我的PHP学习成长之路

秦朋@滴滴出行

PHPCON
www.phpconchina.com

个人简介

秦朋（盘古大叔）

- 滴滴高级开发工程师：分单引擎
- **《PHP7内核剖析》**作者
- 开源项目：PHP-AOP扩展、 Memcached Go客户端
- GitHub：**https://github.com/pangudashu**

你是否有过以下困惑?

- 困惑1：整天做业务，不知道该如何成长?
- 困惑2：搞了几年PHP，早已滚瓜烂熟，是否要转其它语言?
- 困惑3：抛开架构，单纯在PHP方向如何进阶?

一个底层码农的PHP之旅.....

1、接触

- 2011年，大二第一次接触PHP
- CMS：织梦、齐博、discuz、ecshop…
- 做企业站：套模板、改html、忽悠…
- 不会PHP…

2、学习

- 接到“大项目”-改一个商城：套模板玩不转了、忽悠不住人了……
- 还想赚钱？怎么办？学习！
- 基础：C、Sql Server、数据结构、计算机网络...
- 看书学语法、数据库->抄示例->抄项目->写项目

- 经历了2个月后get：PHP语法、连数据库、套模板、thinkphp…
- 自我感觉很很很良好…

3、实习

2013.7-2014.3 360问答

- 收获：
  - 打开了相对较广的技术视野：Memcached缓存、Mysql、Linux、PHP框架、Smarty、MVC、Nginx、服务化、消息队列…
  - 团队约束下养成了规范的编码习惯
- 感谢 @黑夜路人

4、工作

2014.7-2015.11 搜狗

- 搞业务逻辑、写API接口
- CURD、套模板、CURD、套模板、CURD、套模板...
- 感觉掌握的技术完全能招架业务了，自我感觉很很很良好...

困惑1：
整天做业务，不知道该如何成长？

经验1：看源码

- **有选择的、带着目的看**：从业务中常用的技术开始看
- **善于分解源码结构**：理清楚源码的结构
- **掌握看源码的力度**：关键的部分深入看，其它部分粗略看
- **曲线定律**：遇到难解的部分可以选择先绕过

- **构建知识体系**：源码只是引子、抓手，通过看源码找到自己技术上的短板，然后学习、掌握
- **领会架构、设计模式**：为业务提供参考，学以致用，**反驱业务**
- **不止于看**：善于揣测，看一部分先停下来，从实现者的角度思考后面会如何实现，然后去验证，边看边猜，**训练思维方式**

我看过几个项目：

- **ThinkPHP源码**： 统一入口、MVC分层、自动加载、路由、模板引擎
- **Memcached源码**： 多线程模式、TCP、二进制/文本协议、解包/封包、生成者-消费者模型、IO复用
- **Yaf源码**： PHP扩展的基本结构/写法、内部函数/类等的定义
- **Beego源码**： 带着PHP框架的思维看，比较差别

源码只是看看就行了？NO！

经验2：抄源码

- 为什么要抄?
  - 纸上得来终觉浅，事必躬行，加深印象、真正掌握
- 什么时候抄?
  - 对源码有个整体概念、看个差不多、总结后

- 怎么抄?
  - 不要大面积、直接拷贝
  - 先自己写，不会的地方再参考源码
  - 尽可能的根据自己的思路实现

- 抄过Yaf：扩展看的懂了，也会写了
- 抄过QFrame、ThinkPHP：PHP框架？So Easy!
- 抄过Memcached：TCP服务？No Problem！
- 抄过Beego：Go也算入门了

……

4、工作

回到工作中：

- ~~CURD、套模板、CURD、套模板、CURD、套模板……~~
- ~~感觉掌握的技术完全能招架业务了，自我感觉很很很良好…~~

之前看似没有挑战的业务也有很多可做之处了

总结：

- 如果在业务中看不到成长，更可能的原因是功力不够
- 做业务并非没有成长，善于借鉴开源的力量
- 了解底层并不一定就要写底层，设计模式、架构设计这些都是收获

2015.11 滴滴

- 主导新业务线技术实现：模块划分、框架分层规划、分库分表设计、缓存方案……
- 技术得到提升了，也应用到业务中去了，也是团队核心成员了
- 觉得以后项目完全可以胜任了，自我感觉很很很良好…

