h"
```

```
static PHP_MINFO_FUNCTION(json);  
static PHP_FUNCTION(json_encode);  
static PHP_FUNCTION(json_decode);  
static PHP_FUNCTION(json_last_error);
```

```
static PHP_MINIT_FUNCTION(json)  
{  
    REGISTER_LONG_CONSTANT("JSON_HEX_TAG", PHP_JSON_HEX_TAG,  
                           CONST_CS | CONST_PERSISTENT);  
    REGISTER_LONG_CONSTANT("JSON_HEX_AMP", PHP_JSON_HEX_AMP,  
                           CONST_CS | CONST_PERSISTENT);  
    return SUCCESS;  
}
```

Exemple concrêt (ext/json)

```

PHP_JSON_API void php_json_encode(smart_str *buf, zval *val, int options TSRMLS_DC) /* {{{ */
{
    switch (Z_TYPE_P(val))
    {
        ... ..
        case IS_DOUBLE:
        {
            char *d = NULL;
            int len;
            double dbl = Z_DVAL_P(val);

            if (!zend_isinf(dbl) && !zend_isnan(dbl)) {
                len = sprintf(&d, 0, "%.*k", (int) EG(precision), dbl);
                smart_str_appendl(buf, d, len);
                efree(d);
            } else {
                php_error_docref(NULL TSRMLS_CC, E_WARNING, "double %.9g does not
                    conform to the JSON spec, encoded as 0", dbl);
                smart_str_appendc(buf, '0');
            }
        }
        break;
    }
}
  
```


Concrètement, les extensions existantes

- <http://pecl.php.net>
 - APC, MemCache
 - bbcode, haru, yaml, docblock
 - amfext, svn
 - apm, xdebug, xhprof

Veiller sur les extensions

PECL Announce	PHPInternals	[PECL-DEV] [ANNOUNCEMENT] lua-0.9.1 (beta) Released. - The new PECL pac
Olaf Sosa	PHPInternals	[PECL-DEV] SVN Account Request: olafsz - actualizarme en programas nuevos -
PECL Announce	PHPInternals	[PECL-DEV] [ANNOUNCEMENT] CUBRID-8.4.1.0001 (stable) Released. - The ne
PECL Announce	PHPInternals	[PECL-DEV] [ANNOUNCEMENT] CUBRID-8.4.0.0005 (stable) Released. - The ne
Stas, Gustavo (2)	PHPInternals	[PHP-DEV] Intl IDNA patch (was: Re: [PECL-DEV] libidn2 extension for php) -
PECL Announce	PHPInternals	[PECL-DEV] [ANNOUNCEMENT] CUBRID-8.3.1.0008 (stable) Released. - The ne
Sven .. David, David (39)	PHPDoc PHPInternals	[PECL-DEV] libidn2 extension for php - Hey all, sry I'm new at mailing
PECL Announce	PHPInternals	[PECL-DEV] [ANNOUNCEMENT] expect-0.3.1 (beta) Released. - The new PECL
Ferenc .. Pierre (13)	PHPInternals	[PHP-DEV] pecl sqlite - Hi. We moved the sqlite ext from core to pecl with 5.4, bu
Asimananda Mohanty	PHPInternals	[PECL-DEV] I am my own boss try it out for yourself... - p>Hey Friend! you i
Bryan, Antony (2)	PHPInternals	[PECL-DEV] PDO_INFORMIX error - pdo_informix Team, Receiving an error when
Remi Collet	PHPInternals	[PECL-DEV] Link PECL RSS - Don't really know if this is the right list to post, but,
PECL Announce	PHPInternals	[PECL-DEV] [ANNOUNCEMENT] rrd-1.0.5 (stable) Released. - The new PECL p
Johannes .. Antony (15)	PHPInternals	[PECL-DEV] Lowercase or camelcase function names? - Dear developers, I am
Alex, Israel (2)	PHPInternals	[PECL-DEV] Updating erroneous package solr-PECL 1.0.1 - ... hotmail.com > T

De l'aide ?

- Extending and embedding PHP
 - Attention, API de PHP5.1
- <http://julien-pauli.developpez.com>
- <http://www.php.net/manual/fr/internals2.php>
- <http://lxr.php.net>
- <https://wiki.php.net/internals>



Merci !

- Merci à vous !

- Nous recrutons :)
 - Passez me voir



Questions ?

