Neeke / SeasLog

A effective,fast,stable log extension for PHP. http://neeke.github.io/SeasLog/

135 commits | 3 branches | 9 releases | 5 contributors

Branch: master ▾ SeasLog / +

fixed #52

Neeke authored 8 days ago — latest commit b17e8611cd

Analyzer	update version start with 1.0.0	a year ago
CodeTips	format file directory	3 months ago
tests	add tests for issue #50	17 days ago
.gitignore	add some ignored file.	a year ago
CREDITS	add nly xlittle to CREDITS	3 months ago
EXPERIMENTAL	format file directory	3 months ago
LICENSE	update CREDITS	a year ago
README.md	fixed #52	8 days ago
config.m4	format file directory	3 months ago
config.w32	format file directory	3 months ago
package.xml	update package.xml	15 days ago
php_seaslog.h	fixed issue #50	15 days ago
seaslog.c	fixed issue #50	15 days ago

Code
Issues 4
Pull requests 0
Pulse
Graphs

HTTPS clone URL

https://github.com

You can clone with HTTPS or Subversion.

Clone in Desktop

Download ZIP

README.md

SeasLog

A effective,fast,stable log extension for PHP

@author Chitao.Gao [neeke@php.net]

@交流群 312910117

简介

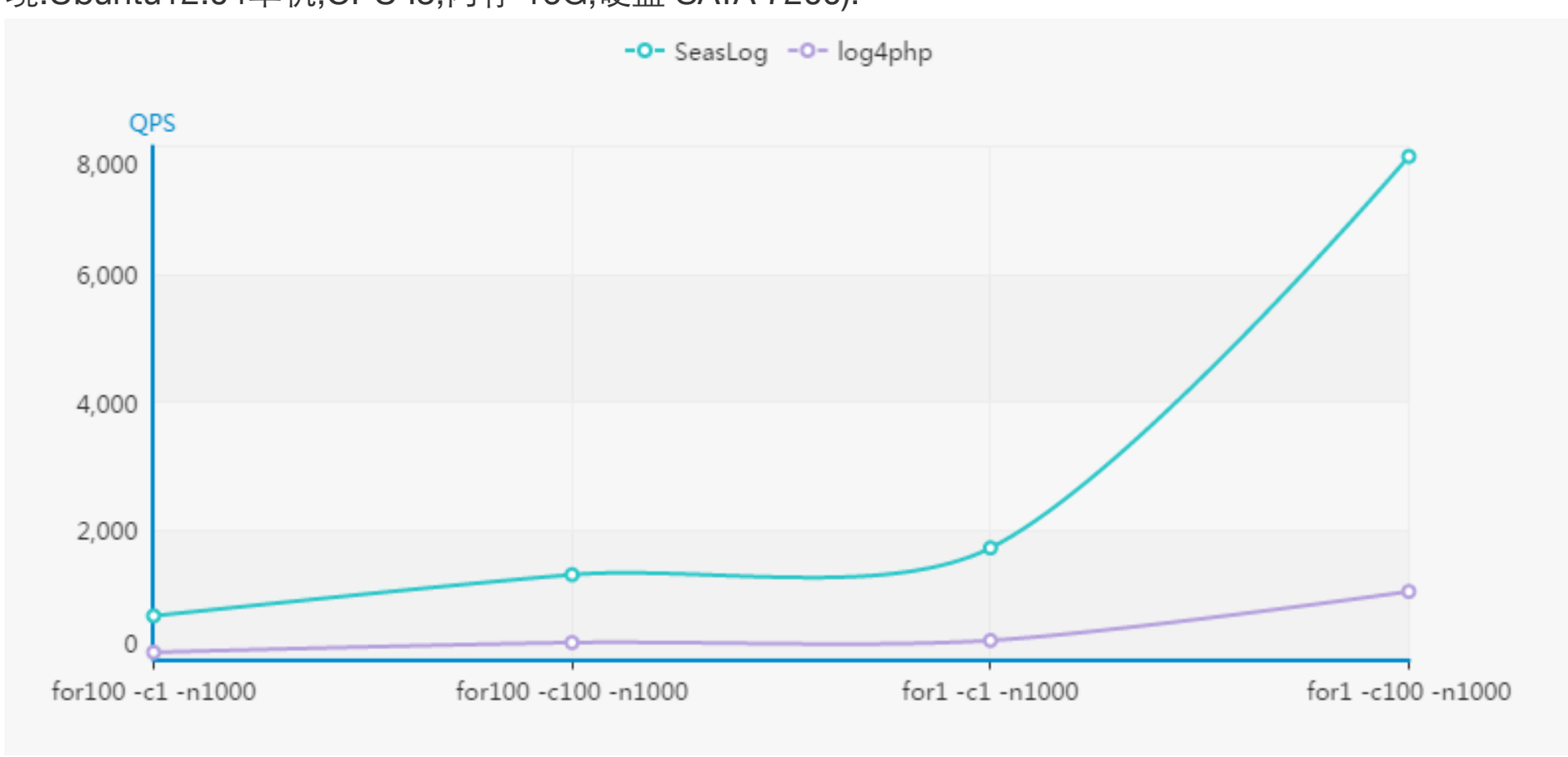
为什么使用SeasLog

log日志，通常是系统或软件、应用的运行记录。通过log的分析，可以方便用户了解系统或软件、应用的运行情况；如果你的应用log足够丰富，也可以分析以往用户的操作行为、类型喜好、地域分布或其他更多信息；如果一个应用的log同时也分了多个级别，那么可以很轻易地分析得到该应用的健康状况，及时发现问题并快速定位、解决问题，补救损失。

php内置error_log、syslog函数功能强大且性能极好，但由于各种缺陷(error_log无错误级别、无固定格式，syslog不分模块、与系统日志混合)，灵活度降低了很多，不能满足应用需求。

好消息是，有不少第三方的log类库弥补了上述缺陷，如log4php、plog、Analog等(当然也有很多应用在项目中自己开发的log类)。其中以log4php最为著名，设计精良、格式完美、文档完善、功能强大。推荐。

不过log4php在性能方面表现非常差,下图是SeasLog与log4php的ab并发性能测试(测试环境:Ubuntu12.04单机,CPU I3,内存 16G,硬盘 SATA 7200):



