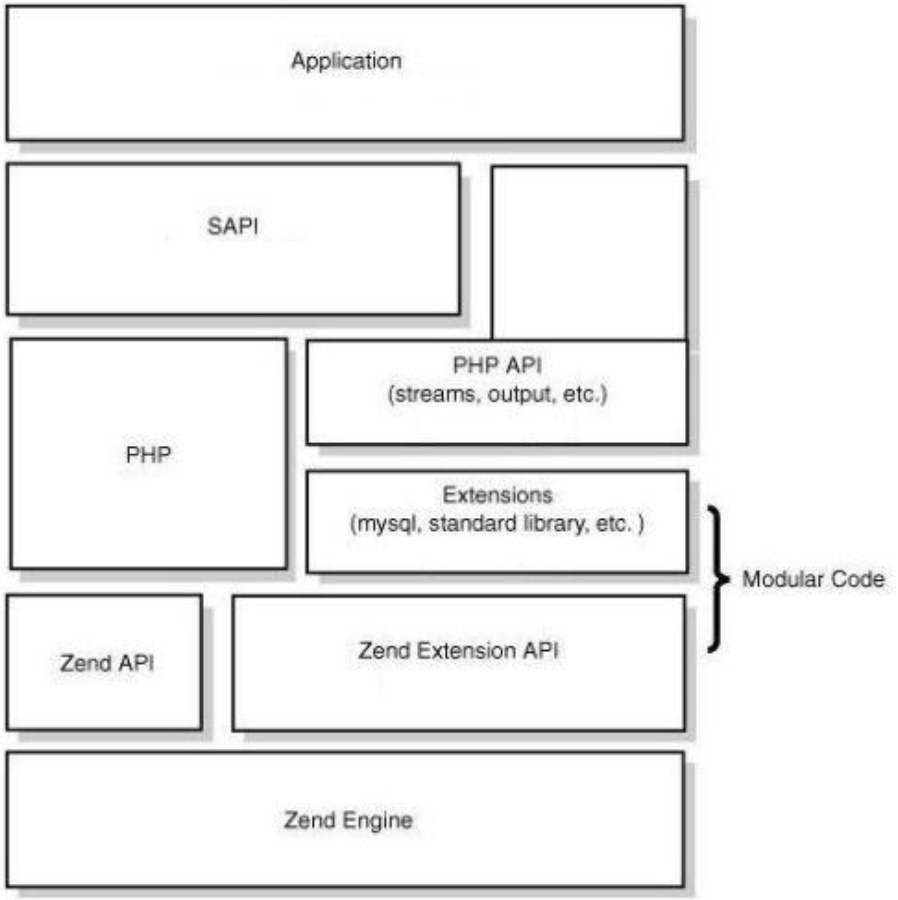
PHP EKSTENZIJE

Robert Petranovic, 2013.

PHP ARHITEKTURA

- ZEND ENGINE (ZE)
  - Syntax parsing
  - Code execution
  - Memory management
  - Variable scope
- PHP CORE
  - Komunikacija sa SAPI (apache, CLI, CGI,...)
  - Control layer (ala sandbox)
  - Stream layer (I/O)

