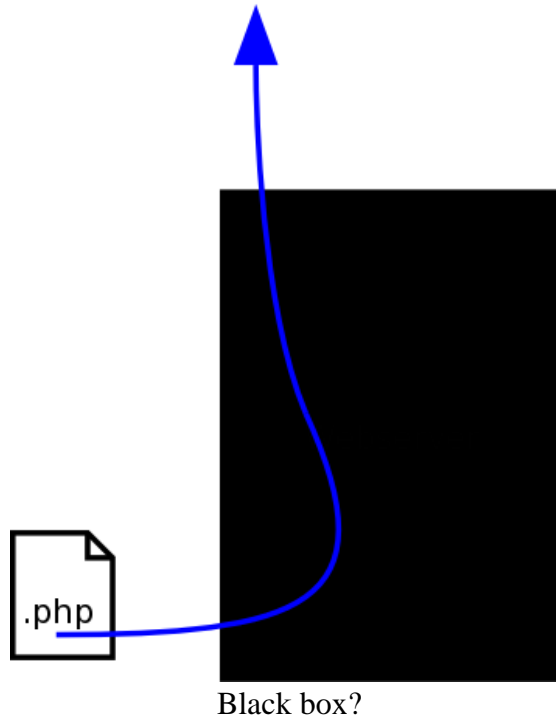


PHP Internals

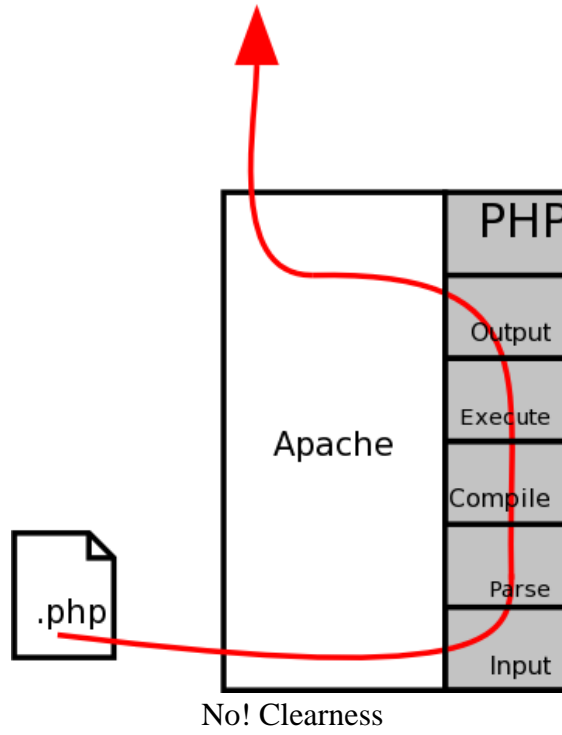
International PHP Conference 2003

November 5, 2003. Frankfurt, Germany

Derick Rethans <derick@php.net>



Black box?



- o PHP registers handlers for:
- o Mime types:
 - o application/x-httpd-php
 - o application/x-httpd-php-source
- o Others:
 - o text/html (Apache xbithack)
 - o php-script (Apache 2)
- o On a request, Apache:
 - o checks if there is a handler for the mime type
 - o opens the file
 - o fills a meta data record
 - o hands PHP a filepointer

- o CGI binary is linked to mime-type:

