PHP Internals Deep Dive

IPC 2023

Derick Rethans

derick@php.net — @derickr@phpc.social

https://derickrethans.nl/talks/php-ipc23

About Me

Derick Rethans

- I'm European, living in London
- ~~PHP 7.4 **Release Manager**~~
- **PHP Foundation**
- **Xdebug** & PHP's Date/Time support
- I ♥🌍 maps, I ♥🍺 beer, I ♥🥃 whisky
- mastodon: @derickr@phpc.social
- twitter: @derickr

Agenda

- Stages
- Conversion Stages
- Conclusion

Stages

Stages

Parse Script

Create Logical Representation

Create Executable Code

Execute Bytecode

Stages

Parse Script

Convert code into Tokens

Create Logical Representation

Create Executable Code

Execute Bytecode

Stages

Parse Script

Create Logical Representation

Convert Tokens into Abstract Syntax Tree (AST)

Create Executable Code

Execute Bytecode

Stages

Parse Script

Create Logical Representation

Create Executable Code

Convert AST into Opcodes (Bytecode)

Execute Bytecode

Stages

Parse Script

Create Logical Representation

Create Executable Code

Execute Bytecode

Run the Opcodes with the Zend Engine

Collection: Syntax

```php
<?php
collection Articles(int|string => Class)
{
}
?>
```

Collection: Syntax

Example Usage:

```
<?php
collection Articles(string => Article)
{
}

class Article
{
    function __construct(public string $title)
    {
    }
}

$c = new Articles;

// OK:
$c["nine"] = new Article("The Ninth Planet: Pluto");

// Error:
$c["six"] = "Henry VIII";
?>
```

Collection: Syntax

Zend/tests/collection/collection_3.phpt

```
--TESTS--
Collections: Syntax
--FILE--
<?php
collection Articles(string => Article) { }

class Article
{
    function __construct(public string $title) { }
}

$c = new Articles;

// OK:
$c["nine"] = new Article("The Ninth Planet: Pluto");
print_r($c);

// Error:
try {
    $c["six"] = "Henry VIII";
} catch (Error $e) {
    echo get_class($e), ': ', $e->getMessage(), "\n";
}
?>
--EXPECT--
Articles Object
(
    [value] => Array
        (
            [nine] => Article Object
                (
                    [title] => The Ninth Planet: Pluto
                )
        )
)
TypeError: Value type string does not match collection item type Article
```

Parsing

Tokenization

- It's a big state machine starting with `INITIAL`
- States: `INITIAL`, `ST_IN_SCRIPTING`, `ST_DOUBLE_QUOTES`, `ST_NOWDOC`…
- Tokens can change the state:

```
<INITIAL>"<?php"([ \t]|{NEWLINE}) {
    HANDLE_NEWLINE(yytext[yyleng-1]);
    BEGIN(ST_IN_SCRIPTING);
    RETURN_TOKEN(T_OPEN_TAG);
}
```

- No **meaning** is given to these tokens
- PHP comes with a **tokenizer** extension

Tokenization

Example script:

```
<?php
namespace DramIO;
class Whisky
{
    public function __construct( $name )
    {
        $this->name = $name;
    }

    public function drink()
    {
        echo "Drinking {$this->name}\n";
    }
}
?>
```

Tokenization

Example script:

```
<?php
namespace DramIO;
class Whisky
{
    public function __construct($name)
    {
        $this->name = $name;
```

In tokens:

```
T_OPEN_TAG <?php
T_NAMESPACE T_WS T_STRING DramIO ; T_WS
T_CLASS T_WS T_STRING Whisky T_WS
{ T_WS
    T_PUBLIC T_WS T_FUNCTION T_WS T_STRING __construct ( T_VARIABLE $name ) T_WS
    { T_WS
        T_VARIABLE $this T_OBJECT_OPERATOR -> T_STRING name T_WS = T_WS T_VARIABLE $
```

Collection: Add Token

Zend/zend_language_parser.y

```
 %token  T_ENUM            "'enum'"
+%token  T_COLLECTION      "'collection'"
 %token  T_EXTENDS         "'extends'"
```

Zend/zend_language_scanner.l

```
        RETURN_TOKEN_WITH_IDENT(T_ENUM);
 }

+/*
+ * The collection keyword must be followed by whitespace and another identifier.
+ * This avoids the BC break of using collection in classes, namespaces, functions
+ */
+<ST_IN_SCRIPTING>"collection"{WHITESPACE_OR_COMMENTS}("extends"|"implements") {
+       yyless(10);
+       RETURN_TOKEN_WITH_STR(T_STRING, 0);
+}
+<ST_IN_SCRIPTING>"collection"{WHITESPACE_OR_COMMENTS}[a-zA-Z_\x80-\xff] {
+       yyless(10);
+       RETURN_TOKEN_WITH_IDENT(T_COLLECTION);
+}
+
```

Scanner Rules

```
top_statement:
        statement                                   { $$ = $1; }
    |   function_declaration_statement              { $$ = $1; }
    |   class_declaration_statement                 { $$ = $1; }

class_declaration_statement:
        class_modifiers T_CLASS { $<num>$ = CG(zend_lineno); }
        T_STRING extends_from implements_list backup_doc_comment '{' class_stateme
            { $$ = zend_ast_create_decl(ZEND_AST_CLASS, $1, $<num>3, $7, zend_ast_
    |   T_CLASS { $<num>$ = CG(zend_lineno); }
        T_STRING extends_from implements_list backup_doc_comment '{' class_stateme
            { $$ = zend_ast_create_decl(ZEND_AST_CLASS, 0, $<num>2, $6, zend_ast_g

class_modifiers:
        class_modifier                          { $$ = $1; }
    |   class_modifiers class_modifier          { $$ = zend_add_class_modifier($1, $2); }

class_modifier:
        T_ABSTRACT      { $$ = ZEND_ACC_EXPLICIT_ABSTRACT_CLASS; }
    |   T_FINAL         { $$ = ZEND_ACC_FINAL; }
```

Scanner Rules

- Gives Meaning to Tokens
- Constructs AST through Rules:

```
class_declaration_statement:
        class_modifiers T_CLASS { $<num>$ = CG(zend_lineno); }
        T_STRING extends_from implements_list backup_doc_comment '{' class_state
            { $$ = zend_ast_create_decl(ZEND_AST_CLASS, $1, $<num>3, $7, zend_as
```

```
  case 167:
#line 493 "/home/derick/dev/php/php-src.git/Zend/zend_language_parser.y"
    { (yyval.num) = CG(zend_lineno); }
    break;

  case 168:
#line 495 "/home/derick/dev/php/php-src.git/Zend/zend_language_parser.y"
    { (yyval.ast) = zend_ast_create_decl(ZEND_AST_CLASS, (yyvsp[(1) - (10)].num)
    break;
```

Collection: Add Parser Rules

Zend/zend_language_parser.y

```
%type <ast> enum_declaration_statement enum_backing_type enum_case enum_case_expr
…

enum_declaration_statement:
  T_ENUM { $<num>$ = CG(zend_lineno); }
  T_STRING enum_backing_type implements_list backup_doc_comment '{' class_statement_list '}'
    { $$ = zend_ast_create_decl(ZEND_AST_CLASS, ZEND_ACC_ENUM|ZEND_ACC_FINAL, $<num>2, $6, zend_ast_get_str($3), NULL, $5, $8, NULL, $4);
;
```

