# 

# PHP后台1.0版本

编写日期：2018.1.20

编写人：刘剑华

[PHP后台1.0版本 0](#_Toc17797)

[1. 关于架构 2](#_Toc10380)

[1.1 PHP底层架构 2](#_Toc12657)

[2. 关于框架（TP5、 YII、 laravel） 6](#_Toc3457)

[(1) TP5 免费开源 框架 适用于中小型项目 6](#_Toc12041)

[(2) Laravel 为WEB艺术家创造的PHP框架 7](#_Toc7036)

[3. 关于组件库 8](#_Toc19568)

[4. 关于解决方案 9](#_Toc23132)

[4.1 php扩展中如何定义线程安全的全局对象？ 9](#_Toc815)

[4.2 怎么实现PHP伪静态 9](#_Toc12834)

[5.关于项目开发方案： 9](#_Toc1397)

[附录1 10](#_Toc5545)

[过滤信息（Filtering） 10](#_Toc1649)

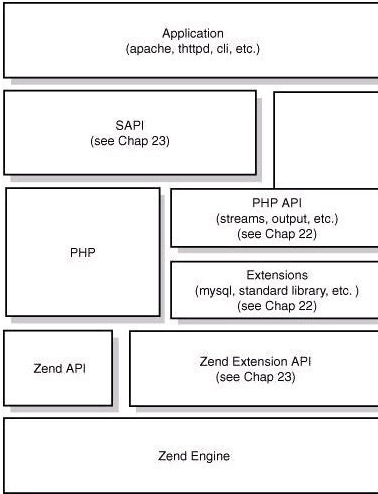
[状态码（Status Codes） 11](#_Toc20514)

[返回结果 12](#_Toc24162)

[关于：Hypermedia API 12](#_Toc29524)

## 关于架构

### PHP底层架构



* + - 1. 简介：底层架构一般分为4层体系。
* Zend引擎：Zend整体用纯C实现，是PHP的内核部分，它将PHP代码翻译（词法、语法解析等一系列编译过程）为可执行opcode的处理并实现相应的处理方法、实现了基本的数据结构（如hashtable、oo）、内存分配及管理、提供了相应的api方法供外部调用，是一切的核心，所有的外围功能均围绕Zend实现。
* Extensions：围绕着Zend引擎，extensions通过组件式的方式提供各种基础服务，我们常见的各种内置函数（如array系列）、标准库等都是通过extension来实现，用户也可以根据需要实现自己的extension以达到功能扩展、性能优化等目的（如贴吧正在使用的PHP中间层、富文本解析就是extension的典型应用）。
* Sapi：Sapi全称是Server Application Programming Interface，也就是服务端应用编程接口，Sapi通过一系列钩子函数，使得PHP可以和外围交互数据，这是PHP非常优雅和成功的一个设计，通过sapi成功的将PHP本身和上层应用解耦隔离，PHP可以不再考虑如何针对不同应用进行兼容，而应用本身也可以针对自己的特点实现不同的处理方式。
* 上层应用：这就是我们平时编写的PHP程序，通过不同的sapi方式得到各种各样的应用模式，如通过webserver实现web应用、在命令行下以脚本方式运行等等

#### 关于架构解读示例

假设有如下4个变量：$strA=‘123’; $strB = ‘456’; $intA=123; intB=456;

例：现在对如下的几种字符串拼接方式做一个比较和说明

$res = $strA.$strB和$res = “$strA$strB”

这种情况下，zend会重新malloc一块内存并进行相应处理，其速度一般

$strA = $strA.$strB

这种是速度最快的，zend会在当前strA基础上直接relloc，避免重复拷贝

$res = $intA.$intB

这种速度较慢，因为需要做隐式的格式转换，实际编写程序中也应该注意尽量避免

$strA = sprint(“%s%s”,$strA.$strB);