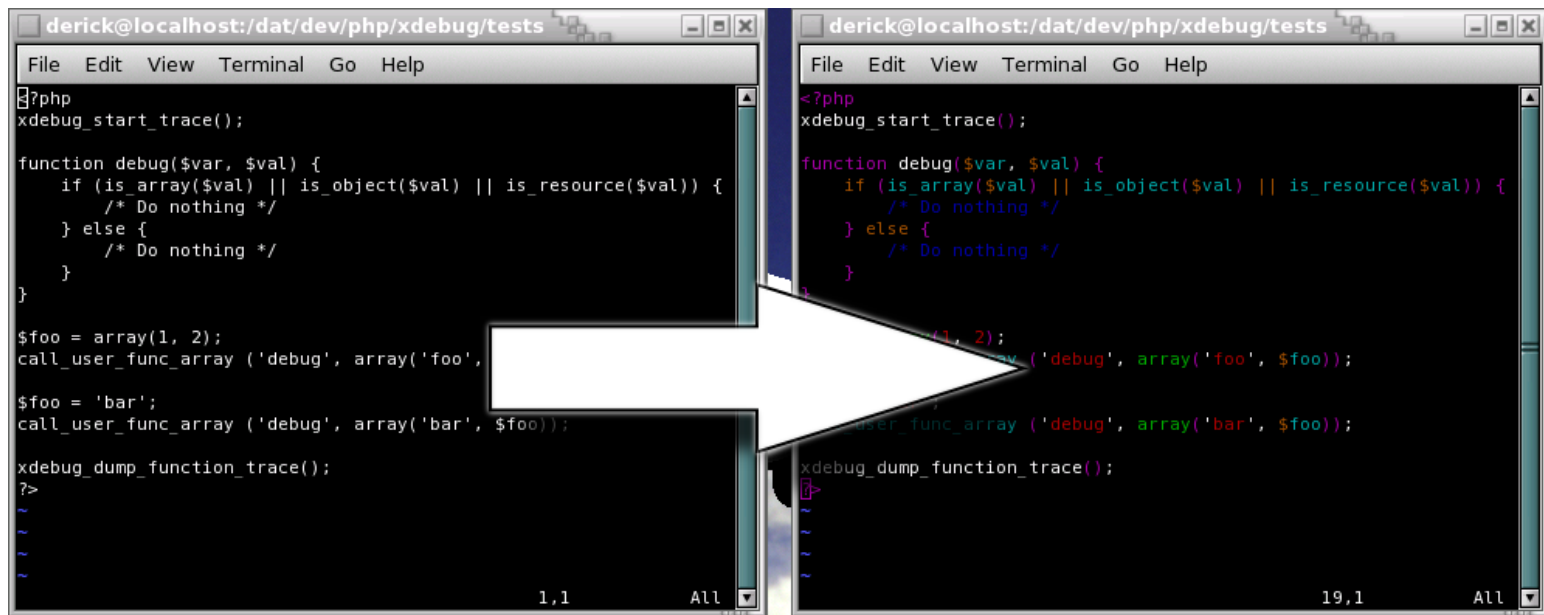
```
ScriptAlias /php/ /usr/local/bin/php-cgi
AddType application/x-httpd-php .php
Action application/x-httpd-php "/php/php.exe"
```

- o On a request, Apache:
- o checks if there is a handler for the mime type
- o fills in needed environment variables
- o executes the CGI processor

- o Lexical analyze script source
- o Divide into logical blocks of characters
- o Give special blocks a meaning
- o flex

Our friend Mr Parse Error:

```
Parse error: parse error,  
unexpected T_CLASS, expecting ',' or ';' in - on line 2
```



Syntax highlighting = Tokenizing

- o The building blocks of a script
- o Strings, variables, keywords
- o Action rules:

```
<ST_IN_SCRIPTING>"::" {  
    return T_PAAMAYIM_NEKUDOTAYIM;  
}  
  
<ST_IN_SCRIPTING>{DNUM}|{EXPONENT_DNUM} {  
    zendlval->value.dval = strtod(yytext, NULL);  
    zendlval->type = IS_DOUBLE;  
    return T_DNUMBER;  
}
```