那么有没有一种log类库满足以下需求呢：

- 分模块、分级别
- 配置简单(最好是勿须配置)
- 日志格式清晰易读
- 应用简单、性能很棒

`SeasLog` 正是应此需求而生。

目前提供了什么

- 在PHP项目中便捷、规范地记录log
- 可配置的默认log目录与模块
- 指定log目录与获取当前配置

- 初步的分析预警框架
- 高效的日志缓冲、便捷的缓冲debug
- 遵循 PSR-3 日志接口规范
- 自动记录错误信息
- 自动记录异常信息

目标是怎样的

- 便捷、规范的log记录
- 高效的海量log分析
- 可配置、多途径的log预警

安装

编译安装 SeasLog

```
$ /path/to/phpize
$ ./configure --with-php-config=/path/to/php-config
$ make && make install
```

PECL安装SeasLog

```
$ pecl install seaslog
```

seaslog.ini的配置

```
; configuration for php SeasLog module
extension = seaslog.so
seaslog.default_basepath = /log/seaslog-test    ;默认log根目录
seaslog.default_logger = default                ;默认logger目录
seaslog.disting_type = 1                        ;是否以type分文件 1是 0否(默认)
seaslog.disting_by_hour = 1                     ;是否每小时划分一个文件 1是 0否(默认)
seaslog.use_buffer = 1                          ;是否启用buffer 1是 0否(默认)
seaslog.buffer_size = 100                       ;buffer中缓冲数量 默认0(不使用buffer_size)
seaslog.level = 0                               ;记录日志级别 默认0(所有日志)
seaslog.trace_error = 1                         ;自动记录错误 默认1(开启)
seaslog.trace_exception = 0                     ;自动记录异常信息 默认0(关闭)
```

> `seaslog.disting_type = 1` 开启以type分文件，即log文件区分info\warn\erro
>
> `seaslog.disting_by_hour = 1` 开启每小时划分一个文件
>
> `seaslog.use_buffer = 1` 开启buffer。默认关闭。当开启此项时，日志预存于内存，当请求结束时(或异常退出时)一次写入文件。
>
> `seaslog.buffer_size = 100` 设置缓冲数量为100. 默认为0,即无缓冲数量限制.当buffer_size大于0时,缓冲量达到该值则写一次文件.
>
> `seaslog.level = 3` 记录的日志级别.默认为0,即所有日志均记录。当level为1时,关注debug以上级别(包括debug)，以此类推。level大于8时，所有日志均不记录。