Add:

```
%type <ast> collection_declaration_statement collection_type
…

collection_declaration_statement:
  T_COLLECTION { $<num>$ = CG(zend_lineno); }
  T_STRING '(' collection_type T_DOUBLE_ARROW collection_type ')' backup_doc_comment '{' class_statement_list '}'
    { $$ = zend_ast_create_decl(ZEND_AST_CLASS, ZEND_ACC_COLLECTION|ZEND_ACC_FINAL, $<num>2, $9, zend_ast_get_str($3), NULL, NULL, $11, $
;

collection_type:
  type_expr { $$ = $1; }
…

    |       enum_declaration_statement          { $$ = $1; }
    |       collection_declaration_statement    { $$ = $1; }
```

Zend/zend_compile.h

```
/* Special class types                                  |     |     |     */
…
#define ZEND_ACC_ENUM                   (1 << 28) /*  X  |     |     |     */
#define ZEND_ACC_COLLECTION             (1 << 30) /*  X  |     |     |     */
```

Abstract Syntax Tree

Abstract Syntax Tree

- An AST describes the structure of the parsed script
- Each node is a language construct
- Nested structures are represented through a tree
- Not all the original information from the original source code is kept
- The **php-ast** extension on Nikita Popov's (nikic) GitHub account can visualize them
- Optimisations are possible by modifications of the tree

Abstract Syntax Tree

Raw output from `ast\parse_code` function:

```
object(ast\Node)#1 (4) {
  ["kind"]=>
  int(133)
  ["flags"]=>
  int(0)
  ["lineno"]=>
  int(1)
  ["children"]=>
  array(2) {
    [0]=>
    object(ast\Node)#2 (4) {
      ["kind"]=>
      int(257)
      ["flags"]=>
      int(0)
      ["lineno"]=>
      int(2)
      ["children"]=>
      array(1) {
        ["name"]=>
        object(ast\Node)#3 (4) {
          ["kind"]=>
          int(2048)
```

Abstract Syntax Tree (formatted)

```
AST_STMT_LIST
    0: AST_CONST
        …
    1: AST_CLASS
        flags: 0
        name: Whisky
        extends: null
        implements: null
        stmts: AST_STMT_LIST
            0: AST_PROP_DECL
                flags: MODIFIER_PRIVATE (1024)
                0: AST_PROP_ELEM
                    name: "name"
                    default: null
            1: AST_METHOD
                flags: MODIFIER_PUBLIC (256)
                name: __construct
                params: AST_PARAM_LIST
                    0: AST_PARAM
                        flags: 0
                        type: null
                        name: "name"
                        default: null
                uses: null
                stmts: AST_STMT_LIST
                    0: AST_ASSIGN
```

Abstract Syntax Tree (formatted)

```php
public function __construct( $name )
{
    $this->name = $name
}
```

```
1: AST_METHOD
    flags: MODIFIER_PUBLIC (256)
    name: __construct
    params: AST_PARAM_LIST
        0: AST_PARAM
            flags: 0
            type: null
            name: "name"
            default: null
    uses: null
    stmts: AST_STMT_LIST
        0: AST_ASSIGN
            var: AST_PROP
                expr: AST_VAR
                    name: "this"
                prop: "name"
            expr: AST_VAR
                name: "name"
    returnType: null
```

Collection: Add AST Code

Zend/zend.h

```
        HashTable *backed_enum_table;

+       uint32_t collection_key_type;
+       zend_type collection_item_type;
```

Zend/zend_compile.c

```
 static void zend_compile_class_decl(znode *result, zend_ast *ast, bool toplevel) /* {{{ */
 {
        zend_ast_decl *decl = (zend_ast_decl *) ast;
        zend_ast *extends_ast = decl->child[0];
        zend_ast *implements_ast = decl->child[1];
        zend_ast *stmt_ast = decl->child[2];
+       zend_ast *collection_key_type_ast = decl->child[3];
+       zend_ast *collection_item_type_ast = decl->child[4];
        zend_ast *enum_backing_type_ast = decl->child[4];
        zend_string *name, *lcname;
        zend_class_entry *ce = zend_arena_alloc(&CG(arena), sizeof(zend_class_entry));
@@ -8003,7 +8039,7 @@ static void zend_compile_class_decl(znode *result, zend_ast *ast, bool toplevel)

        CG(active_class_entry) = ce;

-       if (decl->child[3]) {
+       if (decl->child[3] && !(ce->ce_flags & ZEND_ACC_COLLECTION)) {
                zend_compile_attributes(&ce->attributes, decl->child[3], 0, ZEND_ATTRIBUTE_TARGET_CLASS, 0);
        }

@@ -8019,6 +8055,13 @@ static void zend_compile_class_decl(znode *result, zend_ast *ast, bool toplevel)
                zend_enum_register_props(ce);
        }

+       if (ce->ce_flags & ZEND_ACC_COLLECTION) {
+               zend_compile_collection_key_type(ce, collection_key_type_ast);
+               zend_compile_collection_item_type(ce, collection_item_type_ast);
+       }
+
```

Collection: Add AST Code

Zend/zend_compile.c

```
+static void zend_compile_collection_key_type(zend_class_entry *ce, zend_ast *collection_key_type_ast)
+{
+       ZEND_ASSERT(ce->ce_flags & ZEND_ACC_COLLECTION);
+       zend_type type = zend_compile_typename(collection_key_type_ast, 0);
+       uint32_t type_mask = ZEND_TYPE_PURE_MASK(type);
+       if (ZEND_TYPE_IS_COMPLEX(type) || (type_mask != MAY_BE_LONG && type_mask != MAY_BE_STRING)) {
+               zend_string *type_string = zend_type_to_string(type);
+               zend_error_noreturn(E_COMPILE_ERROR,
+                       "Collection key type must be int or string, %s given",
+                       ZSTR_VAL(type_string));
+       }
+       if (type_mask == MAY_BE_LONG) {
+               ce->collection_key_type = IS_LONG;
+       } else {
+               ZEND_ASSERT(type_mask == MAY_BE_STRING);
+               ce->collection_key_type = IS_STRING;
+       }
+       zend_type_release(type, 0);
+}
```

```
+static void zend_compile_collection_item_type(zend_class_entry *ce, zend_ast *collection_item_type_ast)
+{
+       ZEND_ASSERT(ce->ce_flags & ZEND_ACC_COLLECTION);
+       zend_type type = zend_compile_typename(collection_item_type_ast, 0);
+
+       if (ZEND_TYPE_FULL_MASK(type) & (MAY_BE_VOID|MAY_BE_NEVER|MAY_BE_CALLABLE)) {
+               zend_string *str = zend_type_to_string(type);
+               zend_error_noreturn(E_COMPILE_ERROR,
+                       "Collection item type cannot have type %s", ZSTR_VAL(str));
+       }
+
+       ce->collection_item_type = type;
+}
+
 static void zend_compile_class_decl(znode *result, zend_ast *ast, bool toplevel) /* {{{ */
```

