Atitit 脚本语言调试方法总结

目录

[1.1. 提供跟踪执行信息，将执行的每一条命令和结果依次打印出来 1](#_Toc11936)

[1.2. 每一个结果和命令--------- 方法（参数） 1](#_Toc5425)

[1.3. 基础调试法/打印变量值 1](#_Toc14002)

[1.4. 基础调试法/打印脚本运行时间 2](#_Toc4865)

[1.5. 基础调试法/打印脚本运行时间 2](#_Toc14931)

[1.6. 基础调试法/堆栈追踪(Stack Trace) 2](#_Toc21387)

[1.7. 基础调试法/打印到文件 3](#_Toc20102)

[2. 关于打Log和单步调试这两种调试手段 3](#_Toc27935)

[2.1. 输出到屏幕 输出到文件 4](#_Toc30825)

[3. Bug复现 4](#_Toc8257)

-v

一边执行脚本，一边将执行过的脚本命令打印到标准错误输出

-x

## 提供跟踪执行信息，将执行的每一条命令和结果依次打印出来

日志

## 每一个结果和命令--------- 方法（参数）

## 基础调试法/打印变量值

php有几种非常简便的调试,相信大多数人都是知道的,echo,print,print\_r,var\_dump,var\_export  
这几种都是php自带的调试方法,非常常用,为了便于理解,所以做成了表格



## 基础调试法/打印脚本运行时间

php的执行速度是非常快速的,往往执行一个脚本的时间不到一秒中,所以如果我们想测试一下执行时间的话,用时间戳就不行了,就要借助一个函数microtime它会返回当前时间戳和微秒数,但是他的格式并不是很方便,我们需要写一个函数方便调试

function get\_microtime() {

list( $usec, $sec) = explode(" ", microtime());

return ((float)$usec + (float)$sec); }

然后在开始出调用一次,再在结束前调用此,最后用开始的时间减去结束的时间便可以拿到脚本运行时间

## 基础调试法/打印脚本运行时间

PHP提供了一个函数可以直接打印出PHP占用的内存量,memory\_get\_usage,但是返回的单位是字节(byte),这非常不方便我们的调试,于是我们需要一个函数转化单位

function convert($size){

$unit=array('b','kb','mb','gb','tb','pb');

return @round($size/pow(1024,($i=floor(log($size,1024)))),2).' '.$unit[$i];}

## 基础调试法/堆栈追踪(Stack Trace)

PHP提供可以打印堆栈的函数,所谓(堆栈追踪/堆栈打印)就是打印出程序运行到异常时所加载的文件和执行的函数,有了它我们可以很方便的确定是程序在哪里出了问题(特别是在自己不熟悉的项目内)  
debug\_print\_backtrace(),debug\_backtrace()  
还可以直接调用异常类中的函数

$e = new Exception; var\_dump($e->getTraceAsString());

当然,,大多数框架都会为你集成这个功能,但是掌握此函数会让你更加灵活的利用堆栈去熟悉,调试代码

## 基础调试法/打印到文件

在很多场景下,比如第三方服务的回调,又或者线上调试时,是不能直接打印出来的,所以我们需要一个能让我们确定程序是否正确运行的但是又不直接打印到页面上的函数

function mlog(){

$args = func\_get\_args();

foreach ($args as $arg){

file\_put\_contents('debug.txt', var\_export($arg,true)."\n",FILE\_APPEND);

}

}

借助于此函数能把变量写到根目录下的debug,txt里面,从而让我们确定程序的执行情况

# 关于打Log和单步调试这两种调试手段

谁更好，可以参见：[打印日志 (log) 是比单步跟踪 (debugger) 更好的 Python 排错手段吗？](https://www.zhihu.com/question/20626825)

## 输出到屏幕 输出到文件

# Bug复现

## 多线程环境调用单元测试

## 增加sleep，模拟压力大时延迟增加

有时候bug自然触发的概率很低，需要用一些人为的手段来帮助触发（比如故意在某个原本比较快的过程中增加sleep，模拟压力大时延迟增加的现象），需要增加一些额外的代码