From Zend/zend_language_parser.y:

```

%left T_INCLUDE T_INCLUDE_ONCE T_EVAL T_REQUIRE T_REQUIRE_ONCE
%left ','
%left T_LOGICAL_OR
%left T_LOGICAL_XOR
%left T_LOGICAL_AND
%right T_PRINT
%left '=' T_PLUS_EQUAL T_MINUS_EQUAL T_MUL_EQUAL T_DIV_EQUAL T_CONCAT_EQUAL
      T_MOD_EQUAL T_AND_EQUAL T_OR_EQUAL T_XOR_EQUAL T_SL_EQUAL T_SR_EQUAL
%left '?' ':'
%left T_BOOLEAN_OR
%left T_BOOLEAN_AND
%left '|'
%left '^'
%left '&'
%nonassoc T_IS_EQUAL T_IS_NOT_EQUAL T_IS_IDENTICAL T_IS_NOT_IDENTICAL
%nonassoc '<' T_IS_SMALLER_OR_EQUAL '>' T_IS_GREATER_OR_EQUAL
%left T_SL T_SR
%left '+' '-' '.'
left '*' '/' ''
%right '!' '~' T_INC T_DEC T_INT_CAST T_DOUBLE_CAST T_STRING_CAST T_ARRAY_CAST
      T_OBJECT_CAST T_BOOL_CAST T_UNSET_CAST '@'
%right '['
%nonassoc T_NEW T_INSTANCEOF
%token T_EXIT
%token T_IF
%left T_ELSEIF
%left T_ELSE
%left T_ENDIF
%token T_LNUMBER
%token T_DNUMBER
%token T_STRING
%token T_STRING_VARNAME
%token T_VARIABLE
%token T_NUM_STRING
%token T_INLINE_HTML
%token T_CHARACTER
%token T_BAD_CHARACTER
%token T_ENCAPSED_AND_WHITESPACE
%token T_CONSTANT_ENCAPSED_STRING
%token T_ECHO
%token T_DO
%token T_WHILE
%token T_ENDWHILE
%token T_FOR
%token T_ENDFOR
%token T_FOREACH
%token T_ENDFOREACH
%token T_DECLARE
%token T_ENDDECLARE
%token T_INSTANCEOF
%token T_AS
%token T_SWITCH
%token T_ENDSWITCH
%token T_CASE
%token T_DEFAULT
%token T_BREAK
%token T_CONTINUE
%token T_FUNCTION
%token T_CONST
%token T_RETURN
%token T_TRY
%token T_CATCH
%token T_THROW
%token T_USE
%token T_GLOBAL
%right T_STATIC T_ABSTRACT T_FINAL T_PRIVATE T_PROTECTED T_PUBLIC
%token T_VAR
%token T_UNSET
%token T_ISSET

```

```
%token T_EMPTY
%token T_CLASS
%token T_INTERFACE
%token T_EXTENDS
%token T_IMPLEMENTES
%token T_OBJECT_OPERATOR
%token T_DOUBLE_ARROW
%token T_LIST
%token T_ARRAY
%token T_CLASS_C
%token T_FUNC_C
%token T_LINE
%token T_FILE
%token T_COMMENT
%token T_DOC_COMMENT
%token T_OPEN_TAG
%token T_OPEN_TAG_WITH_ECHO
%token T_CLOSE_TAG
%token T_WHITESPACE
%token T_START_HEREDOC
%token T_END_HEREDOC
%token T_DOLLAR_OPEN_CURLY_BRACES
%token T_CURLY_OPEN
%token T_PAAMAYIM_NEKUDOTAYIM
}
```

Tokenizer extension:

magic.php:

```
<?php
    function apply_magic(&$a) {
        $a = $a + rand(0,3);
    }
? >
```