Collection: Clean Up Class Entry

Zend/zend_opcode.c

```
--- Zend/zend_opcode.c
+++ Zend/zend_opcode.c
@@ -430,6 +430,9 @@ ZEND_API void destroy_zend_class(zval *zv)
                        if (ce->backed_enum_table) {
                                zend_hash_release(ce->backed_enum_table);
                        }
+                       if (ZEND_TYPE_IS_SET(ce->collection_item_type)) {
+                               zend_type_release(ce->collection_item_type, 0);
+                       }
                        if (ce->default_properties_table) {
                                zval *p = ce->default_properties_table;
                                zval *end = p + ce->default_properties_count;
```

Collection: Init and Storage

Zend/zend_collection.h

```
#ifndef ZEND_COLLECTION_H
#define ZEND_COLLECTION_H

#include "zend.h"
#include <stdint.h>

BEGIN_EXTERN_C()

extern ZEND_API zend_object_handlers zend_collection_object_handlers;

void zend_collection_register_handlers(zend_class_entry *ce);
void zend_collection_register_props(zend_class_entry *ce);

END_EXTERN_C()
#endif /* ZEND_COLLECTION_H */
```

Collection: Init and Storage

Zend/zend_collection.h

```
#ifndef ZEND_COLLECTION_H
#define ZEND_COLLECTION_H

#include "zend.h"
#include <stdint.h>

BEGIN_EXTERN_C()

extern ZEND_API zend_object_handlers zend_collection_object_handlers;

void zend_collection_register_handlers(zend_class_entry *ce);
void zend_collection_register_props(zend_class_entry *ce);

END_EXTERN_C()
#endif /* ZEND_COLLECTION_H */
```

Zend/zend_collection.c

```
#include "zend.h"
#include "zend_API.h"

void zend_collection_register_handlers(zend_class_entry *ce)
{
	memcpy(&zend_collection_object_handlers, &std_object_handlers, sizeof(zend_object_handlers));
	zend_collection_object_handlers.clone_obj = NULL;
	zend_collection_object_handlers.compare = zend_objects_not_comparable;
	ce->default_object_handlers = &zend_collection_object_handlers;
}

void zend_collection_register_props(zend_class_entry *ce)
{
	zval name_default_value;
	ZVAL_UNDEF(&name_default_value);
	zend_type name_type = ZEND_TYPE_INIT_CODE(IS_ARRAY, 0, 0);
	zend_declare_typed_property(ce, ZSTR_KNOWN(ZEND_STR_VALUE), &name_default_value, ZEND_ACC_PUBLIC | ZEND_ACC_R
	ce->ce_flags |= ZEND_ACC_NO_DYNAMIC_PROPERTIES;
}
```

Collection: Hook-Up Init and Storage

zend_compile.c

```
                zend_compile_collection_key_type(ce, collection_key_type_ast);
                zend_compile_collection_item_type(ce, collection_item_type_ast);
+               zend_collection_register_handlers(ce);
+               zend_collection_register_props(ce);
        }
```

configure.ac

```
@@ -1722,7 +1722,7 @@ PHP_ADD_SOURCES(Zend, \
     zend_closures.c zend_weakrefs.c zend_float.c zend_string.c zend_signal.c zend_generators.c \
     zend_virtual_cwd.c zend_ast.c zend_objects.c zend_object_handlers.c zend_objects_API.c \
     zend_default_classes.c zend_inheritance.c zend_smart_str.c zend_cpuinfo.c zend_gdb.c \
-    zend_observer.c zend_system_id.c zend_enum.c zend_fibers.c zend_atomic.c \
+    zend_observer.c zend_system_id.c zend_enum.c zend_collection.c zend_fibers.c zend_atomic.c \
     zend_max_execution_timer.c \
```

Collection: Hook-Up Init and Storage

zend_compile.c

```
                zend_compile_collection_key_type(ce, collection_key_type_ast);
                zend_compile_collection_item_type(ce, collection_item_type_ast);
+               zend_collection_register_handlers(ce);
+               zend_collection_register_props(ce);
        }
```

configure.ac

```
@@ -1722,7 +1722,7 @@ PHP_ADD_SOURCES(Zend, \
     zend_closures.c zend_weakrefs.c zend_float.c zend_string.c zend_signal.c zend_generators.c \
     zend_virtual_cwd.c zend_ast.c zend_objects.c zend_object_handlers.c zend_objects_API.c \
     zend_default_classes.c zend_inheritance.c zend_smart_str.c zend_cpuinfo.c zend_gdb.c \
-    zend_observer.c zend_system_id.c zend_enum.c zend_fibers.c zend_atomic.c \
+    zend_observer.c zend_system_id.c zend_enum.c zend_collection.c zend_fibers.c zend_atomic.c \
     zend_max_execution_timer.c \
```

win32/build/config.w32

```
@@ -240,7 +240,7 @@ ADD_SOURCES("Zend", "zend_language_parser.c zend_language_scanner.c \
        zend_default_classes.c zend_execute.c zend_strtod.c zend_gc.c zend_closures.c zend_weakrefs.c \
        zend_float.c zend_string.c zend_generators.c zend_virtual_cwd.c zend_ast.c \
        zend_inheritance.c zend_smart_str.c zend_cpuinfo.c zend_observer.c zend_system_id.c \
-       zend_enum.c zend_fibers.c zend_atomic.c");
+       zend_enum.c zend_collection.c zend_fibers.c zend_atomic.c");
```

Collection: Add Interface

Zend/zend_collection.stub.php

```
<?php
/** @generate-class-entries */
interface Collection
{
}
```

Zend/zend_collection.h

```
 extern ZEND_API zend_object_handlers zend_collection_object_handlers;
+extern ZEND_API zend_class_entry *zend_ce_collection;
+
+void zend_register_collection_ce(void);
+void zend_collection_add_interfaces(zend_class_entry *ce);
```

Zend/zend_collection.c

```
+#include "zend_collection_arginfo.h"
+#include "zend_execute.h"

+ZEND_API zend_class_entry *zend_ce_collection;
 ZEND_API zend_object_handlers zend_collection_object_handlers;
```

Collection: Add Interface Handlers

Zend/zend_collection.c

```
static int zend_implement_collection(zend_class_entry *interface, zend_class_entry *class_type)
{
	if (class_type->ce_flags & ZEND_ACC_COLLECTION) {
		return SUCCESS;
	}

	zend_error_noreturn(E_ERROR, "Non-collection class %s cannot implement interface %s",
		ZSTR_VAL(class_type->name),
		ZSTR_VAL(interface->name));

	return FAILURE;
}
```

Collection: Add Interface Handlers

Zend/zend_collection.c

```
static int zend_implement_collection(zend_class_entry *interface, zend_class_entry *class_type)
{
	if (class_type->ce_flags & ZEND_ACC_COLLECTION) {
		return SUCCESS;
	}

	zend_error_noreturn(E_ERROR, "Non-collection class %s cannot implement interface %s",
		ZSTR_VAL(class_type->name),
		ZSTR_VAL(interface->name));

	return FAILURE;
}
```

```
void zend_register_collection_ce(void)
{
	zend_ce_collection = register_class_Collection();
	zend_ce_collection->interface_gets_implemented = zend_implement_collection;

	memcpy(&zend_collection_object_handlers, &std_object_handlers, sizeof(zend_object_handlers));
	zend_collection_object_handlers.clone_obj = NULL;
	zend_collection_object_handlers.compare = zend_objects_not_comparable;
}
```