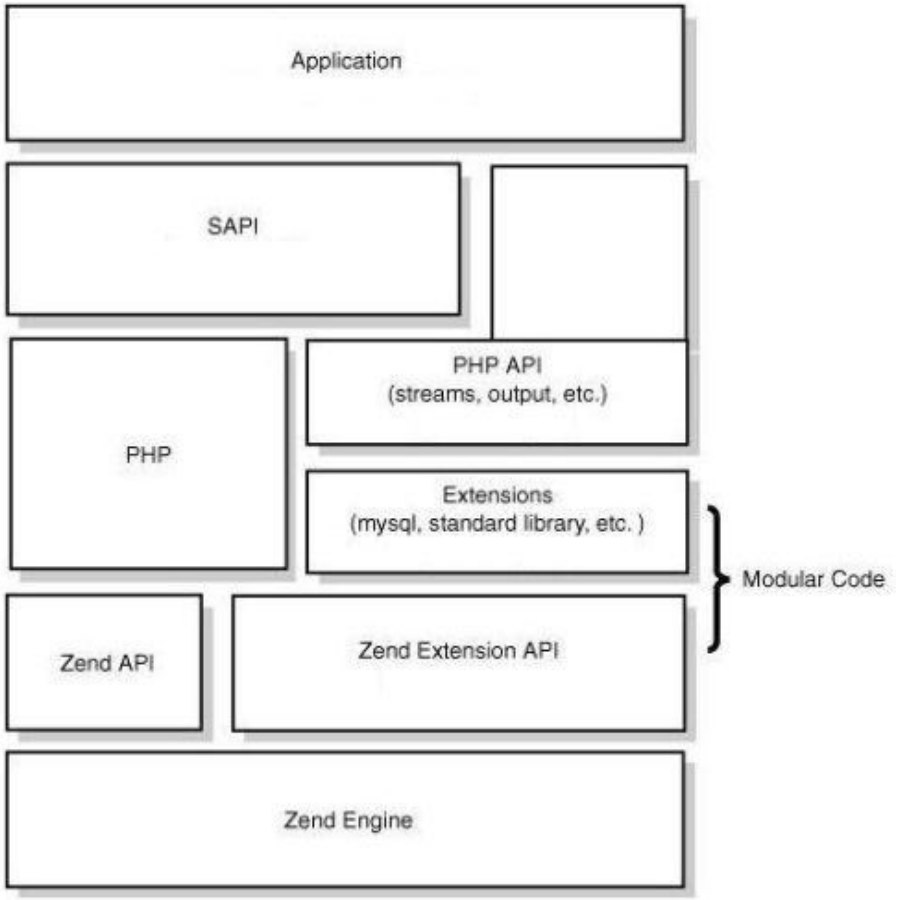
PHP ARHITEKTURA



SAPI

- Server abstraction API
- SAPI zamislite kao interface za komunikaciju PHP-a sa vanjskim svijetom.
- Npr. Apache, CGI, CLI, IIS

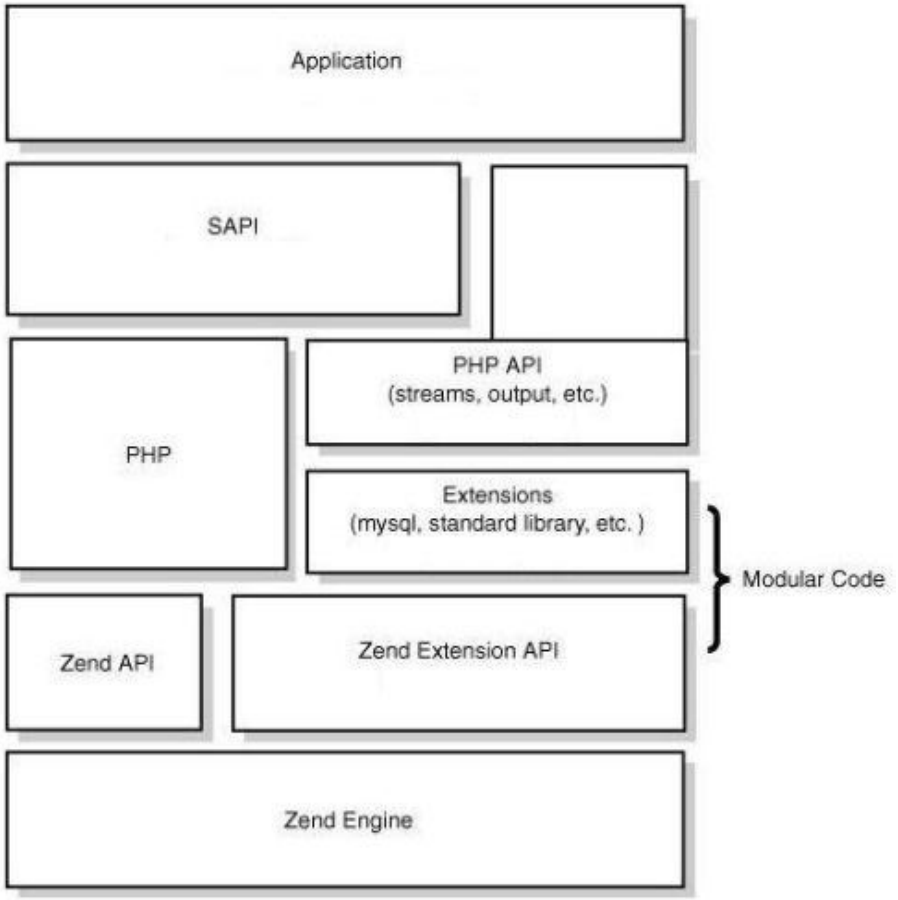
PHP ARHITEKTURA



Core PHP

- Podesava startup env.
  - Globalne varijable
  - Defaultne php.ini opcije
- Control Layer(ala sandbox)
- Streams (System I/O)

PHP ARHITEKTURA



Zend Engine

- Syntax parsing
- Code execution
- Memory management
- Variable scope

Life cycle

Primjer 1: HelloWorld

- Building PHP
- Pisanje ekstenzije
- Registriranje ekstenzije
- Buildanje ekstenzije

Compiling PHP

- Build-time dependencies
  - Potrebni libovi i njihovi development paketi
  - Najbolje pogledati u source pakete PHP-a vaše distribucije
- Konfiguracijske opcije:
  - Kopirati postojeće (piše u phpinfo)
  - Dodati: --enable-debug –enable-maintainer-zts

Registering extension

- Moramo PHP-u objasniti kako koristiti našu ekstenziju.
- Treba popuniti slijedeću zend_module_entry strukturu:

```
zend_module_entry counter_module_entry = {
    STANDARD_MODULE_HEADER,
    "counter",                  // Name
    counter_functions,          // Functions
    PHP_MINIT(counter),         // MINIT
    NULL,                       // MSHUTDOWN
    PHP_RINIT(counter),         // RINIT
    NULL,                       // RSHUTDOWN
    PHP_MINFO(counter),         // MINFO
    NO_VERSION_YET,             // Version
    STANDARD_MODULE_PROPERTIES
};
```