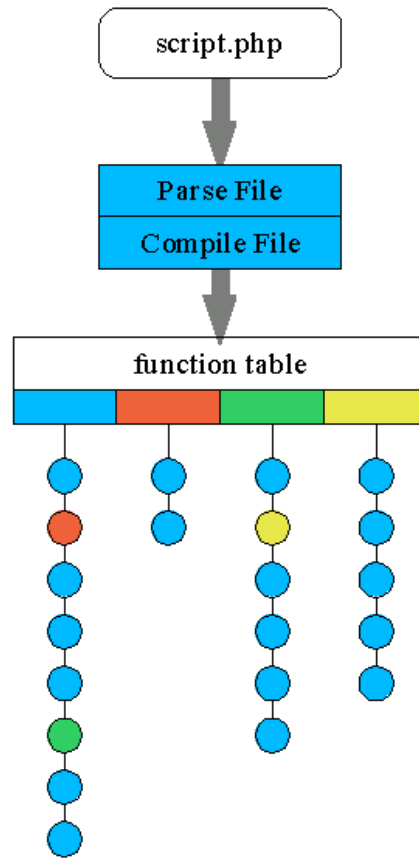
tokenize.php:

```
<?php
    foreach (token_get_all($script) as $token) {
        if (count($token) == 2) {
            printf ("-25s [s]\n", token_name($token[0]), $token[1]);
        } else {
            printf ("-25s [s]\n", "", $token[0]);
        }
    }
? >
```

```
<?php
    function apply_magic(&$a) {
        $a = $a + rand(0,3);
    }
? >
```

T_OPEN_TAG	[<?php\n]
T_WHITESPACE	[]
T_FUNCTION	[function]
T_WHITESPACE	[]
T_STRING	[apply_magic]
	[(]
	[&]
T_VARIABLE	[\$a]
	[)]
T_WHITESPACE	[]
	[{]
T_WHITESPACE	[\n]
T_VARIABLE	[\$a]
T_WHITESPACE	[]
	[=]
T_WHITESPACE	[]
T_VARIABLE	[\$a]
T_WHITESPACE	[]
	[+]
T_WHITESPACE	[]
T_STRING	[rand]
	[(]
T_LNUMBER	[0]
	[,]
T_LNUMBER	[3]
	[)]
	[;]
T_WHITESPACE	[\n]
	[}]
T_WHITESPACE	[\n]
T_CLOSE_TAG	[? >]

- o Interpreting tokens
- o Generating execution units
- o Generating class and function tables



- o Oparrays
- o Compiled code
- o One for every script element

- o Opcode
- o Basic execution unit
- o Two operands
- o One result

fibonacci.php:

```
<?php
    $cache = array();

    function fibonacci($nr) {
        global $cache;

        if (isset($cache[$nr])) {
            return $cache[$nr];
        }
        switch ($nr) {
            case 0:
                die("Invalid Nr\n");
            case 1:
                return 1;
            case 2:
                return 1;
            default:
                $r = fibonacci($nr - 2) + fibonacci($nr - 1);
                $cache[$nr] = $r;
                return $r;
        }
    }

    echo fibonacci($argv[1])."\n";
? >
```



```

compile.php
name: (null)
ops: 11
# op operands code
-----
0 INIT_ARRAY ~1, , $cache = array();
1 FETCH_W $0, 'cache'
2 ASSIGN $0, ~1
3 NOP
4 FETCH_R $3, 'argv'
5 FETCH_DIM_R $4, $3, 1
6 SEND_VAR $4
7 DO_FCALL $5, 'fibonacci', 0
8 CONCAT ~6, $5, '%0A'
9 ECHO ~6
0 RETURN 1