这会是最慢的一种方式，因为sprintf在PHP中并不是一个语言结构，本身对于格式识别和处理就需要耗费比较多时间，另外本身机制也是malloc。不过sprintf的方式最具可读性，实际中可以根据具体情况灵活选择。

foreach操作如何实现？对一个数组的foreach就是通过遍历hashtable中的双向链表完成。对于索引数组，通过foreach遍历效率比for高很多，省去了key->value的查找。count操作直接调用HashTable->NumOfElements，O(1)操作。对于’123’这样的字符串，zend会转换为其整数形式。$arr[‘123’]和$arr[123]是等价的

#### 关于数据库中架设分表

我们测试过这个方法，但是发现如果要扩展它的伸缩性，代价太高。如果用一个整体的数据结构，在大流量面前，你将会面临服务器硬件的问题。在 MU 里面。用户们都被分布到独立的表格当中，并且可以轻易地组织起来。举个例子，[http://WordPress.com](https://link.zhihu.com/?target=http://WordPress.com" \t "_blank) 把用户的数据分散存储到 4096 个数据库中，这些数据库可以分散大规模的数据访问，实现流量和压力分流。

数据表的可迁移性让代码（blog）可以运行得更快，并且让系统具备更强的伸缩性。依靠强大的缓存策略和灵活的数据库运用策略， Matt 向人们展示了时下最流行的 Facebook 和 [http://Wordpress.com](https://link.zhihu.com/?target=http://Wordpress.com" \t "_blank) 都可以在 PHP 下稳定运行，并且处理惊人的访问量。

创建索引时尽量使用聚集索引

#### 关于PHP的缓存机制****Redis/Memcached****（多数作用于中型以上项目）

Redis支持服务器端的数据操作：Redis相比Memcached来说，拥有更多的数据结构和并支持更丰富的数据操作，通常在Memcached里，你需要将数据拿到客户端来进行类似的修改再set回去。这大大增加了网络IO的次数和数据体积。在Redis中，这些复杂的操作通常和一般的GET/SET一样高效。所以，如果需要缓存能够支持更复杂的结构和操作，那么Redis会是不错的选择。

内存使用效率对比：使用简单的key-value存储的话，Memcached的内存利用率更高，而如果Redis采用hash结构来做key-value存储，由于其组合式的压缩，其内存利用率会高于Memcached。

性能对比：由于Redis只使用单核，而Memcached可以使用多核，所以平均每一个核上Redis在存储小数据时比Memcached性能更高。而在100k以上的数据中，Memcached性能要高于Redis，虽然Redis最近也在存储大数据的性能上进行优化，但是比起Memcached，还是稍有逊色。

但是相对而言json更喜欢使用redis

**1.Redis使用最佳方式是全部数据in-memory。  
　　2.Redis更多场景是作为Memcached的替代者来使用。  
　　3.当需要除key/value之外的更多数据类型支持时，使用Redis更合适。  
　　4.当存储的数据不能被剔除时，使用Redis更合适**。

#### 关于Nginx、Apache HTTP Server 、**[Apache Tomcat](https://link.zhihu.com/?target=http://tomcat.apache.org/" \t "https://www.zhihu.com/question/32212996/answer/_blank)**服务

* + 1. Nuinx作为一个高性能，有点逼格的HTTP和反向代理服务器，这个东西还是比较的好用的，通常用来作为静态内容服务和代理服务器，将外来请求转发给应用（Tomcat，Django等）
    2. Tomcat呢更多是用来作为一个容器，让java web app跑起来 当然也可以提供http服务，通常在内网和不需要流控的小型服务场景使用
    3. Apache、主要用于一些语言：perl、Python、tcl和PHP，还有一些SSL服务和URL重写、过滤支持等

## 关于框架（TP5、 YII、 laravel）

### TP5 免费开源 框架 适用于中小型项目

* + 1. 优缺点：version 5.01
       1. **优点**：对API相对友好，支持多入口配置，只需修改配置文件即可达到目的，目前此版本相对稳定，BUG存量较少 推荐使用。
       2. **缺点**：随着业务需求的递增和多样化，会逐渐满足不了开发需求，建议在开发过程中考虑项目迭代问题，适当放弃框架部分语法，为以后版本迭代预留空间。