困惑2：
搞了几年PHP，早已滚瓜烂熟，是否要转其它语言？

2017.4 滴滴

- 毅然抛弃PHP，转Golang、C++，分单引擎

- Go并非完美无缺：
  - Gc问题
  - 内存泄漏
  - 开发效率低
  - 使用繁琐（与PHP相比）

- 总结：
  - 认识到语言只是工具，各有所长，掌握多门语言是必要的，在不同的场景选择适合的语言

重新思考PHP：

- 短板在哪？性能真的低吗？为什么低？
- 简洁==简单？
- Go/C++可以实现的东西PHP为什么不行？
- 能否在PHP上进一步有所突破？

困惑3：
抛开架构，单纯在PHP方向如何进阶?

或许可以这样：

- 通过扩展壮大PHP的功能：比如swoole
- 深入理解PHP：读PHP源码、理解内部实现
- 尝试解决PHP报出的bug：https://bugs.php.net/
- 尝试实现PHP新的语言特性：比如go的协程、defer…

  https://wiki.php.net/rfc/
- 再再进一步…
- 从“写”PHP到写PHP：成为鸟哥那样的男人

为什么要看内核？

- 学习内核的方法论、设计模式、架构模式：即使不写内核也可以学到很多，这些东西更可贵
- 提升扩展开发能力：熟悉内核后再开发扩展事半功倍
- 提高解决PHP问题的能力，比如：

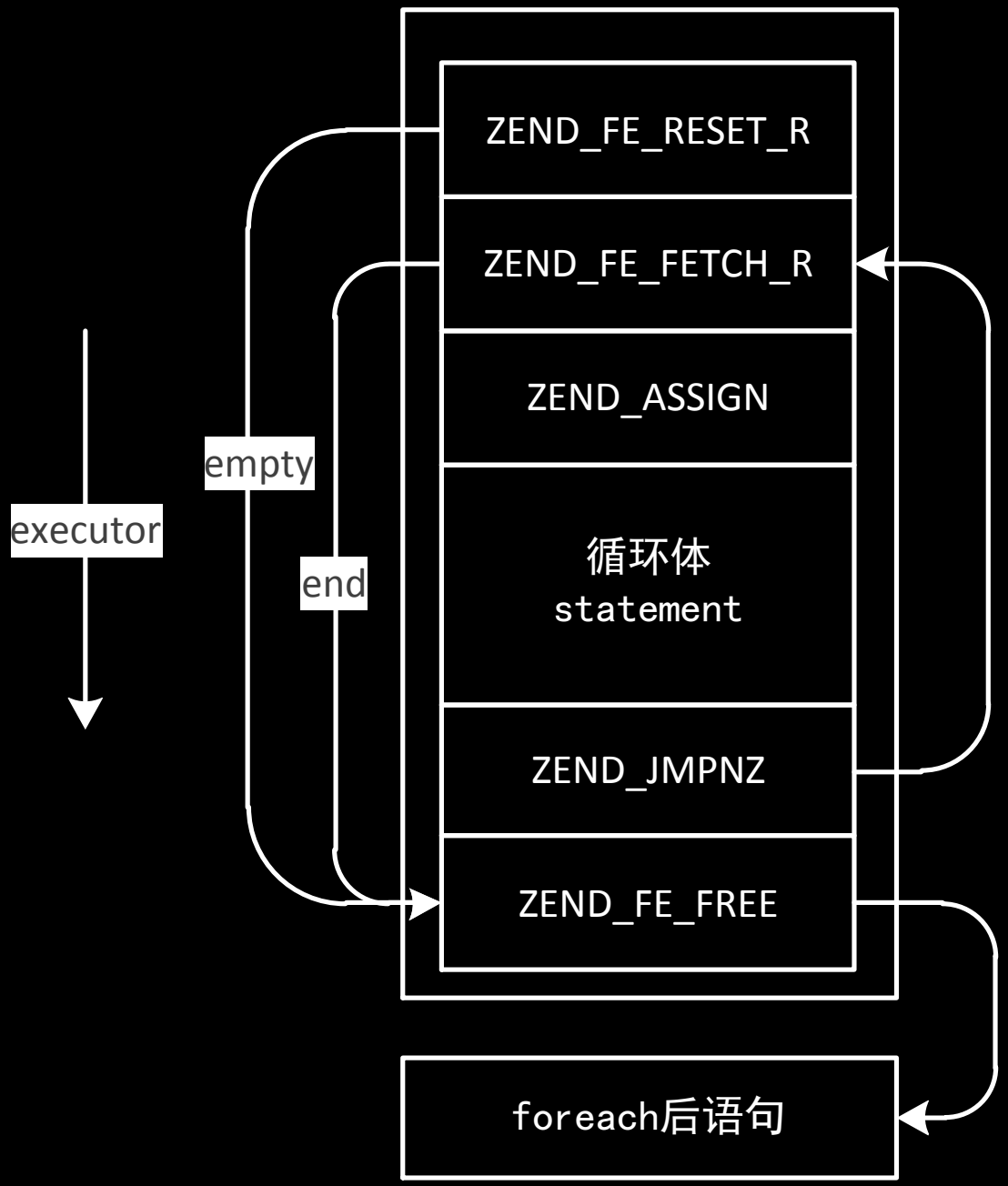
```php
$arr = array(1,2,3);

foreach($arr as &$v){
    //...
}

foreach($arr as $v){
    $v = 4;
}

print_r($arr);
```

```
Array
(
    [0] => 1
    [1] => 2
    [2] => 4
)
```




```php
//第一次遍历
$v = &$arr[0];
$v = &$arr[1];
$v = &$arr[2];

//第二次遍历
$v = 4; //$arr[2] = 4
$v = 4; //$arr[2] = 4
$v = 4; //$arr[2] = 4
```