使用

常量与函数

常量列表

`SeasLog` 共将日志分成8个级别

- SEASLOG_DEBUG "debug"
- SEASLOG_INFO "info"
- SEASLOG_NOTICE "notice"
- SEASLOG_WARNING "warning"
- SEASLOG_ERROR "error"
- SEASLOG_CRITICAL "critical"
- SEASLOG_ALERT "alert"
- SEASLOG_EMERGENCY "emergency"

```
var_dump(SEASLOG_DEBUG,SEASLOG_INFO,SEASLOG_NOTICE);
/*
string('debug') debug级别
string('info')  info级别
string('notice') notice级别
*/
```

函数列表

`SeasLog` 提供了这样一组函数，可以方便地获取与设置根目录、模块目录、快速写入与统计log。 相信从下述伪代码的注释中，您可以快速获取函数信息，具体使用将紧接其后：

```
<?php
/**
 * @author ciogao@gmail.com
 * Date: 14-1-27 下午4:47
 */

class SeasLog
{
    public function __construct()
    {
        #SeasLog init
    }

    public function __destruct()
    {
        #SeasLog distroy
    }

    /**
     * 设置basePath
     * @param $basePath
     * @return bool
     */
    static public function setBasePath($basePath)
    {
        return TRUE;
    }

    /**
     * 获取basePath
     * @return string
     */
    static public function getBasePath()
    {
        return 'the base_path';
    }

    /**
     * 设置模块目录
```

```php
     * @param $module
     * @return bool
     */
    static public function setLogger($module)
    {
        return TRUE;
    }

    /**
     * 获取最后一次设置的模块目录
     * @return string
     */
    static public function getLastLogger()
    {
        return 'the lastLogger';
    }

    /**
     * 统计所有类型（或单个类型）行数
     * @param string $level
     * @param string $log_path
     * @param null $key_word
     * @return array | long
     */
    static public function analyzerCount($level = 'all',$log_path = '*',$key_word = NULL)
    {
        return array();
    }

    /**
     * 以数组形式，快速取出某类型log的各行详情
     * @param $level
     * @param string $log_path
     * @param null $key_word
     * @param int $start
     * @param int $limit
     * @return array
     */
    static public function analyzerDetail($level = SEASLOG_INFO,$log_path = '*',$key_word = N
    {
        return array();
    }

    /**
     * 获得当前日志buffer中的内容
     * @return array
     */
    static public function getBuffer()
    {
        return array();
    }

    /**
     * 记录debug日志
     * @param $message
     * @param array $content
     * @param string $module
     */
    static public function debug($message,array $content = array(),$module = '')
    {
        #$level = SEASLOG_DEBUG
    }

    /**
     * 记录info日志
     * @param $message
     * @param array $content
     * @param string $module
     */
    static public function info($message,array $content = array(),$module = '')
```

```
{
    #$level = SEASLOG_INFO
}

/**
 * 记录notice日志
 * @param $message
 * @param array $content
 * @param string $module
 */
static public function notice($message,array $content = array(),$module = '')
{
    #$level = SEASLOG_NOTICE
}

/**
 * 记录warning日志
 * @param $message
 * @param array $content
 * @param string $module
 */
static public function warning($message,array $content = array(),$module = '')
{
    #$level = SEASLOG_WARNING
}

/**
 * 记录error日志
 * @param $message
 * @param array $content
 * @param string $module
 */
static public function error($message,array $content = array(),$module = '')
{
    #$level = SEASLOG_ERROR
}

/**
 * 记录critical日志
 * @param $message
 * @param array $content
 * @param string $module
 */
static public function critical($message,array $content = array(),$module = '')
{
    #$level = SEASLOG_CRITICAL
}

/**
 * 记录alert日志
 * @param $message
 * @param array $content
 * @param string $module
 */
static public function alert($message,array $content = array(),$module = '')
{
    #$level = SEASLOG_ALERT
}

/**
 * 记录emergency日志
 * @param $message
 * @param array $content
 * @param string $module
 */
static public function emergency($message,array $content = array(),$module = '')
{
    #$level = SEASLOG_EMERGENCY
}
```

```
    /**
     * 通用日志方法
     * @param $level
     * @param $message
     * @param array $content
     * @param string $module
     */
    static public function log($level,$message,array $content = array(),$module = '')
    {

    }
}
```