### Laravel 为WEB艺术家创造的PHP框架

* + 1. 优缺点：version 5.4
       1. **缺点**：臃肿！开发一个组件服务，这种类型的服务一般都是功能比较单一，比如用户服务、搜索服务、支付服务，在这种场景下用laravel开发就显得比较臃肿了，因为在特定服务中可能只用到laravel部分基础服务，而laravel则默认加载了全部，貌似新版本的laravel正在改善这问题
       2. 慢与资源消耗，慢，真的慢。优化只能从**redis memcache opcache php7 hhvm** 去处理。
       3. **优点：**laravel最大的特点和优秀之处就是集合了php比较新的特性，以及各种各样的设计模式，Ioc容器，依赖注入等。因此laravel是一个适合学习的框架。
  1. **Yii 基于组件、用于开发大型 Web 应用的高性能 PHP 框架**

## 关于组件库

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **名称** | **依赖** | **需求版本** | **备注** |
| 1 | **[PHP Error](http://phperror.net/" \t "https://www.evget.com/article/2014/9/19/_blank)** | ****php类库错误组件**** | PHP5.0+ |  |
| 2 | ****[Composer](https://getcomposer.org/" \t "http://blog.csdn.net/u010832551/article/details/_blank)****/ ****[Packagist](https://packagist.org/" \t "http://blog.csdn.net/u010832551/article/details/_blank)**** | ****一个包和类型管理器**** | Php5.4+ | https://github.com/composer/composer.git |
| 3 | ****[pickle](https://github.com/FriendsOfPHP/pickle" \t "http://blog.csdn.net/u010832551/article/details/_blank)**** | ****PHP扩展安装器**** |  |  |
| 4 | ****[PHPUnit](https://github.com/sebastianbergmann/phpunit" \t "http://blog.csdn.net/u010832551/article/details/_blank)**** | 一****个单元测试框架组件**** |  | https://github.com/VisualPHPUnit/VisualPHPUnit.git |
| 5 | ****[xDebug](https://github.com/xdebug/xdebug" \t "http://blog.csdn.net/u010832551/article/details/_blank)**** | ****一个PHP的调试和分析工具**** |  | https://github.com/xdebug/xdebug.git |
| 6 | ****PHPexce**** | ****PHP执行导出导入EXCEL所使用**** | Php5.4+ | https://github.com/PHPOffice/PHPExcel.git |
| 7 | ****PHPmailer**** | ****一个邮件发送组件支持绝大部分需求**** | Php5+ | https://github.com/PHPMailer/PHPMailer.git |
| 8 | .... |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

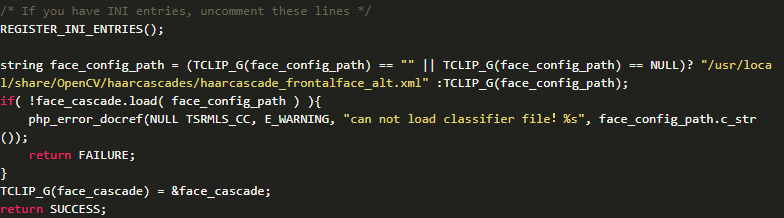
## 关于解决方案

### 4.1 php扩展中如何定义线程安全的全局对象？

答：1.首先定义了一个全局变量。

static CascadeClassifier face\_cascade;  
2.修改PHP\_MINIT\_FUNCTION(tclip)。代码如下：

PHP\_MINIT\_FUNCTION(tclip)  
{



}

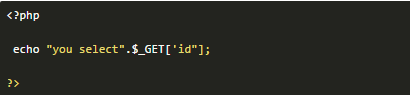
调用全局对象的方式如下：

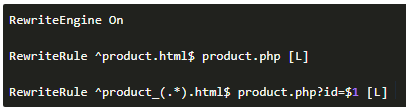
((CascadeClassifier \*)**TCLIP\_G(face\_cascade)**)->detectMultiScale( img\_gray, faces, 1.1, 2, 0|CV\_HAAR\_SCALE\_IMAGE, Size(30, 30) );  
注意，**php扩展想线程安全，那就用相关宏来访问全局变量**。如**TCLIP\_G(face\_cascade)**。