Collection: Add Interface Handlers

Zend/zend_collection.c

```
static int zend_implement_collection(zend_class_entry *interface, zend_class_entry *class_type)
{
	if (class_type->ce_flags & ZEND_ACC_COLLECTION) {
		return SUCCESS;
	}

	zend_error_noreturn(E_ERROR, "Non-collection class %s cannot implement interface %s",
		ZSTR_VAL(class_type->name),
		ZSTR_VAL(interface->name));

	return FAILURE;
}
```

```
void zend_register_collection_ce(void)
{
	zend_ce_collection = register_class_Collection();
	zend_ce_collection->interface_gets_implemented = zend_implement_collection;

	memcpy(&zend_collection_object_handlers, &std_object_handlers, sizeof(zend_object_handlers));
	zend_collection_object_handlers.clone_obj = NULL;
	zend_collection_object_handlers.compare = zend_objects_not_comparable;
}
```

```
void zend_collection_add_interfaces(zend_class_entry *ce)
{
	uint32_t num_interfaces_before = ce->num_interfaces;

	ce->num_interfaces++;

	ZEND_ASSERT(!(ce->ce_flags & ZEND_ACC_RESOLVED_INTERFACES));

	ce->interface_names = erealloc(ce->interface_names, sizeof(zend_class_name) * ce->num_interfaces);

	ce->interface_names[num_interfaces_before].name = zend_string_copy(zend_ce_collection->name);
	ce->interface_names[num_interfaces_before].lc_name = ZSTR_INIT_LITERAL("collection", 0);

	ce->default_object_handlers = &zend_collection_object_handlers;
}
```

Collection: Add Interface Handlers

Zend/zend_compile.c

```
@@ -8059,6 +8059,7 @@ static void zend_compile_class_decl(znode *result, zend_ast *ast, bool toplevel)
        if (ce->ce_flags & ZEND_ACC_COLLECTION) {
                zend_compile_collection_key_type(ce, collection_key_type_ast);
                zend_compile_collection_item_type(ce, collection_item_type_ast);
+               zend_collection_add_interfaces(ce);
                zend_collection_register_handlers(ce);
                zend_collection_register_props(ce);
        }
```

Zend/zend_default_classes.c

```
@@ -40,4 +40,5 @@ ZEND_API void zend_register_default_classes(void)
        zend_register_attribute_ce();
        zend_register_enum_ce();
        zend_register_fiber_ce();
+       zend_register_collection_ce();
 }
```

Collection: Add Write Handler

Zend/zend_object_handlers.c

```
@@ -26,6 +26,7 @@
 #include "zend_objects_API.h"
 #include "zend_object_handlers.h"
 #include "zend_interfaces.h"
+#include "zend_collection.h"
 #include "zend_exceptions.h"
…

@@ -1042,6 +1045,8 @@ ZEND_API void zend_std_write_dimension(zend_object *object, zval *offset, zval *
                zend_call_known_instance_method_with_2_params(funcs->zf_offsetset, object, NULL, &tmp_offset, value);
                OBJ_RELEASE(object);
                zval_ptr_dtor(&tmp_offset);
+       } else if (zend_class_implements_interface(ce, zend_ce_collection)) {
+               zend_collection_add_item(object, offset, value);
        } else {
                zend_bad_array_access(ce);
        }
```

Zend/zend_collection.h

```
@@ -34,6 +34,11 @@ void zend_collection_add_interfaces(zend_class_entry *ce);
 void zend_collection_register_handlers(zend_class_entry *ce);
 void zend_collection_register_props(zend_class_entry *ce);

+void zend_collection_add_item(zend_object *object, zval *offset, zval *value);
```

Collection: Add Write Handler Implementation [1]

Zend/zend_collection.c

```
void zend_collection_add_item(zend_object *object, zval *offset, zval *value)
{
	zend_class_entry *ce = object->ce;
	zval rv;
	zval *value_prop;
	ZEND_ASSERT(ce->ce_flags & ZEND_ACC_COLLECTION);

	if (!zend_check_type(&ce->collection_item_type, value, NULL, ce, 0, false)) {
		zend_string *type_str = zend_type_to_string(ce->collection_item_type);
		zend_type_error(
			"Value type %s does not match collection item type %s",
			zend_zval_type_name(value), ZSTR_VAL(type_str));
		zend_string_release(type_str);
		return;
	}

	if (!offset && ce->collection_key_type == IS_LONG) {
		create_array_if_needed(ce, object);
		value_prop = zend_read_property_ex(ce, object, ZSTR_KNOWN(ZEND_STR_VALUE), true, &rv);
		Z_ADDREF_P(value);
		add_next_index_zval(value_prop, value);
	} else if (offset && ce->collection_key_type == IS_LONG && Z_TYPE_P(offset) == IS_LONG) {
		create_array_if_needed(ce, object);
		value_prop = zend_read_property_ex(ce, object, ZSTR_KNOWN(ZEND_STR_VALUE), true, &rv);
		Z_ADDREF_P(value);
		add_index_zval(value_prop, Z_LVAL_P(offset), value);
	} else if (offset && ce->collection_key_type == IS_STRING && Z_TYPE_P(offset) == IS_STRING) {
		create_array_if_needed(ce, object);
		value_prop = zend_read_property_ex(ce, object, ZSTR_KNOWN(ZEND_STR_VALUE), true, &rv);
		Z_ADDREF_P(value);
		add_assoc_zval_ex(value_prop, Z_STRVAL_P(offset), Z_STRLEN_P(offset), value);
	} else {
		zend_type_error(
			"Key type %s of element does not match collection key type %s",
			offset ? zend_zval_type_name(offset) : zend_get_type_by_const(IS_NULL),
			zend_get_type_by_const(ce->collection_key_type));
	}
}
```

Collection: Add Write Handler Implementation [2]

Zend/zend_collection.c

```
static void create_array_if_needed(zend_class_entry *ce, zend_object *object)
{
	zval *value_prop = zend_read_property_ex(ce, object, ZSTR_KNOWN(ZEND_STR_VALUE), true, NULL);

	if (Z_TYPE_P(value_prop) == IS_ARRAY) {
		return;
	}

	zval new_array;
	array_init(&new_array);
	zend_update_property_ex(ce, object, ZSTR_KNOWN(ZEND_STR_VALUE), &new_array);
	zval_ptr_dtor(&new_array);
}
```

Collection: Make Check Type Public

Zend/zend_execute.h

```
@@ -89,6 +89,7 @@ ZEND_API ZEND_COLD void ZEND_FASTCALL zend_invalid_class_constant_type_error(uin
 ZEND_API ZEND_COLD void ZEND_FASTCALL zend_object_released_while_assigning_to_property_error(const zend_property_inf

 ZEND_API bool zend_verify_scalar_type_hint(uint32_t type_mask, zval *arg, bool strict, bool is_internal_arg);
+ZEND_API bool zend_check_type(zend_type *type, zval *arg, void **cache_slot, zend_class_entry *scope, bool is_return_type, bool is_inter
 ZEND_API ZEND_COLD void zend_verify_arg_error(
                 const zend_function *zf, const zend_arg_info *arg_info, uint32_t arg_num, zval *value);
 ZEND_API ZEND_COLD void zend_verify_return_error(
```

Zend/zend_execute.c

```
@@ -1182,7 +1182,7 @@ static zend_always_inline bool zend_check_type_slow(
         * because this case is already checked at compile-time. */
 }

-static zend_always_inline bool zend_check_type(
+zend_always_inline bool zend_check_type(
                 zend_type *type, zval *arg, void **cache_slot, zend_class_entry *scope,
                 bool is_return_type, bool is_internal)
 {
```