SeasLog Logger的使用

获取与设置basePath

```
$basePath_1 = SeasLog::getBasePath();

SeasLog::setBasePath('/log/base_test');
$basePath_2 = SeasLog::getBasePath();

var_dump($basePath_1,$basePath_2);

/*
string(19) "/log/seaslog-ciogao"
string(14) "/log/base_test"
*/
```

> 直接使用 `SeasLog::getBasePath()`，将获取php.ini(seaslog.ini)中设置的 `seaslog.default_basepath` 的值。
>
> 使用 `SeasLog::setBasePath()` 函数，将改变 `SeasLog::getBasePath()` 的取值。

设置logger与获取lastLogger

```
$lastLogger_1 = SeasLog::getLastLogger();

SeasLog::setLogger('testModule/app1');
$lastLogger_2 = SeasLog::getLastLogger();

var_dump($lastLogger_1,$lastLogger_2);
/*
string(7) "default"
string(15) "testModule/app1"
*/
```

> 与basePath相类似的，
>
> 直接使用 `SeasLog::getLastLogger()`，将获取php.ini(seaslog.ini)中设置的 `seaslog.default_logger` 的值。
>
> 使用 `SeasLog::setLogger()` 函数，将改变 `SeasLog::getLastLogger()` 的取值。

快速写入log

上面已经设置过了basePath与logger，于是log记录的目录已经产生了，

> log记录目录 = basePath / logger / {fileName}.log log文件名，以 `年月日` 分文件，如今天是2014年02月18日期，那么 `{fileName}` = `20140218`;

还记得 `php.ini` 中设置的 `seaslog.disting_type` 吗?

默认的 `seaslog.disting_type = 0`，如果今天我使用了 `SeasLog` ，那么将产生最终的log文件:

- LogFile = basePath / logger / 20140218.log

如果 `seaslog.disting_type = 1`，则最终的log文件将是这样的三个文件

- infoLogFile = basePath / logger / INFO.20140218.log
- warnLogFile = basePath / logger / WARN.20140218.log
- erroLogFile = basePath / logger / ERRO.20140218.log

```
SeasLog::log(SEASLOG_ERROR,'this is a error test by ::log');

SeasLog::debug('this is a {userName} debug',array('{userName}' => 'neeke'));

SeasLog::info('this is a info log');

SeasLog::notice('this is a notice log');

SeasLog::warning('your {website} was down,please {action} it ASAP!',array('{website}' => 'git

SeasLog::error('a error log');

SeasLog::critical('some thing was critical');

SeasLog::alert('yes this is a {messageName}',array('{messageName}' => 'alertMSG'));

SeasLog::emergency('Just now, the house next door was completely burnt out! {note}',array('{r

/*
这些函数同时也接受第3个参数为logger的设置项
注意，当last_logger == 'default'时等同于:
SeasLog::setLogger('test/new/path');
SeasLog::error('test error 3');
如果已经在前文使用过SeasLog::setLogger()函数，第3个参数的log只在此处临时使用，不影响下文。
*/
```

> log格式统一为: `{type} | {pid} | {timeStamp} |{dateTime} | {logInfo}`
>
> ```
> error | 23625 | 1406422432.786 | 2014:07:27 08:53:52 | this is a error test by log
> debug | 23625 | 1406422432.786 | 2014:07:27 08:53:52 | this is a neeke debug
> info | 23625 | 1406422432.787 | 2014:07:27 08:53:52 | this is a info log
> notice | 23625 | 1406422432.787 | 2014:07:27 08:53:52 | this is a notice log
> warning | 23625 | 1406422432.787 | 2014:07:27 08:53:52 | your github.com was down,please
> error | 23625 | 1406422432.787 | 2014:07:27 08:53:52 | a error log
> critical | 23625 | 1406422432.787 | 2014:07:27 08:53:52 | some thing was critical
> emergency | 23625 | 1406422432.787 | 2014:07:27 08:53:52 | Just now, the house next door
> ```