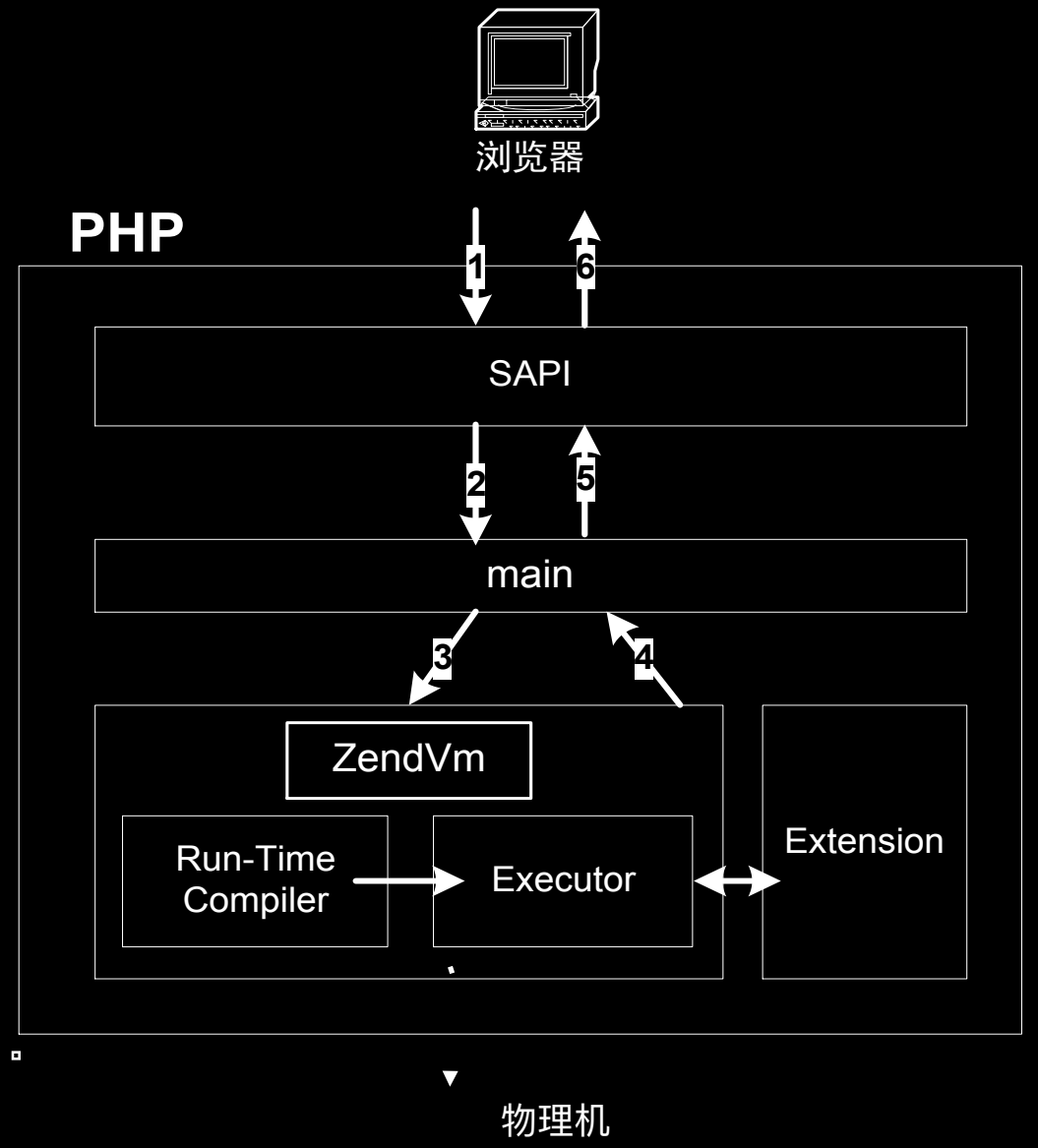
内核怎么看？

- 理清楚PHP框架结构：sapi、php、zend、ext...
- 从最核心的zend模块入手，搞清楚PHP核心解析流程
- 拆解PHP解析、执行阶段，逐个攻克
- 写简单的示例，用gdb追各个阶段
- 先看相关的数据结构，根据结构成员猜测实现方式