Collection: Add Read Handler [1]

Zend/zend_object_handlers.c

```
@@ -1019,6 +1020,8 @@ ZEND_API zval *zend_std_read_dimension(zend_object *object, zval *offset, int ty
                         return NULL;
                 }
                 return rv;
+       } else if (zend_class_implements_interface(ce, zend_ce_collection)) {
+               return zend_collection_read_item(object, offset);
        } else {
                 zend_bad_array_access(ce);
                 return NULL;
```

Zend/zend_collection.h

```
 void zend_collection_add_item(zend_object *object, zval *offset, zval *value);
+zval *zend_collection_read_item(zend_object *object, zval *offset);
```

Collection: Add Read Handler [2]

Zend/zend_collection.c

```
static int key_type_allowed(zend_class_entry *ce, zval *offset)
{
	ZEND_ASSERT(ce->ce_flags & ZEND_ACC_COLLECTION);

	if (ce->collection_key_type != Z_TYPE_P(offset)) {
		zend_type_error(
			"Key type %s of element does not match collection key type %s",
			offset ? zend_zval_type_name(offset) : zend_get_type_by_const(IS_NULL),
			zend_get_type_by_const(ce->collection_key_type)
		);
		return false;
	}

	return true;
}

zval *zend_collection_read_item(zend_object *object, zval *offset)
{
	zval rv;
	zval *value_prop, *value;
	zend_class_entry *ce = object->ce;

	if (!key_type_allowed(ce, offset)) {
		return NULL;
	}

	value_prop = zend_read_property_ex(ce, object, ZSTR_KNOWN(ZEND_STR_VALUE), true, &rv);

	if (Z_TYPE_P(offset) == IS_STRING) {
		value = zend_hash_find(HASH_OF(value_prop), Z_STR_P(offset));
	} else {
		value = zend_hash_index_find(HASH_OF(value_prop), Z_LVAL_P(offset));
	}

	return value;
}
```

Collection: Add Has Handler

Zend/zend_object_handlers.c

```
@@ -1068,6 +1073,8 @@ ZEND_API int zend_std_has_dimension(zend_object *object, zval *offset, int check
                }
                OBJ_RELEASE(object);
                zval_ptr_dtor(&tmp_offset);
+       } else if (zend_class_implements_interface(ce, zend_ce_collection)) {
+               return zend_collection_has_item(object, offset);
        } else {
                zend_bad_array_access(ce);
                return 0;
```

Zend/zend_collection.h

```
 void zend_collection_add_item(zend_object *object, zval *offset, zval *value);
+int zend_collection_has_item(zend_object *object, zval *offset);
```

Collection: Add Has Handler

Zend/zend_object_handlers.c

```
@@ -1068,6 +1073,8 @@ ZEND_API int zend_std_has_dimension(zend_object *object, zval *offset, int check
                }
                OBJ_RELEASE(object);
                zval_ptr_dtor(&tmp_offset);
+       } else if (zend_class_implements_interface(ce, zend_ce_collection)) {
+               return zend_collection_has_item(object, offset);
        } else {
                zend_bad_array_access(ce);
                return 0;
```

Zend/zend_collection.h

```
 void zend_collection_add_item(zend_object *object, zval *offset, zval *value);
+int zend_collection_has_item(zend_object *object, zval *offset);
```

Zend/zend_collection.c

```
int zend_collection_has_item(zend_object *object, zval *offset)
{
       zval rv;
       zval *value_prop;
       zend_class_entry *ce = object->ce;

       if (!key_type_allowed(ce, offset)) {
               return false;
       }

       value_prop = zend_read_property_ex(ce, object, ZSTR_KNOWN(ZEND_STR_VALUE), true, &rv);
       if (Z_TYPE_P(offset) == IS_STRING) {
               return zend_hash_find(HASH_OF(value_prop), Z_STR_P(offset)) != NULL;
       } else {
               return zend_hash_index_find(HASH_OF(value_prop), Z_LVAL_P(offset)) != NULL;
       }

       return false;
}
```

Collection: Add Unset Handler

Zend/zend_object_handlers.c

```
@@ -1251,6 +1258,8 @@ ZEND_API void zend_std_unset_dimension(zend_object *object, zval *offset) /* {{{
                zend_call_known_instance_method_with_1_params(funcs->zf_offsetunset, object, NULL, &tmp_offset);
                OBJ_RELEASE(object);
                zval_ptr_dtor(&tmp_offset);
+       } else if (zend_class_implements_interface(ce, zend_ce_collection)) {
+               zend_collection_unset_item(object, offset);
        } else {
                zend_bad_array_access(ce);
        }
```

Zend/zend_collection.h

```
 zval *zend_collection_read_item(zend_object *object, zval *offset);
+void zend_collection_unset_item(zend_object *object, zval *offset);
```

Collection: Add Unset Handler

Zend/zend_object_handlers.c

```
@@ -1251,6 +1258,8 @@ ZEND_API void zend_std_unset_dimension(zend_object *object, zval *offset) /* {{{
                zend_call_known_instance_method_with_1_params(funcs->zf_offsetunset, object, NULL, &tmp_offset);
                OBJ_RELEASE(object);
                zval_ptr_dtor(&tmp_offset);
+       } else if (zend_class_implements_interface(ce, zend_ce_collection)) {
+               zend_collection_unset_item(object, offset);
        } else {
                zend_bad_array_access(ce);
        }
```

Zend/zend_collection.h

```
 zval *zend_collection_read_item(zend_object *object, zval *offset);
+void zend_collection_unset_item(zend_object *object, zval *offset);
```