```

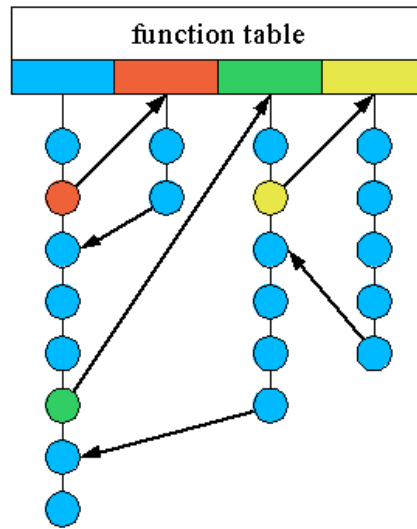
```

fibonacci:
compile.php
name: fibonacci
ops: 58
# op operands code
-----
0 FETCH_W $0, 'nr'
1 RECV $0, 1
2 FETCH_W $1, 'cache'
3 FETCH_W $2, 'cache'
4 ASSIGN_REF $2, $1
5 FETCH_R $4, 'nr'
6 FETCH_IS $3, 'cache',
7 FETCH_DIM_IS $5, $3, $4
8 ISSET_IEMPTY ~6, $5,
9 JMPZ ~6, ->15
0 FETCH_R $8, 'nr'
1 FETCH_R $7, 'cache'
2 FETCH_DIM_R $9, $7, $8
3 RETURN $9
4 JMP ->15
5 FETCH_R $10, 'nr'
6 BOOL ~11,
7 CASE ~11, $10, 0
8 JMPZ ~11, ->21
9 EXIT 'Invalid+Nr%0A',
0 JMP ->23
1 CASE ~11, $10, 1
2 JMPZ ~11, ->26
3 SWITCH_FREE $10,
4 RETURN 1
5 JMP ->28
6 CASE ~11, $10, 2
7 JMPZ ~11, ->31
8 SWITCH_FREE $10,
9 RETURN 1
0 JMP ->33
1 JMP ->55
2 BOOL ~11,
3 INIT_FCALL_BY_NAME 'fibonacci'
4 FETCH_R $13, 'nr'
5 SUB ~14, $13, 2
6 SEND_VAL ~14
7 DO_FCALL_BY_NAME $15, 'fibonacci', 0
8 INIT_FCALL_BY_NAME 'fibonacci'
9 FETCH_R $16, 'nr'
0 SUB ~17, $16, 1
1 SEND_VAL ~17
2 DO_FCALL_BY_NAME $18, 'fibonacci', 0
3 ADD ~19, $15, $18
4 FETCH_W $12, 'r'
5 ASSIGN $12, ~19
6 FETCH_R $22, 'nr'
7 FETCH_R $24, 'r'
function fibonacci($nr) {
    global $cache;
    if (isset($cache[$nr])) {
        return $cache[$nr];
    }
    switch ($nr) {
        case 0:
            die("Invalid Nr\n");
        case 1:
            return 1;
        case 2:
            return 1;
        default:
            $r = fibonacci($nr - 2) + fibonacci($nr - 1);
            $cache[$nr] = $r;
    }
}

```

- o Disassembler: vld
- o Dumps oparray per element:
- o main script
- o function
- o class method

- o Executor executes opcodes
- o 116 or 140 different opcodes
- o Per file execution



test.inc:

```
<?php
    function foo () {
        echo "bar\n";
    }
    echo "inc\n";
? >
```

test.php

```
<?php
    echo "foo\n";
    include "test.inc";
    foo();
    echo "more bar\n";
? >
```

The engine:

- o parses and compiles test.php
- o starts executing test.php
- o parses and compiles test.inc when
the include() statement is encountered
- o starts executing test.inc
- o resumes executing test.php

?

These Slides: <http://pres.derickrethans.nl>

VLD site: <http://www.derickrethans.nl/vld.php>

PHP: <http://www.php.net>

Index

Introduction	2
Introduction (2)	3
Input: Apache module	4
Input: Apache/CGI	5
Parsing	6
Parsing: Tokenize	7
Parsing: Tokens	8
Parsing: Tokens in PHP	9
Parsing: Example (1)	11
Parsing: Example (2)	12
Compiling	13
Compiling: Diagram	14
Compiling: Oparrays	15
Compiling: Example	16
Compiling: Break-down	17
Compiling: Disassembler	19
Executing	20
Executing: Diagram	21
Executing: Example (1)	22
Executing: Example (2)	23
Questions	24
Resources	25