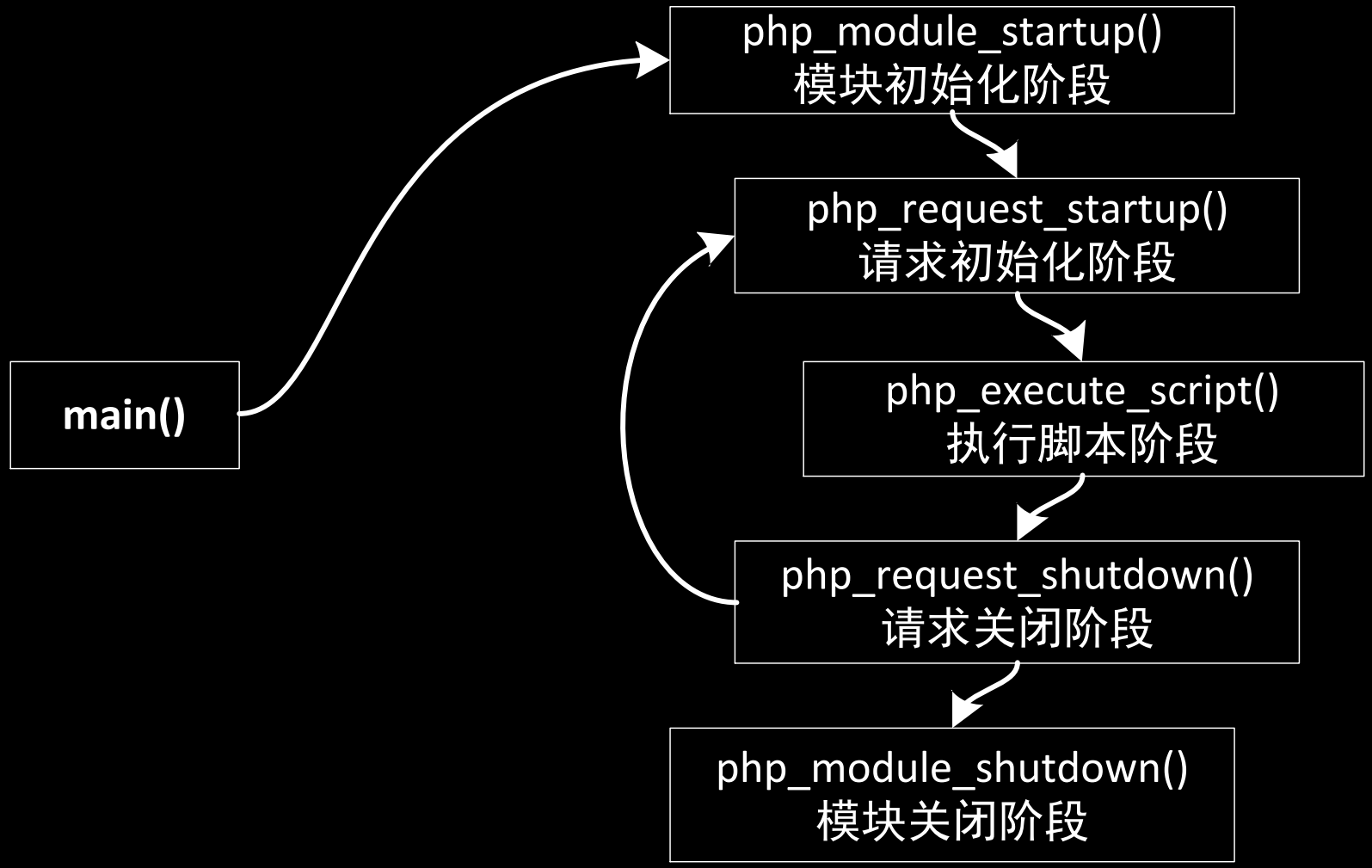
5、内核探索

- PHP内核的框架结构是什么样子的?

  - **SAPI**：接入层
  - **PHP**：负责输入、输出、web通信等工作
  - **Zend VM**：PHP脚本的解析执行
  - **PHP/Zend扩展**

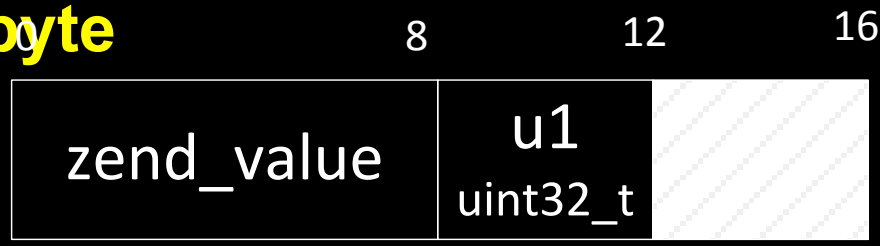


●一个请求都经历那些阶段（生命周期）？



●语言的基础—变量是如何实现的?

- zval、zend_value：变量的内部结构
- double、long、bool直接存储在value
- zend_array、zend_string
  、zend_object、zend_resource
  、zend_reference
- 16byte




```
struct _zval_struct {
    zend_value        value;          /* value */
    union {
        struct {
            ZEND_ENDIAN_LOHI_4(
                zend_uchar    type,         /* active type */
                zend_uchar    type_flags,
                zend_uchar    const_flags,
                zend_uchar    reserved)     /* call info for EX(This) */
        } v;
        uint32_t type_info;
    } u1;
    union {
        uint32_t     var_flags;
        uint32_t     next;                 /* hash collision chain */
        uint32_t     cache_slot;           /* literal cache slot */
        uint32_t     lineno;               /* line number (for ast nodes) */
        uint32_t     num_args;             /* arguments number for EX(This)
 */
        uint32_t     fe_pos;               /* foreach position */
        uint32_t     fe_iter_idx;          /* foreach iterator index */
    } u2;
};
```

●变量自动回收是怎么实现?

- **引用计数**
  - 变量赋值只增加value的引用次数，共用value
  - 引用计数等于0时释放value
- **写时复制**
  - 引用计数大于0的value发生写操作时进行分离

```
$a = array(1);      //a      -> zend_array_1(refcount=1)
$b = $a;            //a、b -> zend_array_1(refcount=2)
$b[] = 2;           //a      -> zend_array_1(refcount=1)
                    //b      -> zend_array_2(refcount=1)
var_dump($a);
```

●PHP脚本解析执行经历了哪些过程?