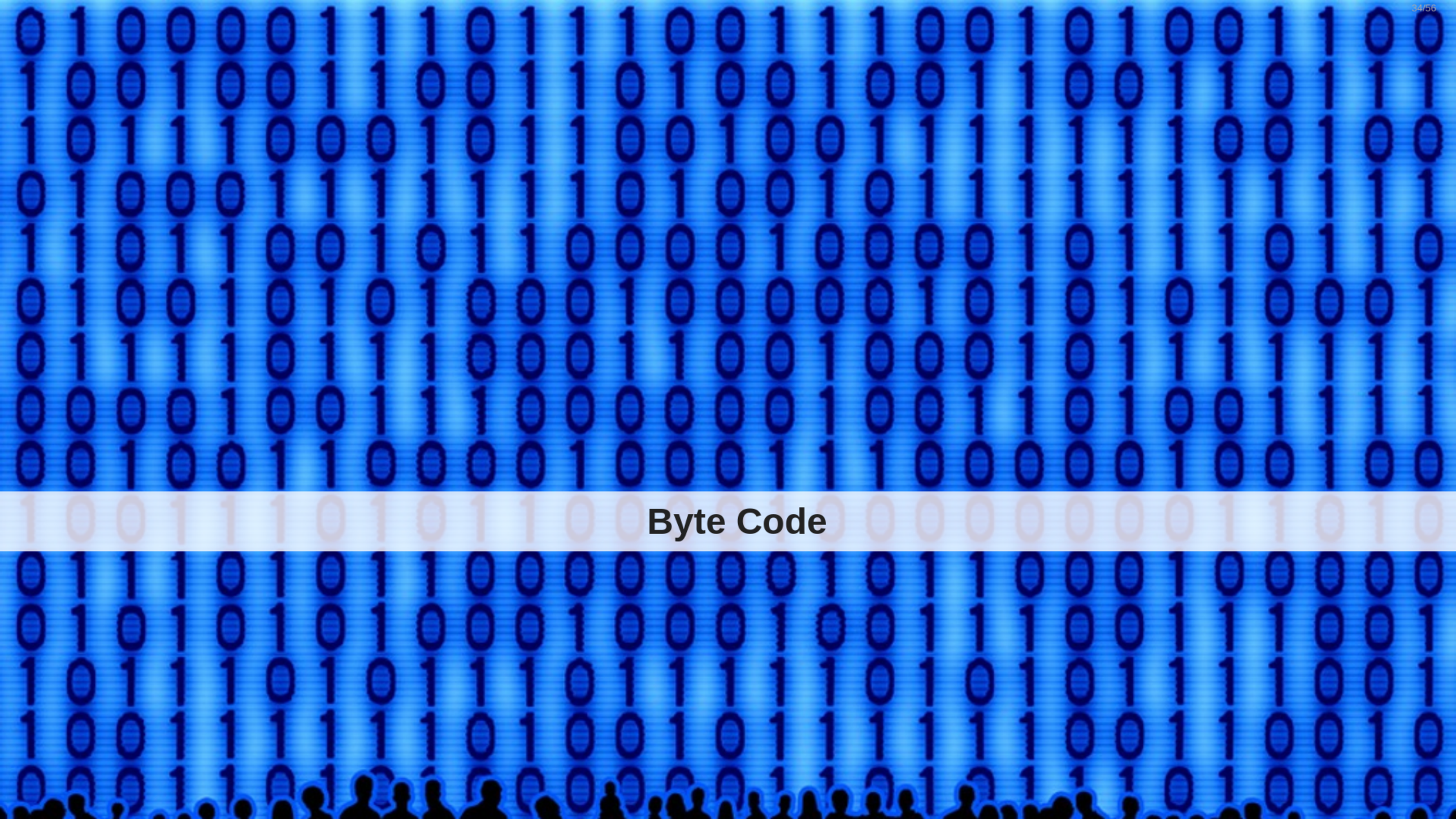
Zend/zend_collection.c

```
void zend_collection_unset_item(zend_object *object, zval *offset)
{
       zval rv;
       zval *value_prop;
       zend_class_entry *ce = object->ce;

       if (!key_type_allowed(ce, offset)) {
               return;
       }

       value_prop = zend_read_property_ex(ce, object, ZSTR_KNOWN(ZEND_STR_VALUE), true, &rv);

       if (Z_TYPE_P(offset) == IS_STRING) {
               zend_hash_del(HASH_OF(value_prop), Z_STR_P(offset));
       } else {
               zend_hash_index_del(HASH_OF(value_prop), Z_LVAL_P(offset));
       }
}
```



Byte Code

Byte Code

- In PHP also called Opcodes
- Each function, method, or main body is represented by an Oparray
- Each Oparray contains Opcodes with instructions for the Zend Engine
- The Zend Engine executes each opcode in turn
- Very similar to Assembler instructions
- You can visualize them with the PECL **vld** extension

AST is converted to bytecode

```
1: AST_METHOD
    flags: MODIFIER_PUBLIC (256)
    name: __construct
    params: AST_PARAM_LIST
        0: AST_PARAM
            name: "name"
    stmts: AST_STMT_LIST
        0: AST_ASSIGN
            var: AST_PROP
                expr: AST_VAR
                    name: "this"
                prop: "name"
            expr: AST_VAR
                name: "name"
```

```
compiled vars:  !0 = $name
line     #* E I O op                                   fetch          ext  return  operands
-------------------------------------------------------------------------------------------
   8     0  E >   EXT_NOP
         1        RECV                                                    !0
  10     2        EXT_STMT
         3        ASSIGN_OBJ                                                   'name'
         4        OP_DATA                                                      !0
  11     5        EXT_STMT
         6      > RETURN                                                       null
```

FOR

```php
<?php
for ( $i = 0; $i < 42; ++$i ) {
    echo $i;
}
```

```
        init: AST_EXPR_LIST
            0: AST_ASSIGN
                var: AST_VAR
                    name: "i"
                expr: 0
        cond: AST_EXPR_LIST
            0: AST_BINARY_OP
                flags: BINARY_IS_SMALLER (19)
                left: AST_VAR
                    name: "i"
                right: 42
        loop: AST_EXPR_LIST
            0: AST_PRE_INC
                var: AST_VAR
                    name: "i"
        stmts: …
```

```
compiled vars:  !0 = $i
line     #* E I O op                           fetch          ext  return  operands
   2     0  E >   EXT_STMT
         1        ASSIGN                                                 !0, 0
         2      > JMP                                                    ->6
   4     3    >   EXT_STMT
         4        ECHO                                                   !0
   2     5        PRE_INC                                                !0
         6    >   IS_SMALLER                                  ~3        !0, 42
         7        EXT_STMT
         8      > JMPNZ                                                  ~3, ->3
         9    > > RETURN                                                 1
```

FOR (rewritten)

```
for ( $i = 0; $i < 42; ++$i ) {
    echo $i;
}
```

```
        $i = 0;
        goto COND;
STMT:
        echo $i;
        ++$i;
COND:
        $_ = $i < 42;
        if ( ! $_ ) {
                goto STMT;
        }
```

```
compiled vars:  !0 = $i
line     #* E I O op                           fetch          ext  return  operands
   2     0  E >   EXT_STMT
         1        ASSIGN                                                 !0, 0
         2      > JMP                                                    ->6
   4     3    >   EXT_STMT
         4        ECHO                                                   !0
   2     5        PRE_INC                                                !0
         6    >   IS_SMALLER                                  ~3        !0, 42
         7        EXT_STMT
         8      > JMPNZ                                                  ~3, ->3
         9    > > RETURN                                                 1
```

DO .. WHILE

```php
<?php
$i = 10;
do {
    echo $i;
} while ( --$i );
```

```
compiled vars:  !0 = $i
line     #* E I O op                                fetch  return  operands
   2     0  E >   EXT_STMT
         1        ASSIGN                                       !0, 10
   3     2        NOP
   4     3    >   EXT_STMT
         4        ECHO                                         !0
   5     5        PRE_DEC                            $2        !0
         6      > JMPNZ                                        $2, ->3
         7    > > RETURN                                       1
```

FOREACH

```
$a = [ 2.7, 4.9 ];
foreach ( $a as $key => $value ) {
    echo $key, $value;
}
```

```
compiled vars:  !0 = $a, !1 = $value, !2 = $key
line     #* E I O op                           fetch  return  operands
   2      0  E >   EXT_STMT
          1        ASSIGN                                     !0, <array>
   3      2        EXT_STMT
          3      > FE_RESET_R                        $4       !0, ->11
          4    > > FE_FETCH_R                        ~5       $4, !1, ->11
          5    >   ASSIGN                                     !2, ~5
   4      6        EXT_STMT
          7        ECHO                                       !2
          8        EXT_STMT
          9        ECHO                                       !1
         10      > JMP                                       ->4
         11    >   FE_FREE                                   $4
         12      > RETURN                                    1
```

