

EasySwoole\Core\():initialize

CommandCommandCore.

```
CommandRunner run Core::getInstance()->initialize(); //
```

```
__construct //
```

```
defined('SWOOLE_VERSION') or define('SWOOLE_VERSION',intval(phpversion('swoole')));  
defined('EASYSWOOLE_ROOT') or define('EASYSWOOLE_ROOT', realpath(getcwd()));  
defined('EASYSWOOLE_SERVER') or define('EASYSWOOLE_SERVER',1);defined('EASYSWOOLE_WEB_SERVER')  
or define('EASYSWOOLE_WEB_SERVER',2);defined('EASYSWOOLE_SOCKET_SERVER') or  
define('EASYSWOOLE_SOCKET_SERVER',3);defined('EASYSWOOLE_REDIS_SERVER') or  
define('EASYSWOOLE_REDIS_SERVER',4);
```

```
initialize //,EasySwooleEvent//
```

```
//██████████  
$this->loadEnv();  
//███████████  
EasySwooleEvent::initialize();  
//█████Log████  
$this->sysDirectoryInit();  
//████████  
$this->registerErrorHandler();
```

produce.php dev.php Conf Config::getInstance()->getConf() .Config(), , , , KV

EasySwooleEvent::initialize() → initialize(), Command

```
'TEMP_DIR' => null,  
'LOG_DIR' => null
```

null,Temp\Log

```
$tempDir = Config::getInstance()->getConf('TEMP_DIR');
if(empty($tempDir)){
    $tempDir = EASYSWOOLE_ROOT.'/Temp';      Config::getInstance()->setConf('TEMP_DIR',$tempDir);
// 
}else{
    $tempDir = rtrim($tempDir,'/');
}
if(!is_dir($tempDir)){
    File::createDirectory($tempDir); // 
}
defined('EASYSWOOLE_TEMP_DIR') or define('EASYSWOOLE_TEMP_DIR',$tempDir);
```

pid.pid swoole.log

registerErrorHandler

```
ini_set("display_errors", "On");
error_reporting(E_ALL | E_STRICT);
```

```
$logger = Di::getInstance()->get(SysConst::LOGGER_HANDLER); // SysConst \LoggerInterface
if(!$logger instanceof LoggerInterface){
// 
$logger = new DefaultLogger(EASYSWOOLE_LOG_DIR);
}
Logger::getInstance($logger); // \Logger
```

Logger log console \LoggerInterface

```
namespace EasySwoole\Log;
interface LoggerInterface
{
    const LOG_LEVEL_INFO = 1;
    const LOG_LEVEL_NOTICE = 2;
    const LOG_LEVEL_WARNING = 3;
```

```
const LOG_LEVEL_ERROR = 4;

function log(?string $msg,int $logLevel = self::LOG_LEVEL_INFO,string $category =
'DEBUG'):string ;      function console(?string $msg,int $logLevel = self::LOG_LEVEL_INFO,string
$category = 'DEBUG');

}
```

log file_put_contents ████

```
function log(?string $msg,int $logLevel = self::LOG_LEVEL_INFO,string $category = 'DEBUG'):string
{
    $date = date('Y-m-d H:i:s');
    $levelStr = $this->levelMap($logLevel);
    $filePath = $this->logDir."/log.log";      $str = "[{$date}][{$category}][{$levelStr}] :
[{$msg}]\n";
    file_put_contents($filePath,"{$str}",FILE_APPEND|LOCK_EX);
    return $str;
}
```

console ████

```
function console(?string $msg,int $logLevel = self::LOG_LEVEL_INFO,string $category = 'DEBUG')
{
    $date = date('Y-m-d H:i:s');
    $levelStr = $this->levelMap($logLevel);      $temp =  $this-
>colorString("[{$date}][{$category}][{$levelStr}] : [{$msg}]",$logLevel)."\n";
    fwrite(STDOUT,$temp);
}
```

[EasySwooleEvent] initialize Di::getInstance()->set(SysConst::LOGGER_HANDLER, ████) ████

Trigger

```
//████
$trigger = Di::getInstance()->get(SysConst::TRIGGER_HANDLER);
// █████████TriggerInterface█
if(!$trigger instanceof TriggerInterface){
```

```
// 旣存のTriggerを登録する
$trigger = new DefaultTrigger(Logger::getInstance());
}

Trigger::getInstance($trigger); // 既存Triggerを登録
```

```
[ set_error_handler ] [ register_shutdown_function ] [ ]
```

`set_error_handler` [] []. [] [].

register_shutdown_function

```
//trigger

$errorHandler = Di::getInstance()->get(SysConst::ERROR_HANDLER);//错误处理器
if(!is_callable($errorHandler)){    $errorHandler = function($errorCode, $description, $file =
null, $line = null){
    $l = new Location();
    $l->setFile($file);
    $l->setLine($line);
    // Trigger: 错误处理器
    Trigger::getInstance()->error($description,$errorCode,$l);
};

}

// 错误处理器
set_error_handler($errorHandler);

$func = Di::getInstance()->get(SysConst::SHUTDOWN_FUNCTION);
if(!is_callable($func)){
    $func = function (){
        // 错误处理器
        $error = error_get_last();
        if(!empty($error)){
            $l = new Location();
            $l->setFile($error['file']);
            $l->setLine($error['line']);
            // Trigger: 错误处理器
            Trigger::getInstance()-
>error($error['message'],$error['type'],$l);
        }
    };
}
```

```
}

// ログを出力する関数を登録する

register_shutdown_function($func);
```

Location [line] [file].ログを出力する関数を登録する.

ログを出力する関数を登録するTriggerを登録する。これで、ログを出力するTrigger.

Revision #2

Created Sun, Dec 22, 2019 11:35 AM by 

Updated Tue, Jan 7, 2020 1:37 PM by 