### 4.2 **怎么实现PHP伪静态**

**前置：**LoadModule rewrite\_module modules/mod\_rewrite.so 开启PHPINI中的服务



product.php  


**答：**  


## 5.**关于项目开发方案：**

#### 5.1输出文档有：

* 1. API文档
  2. Sql文档
  3. 路由文档

项目开发阶段优先考虑可行性，其次为优化，再者为复用性。

新项目的API开发使用RESTFul API（附录1

控制代码的差异性，制定代码统一规范，并逐渐靠拢。

附录1：关于RESTFul API

规则：在RESTful架构中，每个网址代表一种资源（resource），所以网址中不能有动词，只能有名词，而且所用的名词往往与数据库的表格名对应。一般来说，数据库中的表都是同种记录的"集合"（collection），所以API中的名词也应该使用复数

* GET（SELECT）：从服务器取出资源（一项或多项）。
* POST（CREATE）：在服务器新建一个资源。
* PUT（UPDATE）：在服务器更新资源（客户端提供改变后的完整资源）。
* PATCH（UPDATE）：在服务器更新资源（客户端提供改变的属性）。
* DELETE（DELETE）：从服务器删除资源。

## 过滤信息（Filtering）

如果记录数量很多，服务器不可能都将它们返回给用户。API应该提供参数，过滤返回结果。

下面是一些常见的参数。

* ?limit=10：指定返回记录的数量
* ?offset=10：指定返回记录的开始位置。
* ?page=2&per\_page=100：指定第几页，以及每页的记录数。
* ?sortby=name&order=asc：指定返回结果按照哪个属性排序，以及排序顺序。
* ?animal\_type\_id=1：指定筛选条件

## 状态码（Status Codes）

* 200 OK - [GET]：服务器成功返回用户请求的数据，该操作是幂等的（Idempotent）。
* 201 CREATED - [POST/PUT/PATCH]：用户新建或修改数据成功。
* 202 Accepted - [\*]：表示一个请求已经进入后台排队（异步任务）
* 204 NO CONTENT - [DELETE]：用户删除数据成功。
* 400 INVALID REQUEST - [POST/PUT/PATCH]：用户发出的请求有错误，服务器没有进行新建或修改数据的操作，该操作是幂等的。
* 401 Unauthorized - [\*]：表示用户没有权限（令牌、用户名、密码错误）。
* 403 Forbidden - [\*] 表示用户得到授权（与401错误相对），但是访问是被禁止的。
* 404 NOT FOUND - [\*]：用户发出的请求针对的是不存在的记录，服务器没有进行操作，该操作是幂等的。
* 406 Not Acceptable - [GET]：用户请求的格式不可得（比如用户请求JSON格式，但是只有XML格式）。
* 410 Gone -[GET]：用户请求的资源被永久删除，且不会再得到的。
* 422 Unprocesable entity - [POST/PUT/PATCH] 当创建一个对象时，发生一个验证错误。
* 500 INTERNAL SERVER ERROR - [\*]：服务器发生错误，用户将无法判断发出的请求是否成功。

## 返回结果

* GET /collection：返回资源对象的列表（数组）
* GET /collection/resource：返回单个资源对象
* POST /collection：返回新生成的资源对象
* PUT /collection/resource：返回完整的资源对象
* PATCH /collection/resource：返回完整的资源对象
* DELETE /collection/resource：返回一个空文档

## 关于：Hypermedia API

暂不强制要求。