Complex Loops

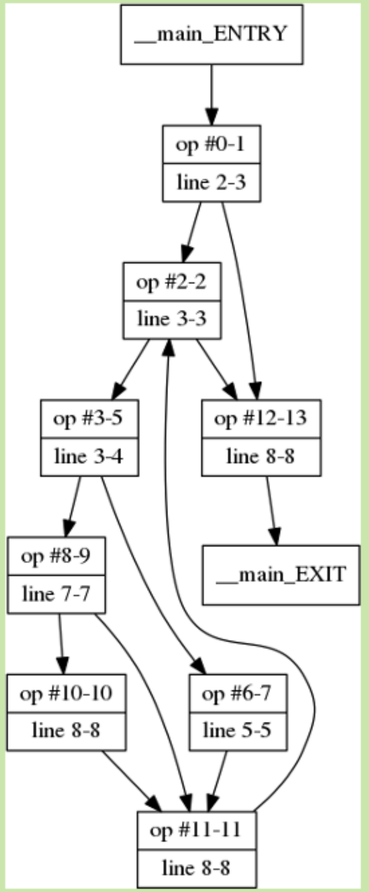
```
<?php
$a = [ 2.7, 4.9 ];
foreach ( $a as $key => $value ) {
        if ( $value < 3 ) {
                echo $key;
        } else {
                if ( $key > 0 ) {
                        echo $value;
                }
        }
}
```

```
line     #* E I O op                           fetch          ext  return  operands
   2      0  E >   ASSIGN                                               !0, <array>
   3      1      > FE_RESET_R                                    $4      !0, ->12
          2    > > FE_FETCH_R                                    ~5      $4, !1, ->12
          3    >   ASSIGN                                               !2, ~5
   4      4        IS_SMALLER                                    ~7      !1, 3
          5      > JMPZ                                                 ~7, ->8
   5      6    >   ECHO                                                 !2
          7      > JMP                                                  ->11
   7      8    >   IS_SMALLER                                    ~8      0, !2
          9      > JMPZ                                                 ~8, ->11
   8     10    >   ECHO                                                 !1
         11    > > JMP                                                  ->2
         12    >   FE_FREE                                             $4
         13      > RETURN                                               1
```

Complex Loops

```php
<?php
$a = [ 2.7, 4.9 ];
foreach ( $a as $key => $value ) {
        if ( $value < 3 ) {
                echo $key;
        } else {
                if ( $key > 0 ) {
                        echo $value;
                }
        }
}
```

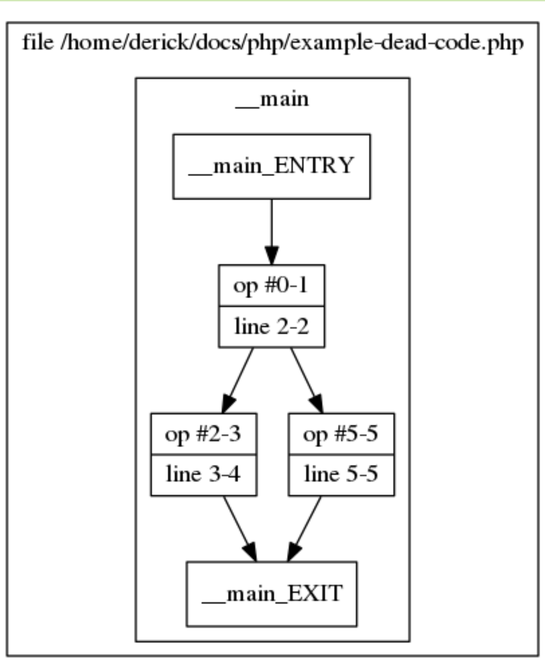
```
line     #* E I O op              return  operands
   2      0  E >   ASSIGN                  !0, <array>
   3      1      > FE_RESET_R     $4       !0, ->12
          2    > > FE_FETCH_R     ~5       $4, !1, ->12
          3    >   ASSIGN                  !2, ~5
   4      4        IS_SMALLER     ~7       !1, 3
          5      > JMPZ                    ~7, ->8
   5      6    >   ECHO                    !2
          7      > JMP                     ->11
   7      8    >   IS_SMALLER     ~8       0, !2
          9      > JMPZ                    ~8, ->11
   8     10    >   ECHO                    !1
         11    > > JMP                     ->2
         12    >   FE_FREE                 $4
         13      > RETURN                  1
```



Dead Code

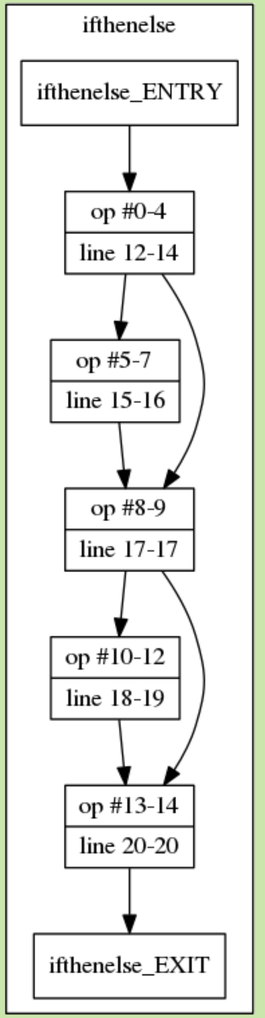
```
<?php
if ( $a < 42 ) {
        echo "fourty";
        return;
        echo "two";
}
```

```
line    #* E I O op                  return  operands
   2    0  E >   IS_SMALLER          ~1      !0, 42
        1       > JMPZ                       ~1, ->5
   3    2     >   ECHO                       'fourty'
   4    3       > RETURN                     null
   5    4*        ECHO                       'two'
        5     > > RETURN                     1
```



- Follow all branches
- Find unreachable opcodes

```
 1 <?php
…
12 function ifthenelse( $a, $b )
13 {
14     if ($a) {
15         echo "A HIT\n";
16     }
17     if ($b) {
18         echo "B HIT\n";
19     }
20 }
21
22 loopy(0);
23 ifthenelse( true, false );
24 ifthenelse( false, true );
25 ?>
```



Xdebug code coverage

```php
<?php
function ifthenelse( $a, $b )
{
    if ($a) {
        echo "A HIT\n";
    }
    if ($b) {
        echo "B HIT\n";
    }
}
```

```php
<?php
require 'vendor/autoload.php';
use SebastianBergmann\CodeCoverage\Filter;
use SebastianBergmann\CodeCoverage\Driver\Selector;
use SebastianBergmann\CodeCoverage\CodeCoverage;
use SebastianBergmann\CodeCoverage\Report\Html\Facade as HtmlReport;

include 'test.php';

$filter = new Filter;
$filter->includeFiles( [ '/test.php' ] );

$coverage = new CodeCoverage(
    (new Selector)->forLineAndPathCoverage($filter),
    $filter
);

$coverage->start('coverage1'); ifthenelse( true, false ); $coverage->stop();
$coverage->start('coverage2'); ifthenelse( false, true ); $coverage->stop();

(new HtmlReport)->process($coverage, '/tmp/code-coverage-report');
```

Xdebug code coverage output

/home/derick/dev/php/xdebug-demos/deep-dive-cov / (Dashboard)

	Code Coverage														
	Lines			Branches			Paths			Functions and Methods			Classes and Traits		
Total		0.17%	11 / 6655		0.27%	7 / 2606		0.01%	2 / 27940		0.33%	2 / 604		0.00%	0 / 113
vendor		0.05%	3 / 6647		0.08%	2 / 2601		0.00%	0 / 27936		0.17%	1 / 603		0.00%	0 / 113
run-cov.php [line] [branch] [path]		100.00%	4 / 4		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0
test.php [line] [branch] [path]		100.00%	4 / 4		100.00%	5 / 5		50.00%	2 / 4		100.00%	1 / 1		n/a	0 / 0

Legend

Low: 0% to 50% **Medium**: 50% to 90% **High**: 90% to 100%

Generated by php-code-coverage 10.1.7 using PHP 8.2.2-dev at Mon Oct 16 9:29:09 UTC 2023.