PHP脚本

编译器
解析脚本，生成opline指令

单词分析、语法解析
生成抽象语法树

编译AST
生成指令集、符号表

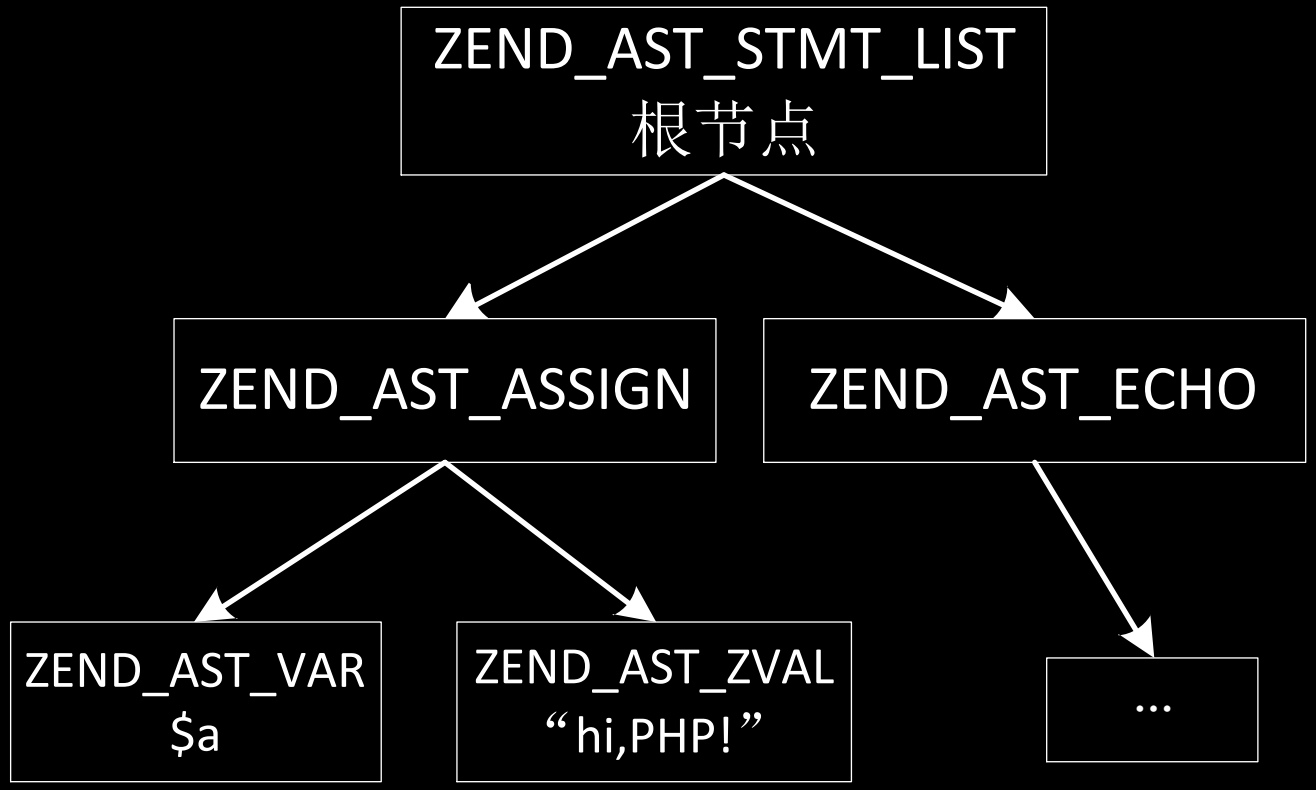
执行器
Call opcode对应的机器指令

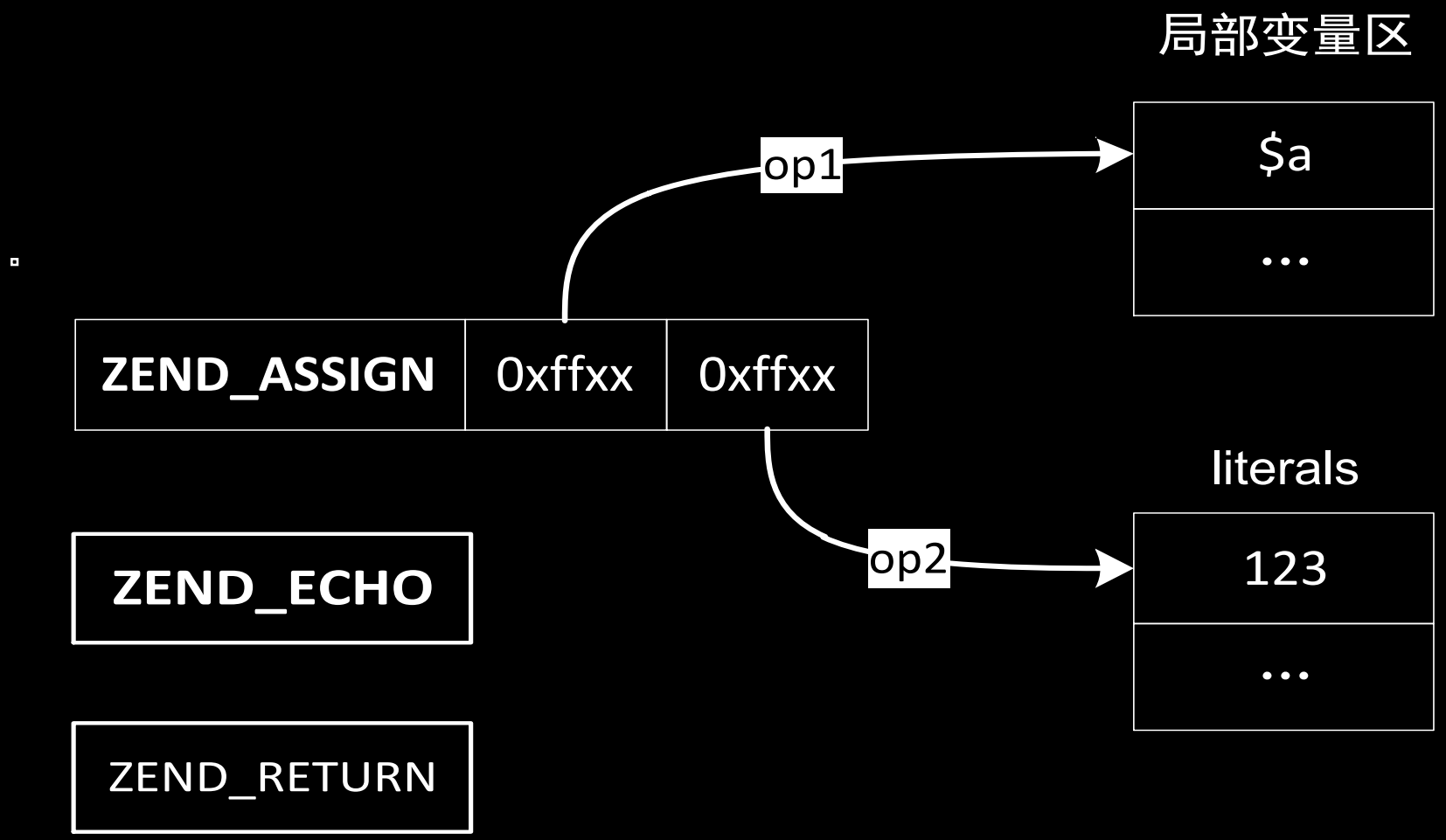
1）编译：

PHP脚本 -----> AST -------> opcodes

```
<?php

$a = “hi, PHP!”;
echo $a;
```





zend_op_array

2）执行：

CALL、GOTO、SWITCH模式

CALL模式：

```
LOOP：
    CALL HANDLER
END
```

```
zend_execute_data *execute_data = ex;
while (1) {
    int ret;
    //执行当前指令
    if (UNEXPECTED((ret = ((opcode_handler_t)execute_data->opline->handler
    )(execute_data)) != 0)) {
        if (EXPECTED(ret > 0)) {
            execute_data = EG(current_execute_data);
        } else {
            return;
        }
    }
}
```

不止于此

- 语言特性：
  - 函数、面向对象、命名空间
  - 全局变量、静态变量、常量
  - 条件分支： if、else、switch
  - 循环结构：do while、while、for、foreach
  - 中断及跳转：break、goto
  - 异常处理：try catch
  - 文件引用：Include、require
  - …
- 内存操作：ZendMM
- 线程安全：TSRM
- Opcache：缓存、优化、JIT
- ……

学习资料

- http://www.laruence.com/
- http://nikic.github.io/
- http://www.phpinternalsbook.com/
- https://github.com/pangudashu/php7-internal
- Or：