Compiling extension

- GNU Autoconf
  - M4 skripta
- PHP_ARG_WITH
  - Dodaje novi argument ./configure skripti
- PHP_NEW_EXTENSION
  - Dodaje ekstenziju u build process PHP-a

Primjer 2

- Primitivni tipovi
- Vraćanje vrijednosti iz funkcija
- Parsiranje argumenata funkcije

Tipovi

Type Specifiers

Spec	Type	Locals
a	array	zval*
A	array or object	zval*
b	boolean	zend_bool
C	class	zend_class_entry*
d	double	double
f	function	zend_fcall_info*, zend_fcall_info_cache*
h	array	HashTable*
H	array or object	HashTable*
l	long	long
L	long (limits out-of-range LONG_MAX/LONG_MIN)	long
o	object	zval*
O	object (of specified zend_class_entry)	zval*, zend_class_entry*
p	string (a valid path)	char*, int
r	resource	char*
s	string	char*, uint
z	mixed	zval*
Z	mixed	zval**

Memory management

- Persistent memory
  - Može koristiti kroz više requestova.
  - pemalloc
  - pecalloc
  - pestrdup
  - pefree
- Non-persistent memory
  - Traje jedan request.
  - emalloc
  - ecalloc
  - estrdup
  - efree

Return value

- Vračanje vrijednosti iz funkcija je jednostavno:
  - RETURN_{ime_tipa}(value)
- Npr:
  - RETURN_STRING("Hello World", 1);

Parametri funkcije

- zend_parse_parameters(
  ZEND_NUM_ARGS() TSRMLS_CC,
  type_string,
  destinations...
)
- 1. parametar copy pasteat :)
- Drugi parametar je string sa tipovima koje prihvaća.
- 3...N. parametari su adrese varijabla gdje spremiti vrijednosti.
- Funkcija vraća SUCCESS ili FAIL

Primjer 3: zval

- Struktura po kojoj su inicijalizirane sve varijable u PHP-u.

Zval

```
typedef struct _zval_struct {
    zvalue_value value;        /* variable value */
    zend_uint refcount__gc;    /* reference counter */
    zend_uchar type;           /* value type */
    zend_uchar is_ref__gc;     /* reference flag */
} zval;
```

```
typedef union _zvalue_value {
    long lval;                 /* long value */
    double dval;               /* double value */
    struct {
        char *val;
        int len;               /* this will always be set for strings */
    } str;                     /* string (always has length) */
    HashTable *ht;             /* an array */
    zend_object_value obj;     /* stores an object store handle, and handlers */
} zvalue_value;
```

Zval: provjera tipa

- Z_TYPE(zval)
  - Vrača tip zval-a
- Konstante IS_{ime_tipa}
- Npr:
  - Z_TYPE(myzval) == IS_STRING

Zval: dohvat vrijednosti

Accessor Macros

Prototype	Accesses	Description
`zend_uchar Z_TYPE(zval zv)`	type	returns the type of the `value`
`long Z_LVAL(zval zv)`	value.lval	
`zend_bool Z_BVAL(zval zv)`	value.lval	cast long `value` to zend_bool
`double Z_DVAL(zval zv)`	value.dval	
`long Z_RESVAL(zval zv)`	value.lval	returns the resource list identifier for `value`
`char* Z_STRVAL(zval zv)`	value.str.val	return the string `value`
`int Z_STRLEN(zval zv)`	value.str.len	return the length of the string `value`
`HashTable* Z_ARRVAL(zval zv)`	value.ht	return the HashTable (array) `value`
`zend_object_value Z_OBJVAL(zval zv)`	value.obj	returns object `value`
`uint Z_OBJ_HANDLE(zval zv)`	value.obj.handle	returns the object handle for object `value`
`zend_object_handlers* Z_OBJ_HT_P(zval zv)`	value.obj.handlers	returns the handler table for object `value`
`zend_class_entry* Z_OBJCE(zval zv)`	value.obj	returns the class entry for object `value`
`HashTable* Z_OBJPROP(zval zv)`	value.obj	returns the properties of object `value`
`HashTable* Z_OBJPROP(zval zv)`	value.obj	returns the properties of object `value`
`HashTable* Z_OBJDEBUG(zval zv)`	value.obj	if an object has the get_debug_info handler set, it is called, else Z_OBJPROP is called

Zval: converzije

Type Conversion

Prototype
`void convert_to_long(zval* pzval)`
`void convert_to_double(zval* pzval)`
`void convert_to_long_base(zval* pzval, int base)`
`void convert_to_null(zval* pzval)`
`void convert_to_boolean(zval* pzval)`
`void convert_to_array(zval* pzval)`
`void convert_to_object(zval* pzval)`
`void convert_object_to_type(zval* pzval, convert_func_t converter)`

Problemi

- Nema dokumentacije
- Nema community content
- Ružan source kod
- Macro hell

Hvala!