Xdebug branch coverage

```
<?php
include 'dump-branch-coverage.inc';
include 'test.php';

xdebug_start_code_coverage(
        XDEBUG_CC_UNUSED |
        XDEBUG_CC_DEAD_CODE |
        XDEBUG_CC_BRANCH_CHECK
);

ifthenelse( true, false );
ifthenelse( false, true );

xdebug_stop_code_coverage(false);

$c = xdebug_get_code_coverage();
dump_branch_coverage($c);
?>
```

Xdebug branch coverage output

```
A HIT
ifthenelse
- branches
  - 00; OP: 00-04; line: 02-04 HIT; out1: 05 HIT; out2: 08 HIT
  - 05; OP: 05-07; line: 05-06 HIT; out1: 08 HIT
  - 08; OP: 08-09; line: 07-07 HIT; out1: 10 HIT; out2: 13 HIT
  - 10; OP: 10-12; line: 08-09 HIT; out1: 13 HIT
  - 13; OP: 13-14; line: 10-10 HIT; out1: EX  X
- paths
  - 0 5 8 10 13:  X
  - 0 5 8 13: HIT
  - 0 8 10 13: HIT
  - 0 8 13:  X
```

Xdebug branch coverage display

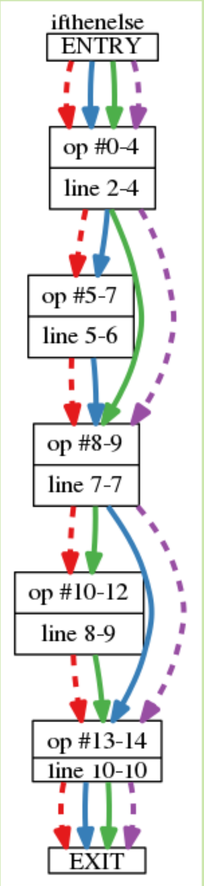
test.php

```
 1 <?php
 2 function ifthenelse( $a, $b )
 3 {
 4     if ($a) {
 5         echo "A HIT\n";
 6     }
 7     if ($b) {
 8         echo "B HIT\n";
 9     }
10 }
11 ?>
```

branch.php

```
…
xdebug_start_code_coverage(
    XDEBUG_CC_UNUSED | XDEBUG_CC_DEAD_CODE | XDEBUG_CC_BRANCH_CHECK
);

ifthenelse( true, false );
ifthenelse( false, true );
…
```



Xdebug code coverage output

/home/derick/dev/php/xdebug-demos/deep-dive-cov / (Dashboard)

	Code Coverage														
	Lines			Branches			Paths			Functions and Methods			Classes and Traits		
Total		0.17%	11 / 6655		0.27%	7 / 2606		0.01%	2 / 27940		0.33%	2 / 604		0.00%	0 / 113
vendor		0.05%	3 / 6647		0.08%	2 / 2601		0.00%	0 / 27936		0.17%	1 / 603		0.00%	0 / 113
run-cov.php [line] [branch] [path]		100.00%	4 / 4		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0
test.php [line] [branch] [path]		100.00%	4 / 4		100.00%	5 / 5		50.00%	2 / 4		100.00%	1 / 1		n/a	0 / 0

Legend

Low: 0% to 50% **Medium**: 50% to 90% **High**: 90% to 100%

Generated by php-code-coverage 10.1.7 using PHP 8.2.2-dev at Mon Oct 16 9:29:09 UTC 2023.

(!) Warning: DOMDocument::load(): I/O warning : failed to load external entity "/home/httpd/presentations/slides/internals/jit.xml" in /home/httpd/pres2/show2.php on line *215*

Call Stack

#	Time	Memory	Function	Location
1	0.0014	670928	{main}( )	.../show2.php:0
2	0.0065	1012144	Presentation->display($slideNr = '50')	.../show2.php:154
3	0.0065	1013152	load($filename = '/home/httpd/pres2/presentations/slides/internals/jit.xml')	.../show2.php:215

(!) Fatal error: Uncaught ezcTemplateRuntimeException: The external (use) variable 'node' is not set in template: /home/httpd/pres2/presentations/templates/default/slide.ezt and called from the application code in /tmp/template-cache/compiled_templates/xhtml-updqr0/slide-5eb6f7fc995a21e41e7fdbd1e4869726.php on line *9*

(!) ezcTemplateRuntimeException: The external (use) variable 'node' is not set in template: /home/httpd/pres2/presentations/templates/default/slide.ezt and called from the application code in /tmp/template-cache/compiled_templates/xhtml-updqr0/slide-5eb6f7fc995a21e41e7fdbd1e4869726.php on line *9*

Call Stack

#	Time	Memory	Function	Location
1	0.0014	670928	{main}( )	.../show2.php:0
2	0.0065	1012144	Presentation->display($slideNr = '50')	.../show2.php:154
3	0.0075	1073624	ezcTemplate->process($location = 'slide.ezt', $config = ???)	.../show2.php:228
4	0.0088	1156176	ezcTemplateCompiledCode->execute( )	.../template.php:341
5	0.0094	1202840	include('/tmp/template-cache/compiled_templates/xhtml-updqr0/slide-5eb6f7fc995a21e41e7fdbd1e4869726.php)	.../compiled_code.php:188

Recap

Recap

- Stages:
  Code → Tokens → Parsing → AST → Byte Code
- All looping structures are jumps
- Code analysis for Fun and Profit
- Collections branch with each individual step:
  `https://github.com/derickr/php-src/tree/collections`

Tools

- PHP's tokenizer: http://php.net/tokenizer
- Nikita's AST extension: https://github.com/nikic/php-ast
- VLD: https://pecl.php.net/vld

Any Queries?

Episode #103
Disjunctive Normal Form (DNF) Types

In this episode of "PHP Internals News" I talk with George Peter Banyard (Website, Twitter, GitHub, GitLab) about the "Disjunctive Normal Form Types" RFC that he has proposed with Larry Garfield.

Friday June 24th, 2022 — 09:07 BST

Episode #102
Add True Type

In this episode of "PHP Internals News" I talk with George Peter Banyard (Website, Twitter, GitHub, GitLab) about the "Add True Type" RFC that he has proposed.

Thursday June 2nd, 2022 — 09:06 BST

Episode #101
More Partially Supported Callable Deprecations

In this episode of "PHP Internals News" I talk with Juliette Reinders Folmer (Website, Twitter, GitHub) about the "More

Spotiy iTunes RSS Feed Twitter

BECOME A PATRON

https://phpinternals.news

Slides & Resources

Contact

derick@php.net—@derickr