SeasLog Analyzer的使用

快速统计某类型log的count值

`SeasLog` 在扩展中使用管道调用shell命令 `grep -wc` 快速地取得count值，并返回值(array || int)给PHP。

```
$countResult_1 = SeasLog::analyzerCount();
$countResult_2 = SeasLog::analyzerCount(SEASLOG_WARNING);
$countResult_3 = SeasLog::analyzerCount(SEASLOG_ERROR,date('Ymd',time()));

var_dump($countResult_1,$countResult_2,$countResult_3);
/*
```

```
array(8) {
  ["debug"]=>
  int(3)
  ["info"]=>
  int(3)
  ["notice"]=>
  int(3)
  ["warning"]=>
  int(3)
  ["error"]=>
  int(6)
  ["critical"]=>
  int(3)
  ["alert"]=>
  int(3)
  ["emergency"]=>
  int(3)
}


int(7)

int(1)

*/
```

获取某类型log列表

`SeasLog` 在扩展中使用管道调用shell命令 `grep -w` 快速地取得列表，并返回array给PHP。

```
$detailErrorArray_inAll    = SeasLog::analyzerDetail(SEASLOG_ERROR);
$detailErrorArray_today    = SeasLog::analyzerDetail(SEASLOG_ERROR,date('Ymd',time()));

var_dump($detailErrorArray_inAll,$detailErrorArray_today);

/*
SeasLog::analyzerDetail(SEASLOG_ERROR) == SeasLog::analyzerDetail(SEASLOG_ERROR,'*');
取当前模块下所有level为 SEASLOG_ERROR 的信息列表:
array(6) {
 [0] =>
  string(66) "ERRO | 8568 | 1393172042.717 | 2014:02:24 00:14:02 | test error 3 "
  [1] =>
  string(66) "ERRO | 8594 | 1393172044.104 | 2014:02:24 00:14:04 | test error 3 "
  [2] =>
  string(66) "ERRO | 8620 | 1393172044.862 | 2014:02:24 00:14:04 | test error 3 "
  [3] =>
  string(66) "ERRO | 8646 | 1393172045.989 | 2014:02:24 00:14:05 | test error 3 "
  [4] =>
  string(66) "ERRO | 8672 | 1393172047.882 | 2014:02:24 00:14:07 | test error 3 "
  [5] =>
  string(66) "ERRO | 8698 | 1393172048.736 | 2014:02:24 00:14:08 | test error 3 "
}

SeasLog::analyzerDetail(SEASLOG_ERROR,date('Ymd',time()));
只取得当前模块下，当前一天内,level为SEASLOG_ERROR 的信息列表:
array(2) {
  [0] =>
  string(66) "ERRO | 8568 | 1393172042.717 | 2014:02:24 00:14:02 | test error 3 "
  [1] =>
  string(66) "ERRO | 8594 | 1393172044.104 | 2014:02:24 00:14:04 | test error 3 "
}

同理，取当月
$detailErrorArray_mouth = SeasLog::analyzerDetail(SEASLOG_ERROR,date('Ym',time()));

*/
```

使用SeasLog进行健康预警

预警的配置

```
[base]
wait_analyz_log_path = /log/base_test

[fork]
;是否开启多线程 1开启 0关闭
fork_open = 1

;线程个数
fork_count = 3

[warning]
email[smtp_host] = smtp.163.com
email[smtp_port] = 25
email[subject_pre] = 预警邮件 -
email[smtp_user] = seaslogdemo@163.com
email[smtp_pwd] = seaslog#demo
email[mail_from] = seaslogdemo@163.com
email[mail_to] = gaochitao@weiboyi.com
email[mail_cc] = ciogao@gmail.com
email[mail_bcc] =

[analyz]
; enum
; SEASLOG_DEBUG      "debug"
; SEASLOG_INFO       "info"
; SEASLOG_NOTICE     "notice"
; SEASLOG_WARNING    "warning"
; SEASLOG_ERROR      "error"
; SEASLOG_CRITICAL   "critical"
; SEASLOG_ALERT      "alert"
; SEASLOG_EMERGENCY  "emergency"

test1[module] = test/bb
test1[level] = SEASLOG_ERROR
test1[bar] = 1
test1[mail_to] = gaochitao@weiboyi.com

test2[module] = 222
test2[level] = SEASLOG_WARNING

test3[module] = 333
test3[level] = SEASLOG_CRITICAL

test4[module] = 444
test4[level] = SEASLOG_EMERGENCY

test5[module] = 555
test5[level] = SEASLOG_DEBUG
```

crontab配置

```
;每天凌晨3点执行
0 3 * * * /path/to/php /path/to/SeasLog/Analyzer/SeasLogAnalyzer.php
```