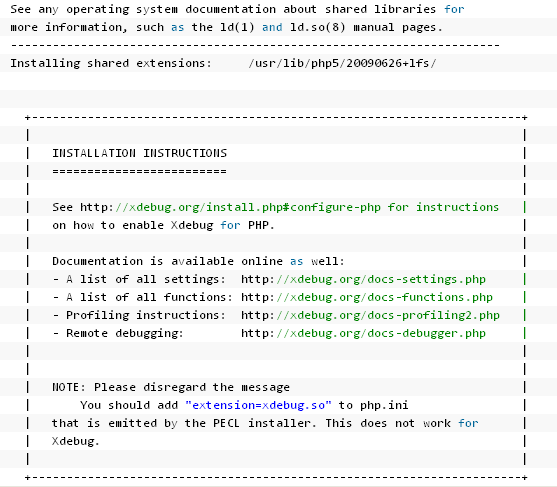
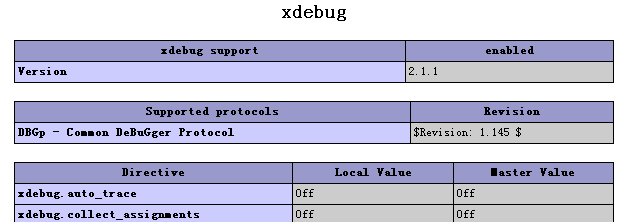
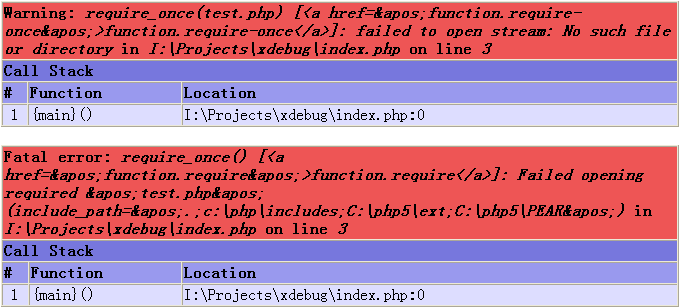
为什么需要Debugger?  
很多PHP程序员调试使用echo、print\_r()、var\_dump()、printf()等，其实对 于有较丰富开发经验的程序员来说这些也已经足够了，他们往往可以在程序执行的过程中，通过输出特定变量的值可以判断程序执行是否正确，甚至效率高低也可以 看出来（当然可能还需要使用一些时间函数）。那么我们为什么还需要一个专门的调试程序来监控我们的程序运行呢？ 这个问题的答案不妨留到后面来揭晓。  
什么是Xdebug?  
Xdebug是一个开放源代码的PHP程序调试器(即一个Debug工具)，可以用来跟踪，调试和分析PHP程序的运行状况。  
**如何安装Xdebug？：**1． 打开http://www.xdebug.org/download.php下载相应的版本  
Win:Windows binaries版本  
  Linux:source  
得到一个dll文件(win)或运行安装文件(linux)  
2． 安装  
Win：将下载的dll文件放到相应的目录中。比如我的就放D:\xampp\php\ext下面；  
**Linux:执行安装文件**tar -xvzf xdebug-2.1.2.tgz  
cd  xdebug-2.1.2  
phpize (如果phpize没有这个command,需要安装一次phpize。phpize可以让php支持扩展模块)安装phpize:sudo apt-get install php5-dev  
如果安装了继续进行下面命令  
./configure  
make  
make install  
**会有这个界面**

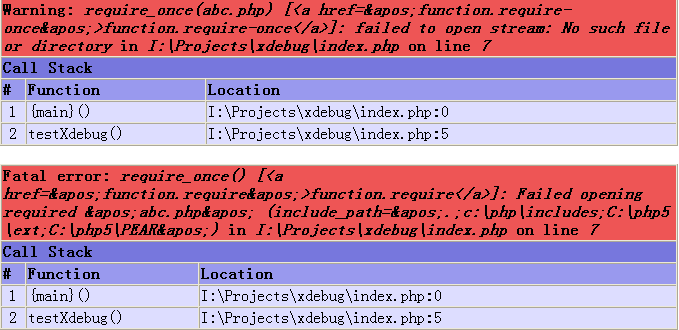
  
  
cp modules/xdebug.so /usr/lib/php5/20090626+lfs 将xdebug.so文件移到php5下面  
3． 编辑php.ini，加入下面几行：  
 [Xdebug]  
zend\_extension=D:\xampp\php\ext\php\_xdebug.dll   (Win)  
zend\_extension=/usr/lib/php5/20090626+lfs/xdebug.so（Linux）  
  
xdebug.profiler\_enable=on  
xdebug.trace\_output\_dir="../Projects/xdebug"  
xdebug.profiler\_output\_dir="../Projects/xdebug"  
后面的目录“../Projects/xdebug”为你想要放置Xdebug输出的数据文件的目录，可自由设置。  
4． 重启Apache；  
5． 写一个test.php，内容为<?php phpinfo(); ?>，如果输出的内容中有看到xdebug，说明安装配置成功。**如下图：**



现在我们来从最简单的程序调试开始一步步介绍Xdebug。  
**调试：**我们先写一个可以导致执行出错的程序，例如尝试包含一个不存在的文件。  
**testXdebug.php**<?php  
require\_once(‘abc.php');  
?>  
然后通过浏览器访问，我们惊奇地发现，出错信息变成了彩色的了：

[](http://files.jb51.net/file_images/article/201306/201306200852134.png)

不过除了样式改变，和我们平时打印的出错信息内容没什么不同，意义不大。好，我们继续改写程序：**testXdebug2.php**<?php  
testXdebug();  
function testXdebug() {  
       require\_once('abc.php');  
}  
?>  
输出信息：

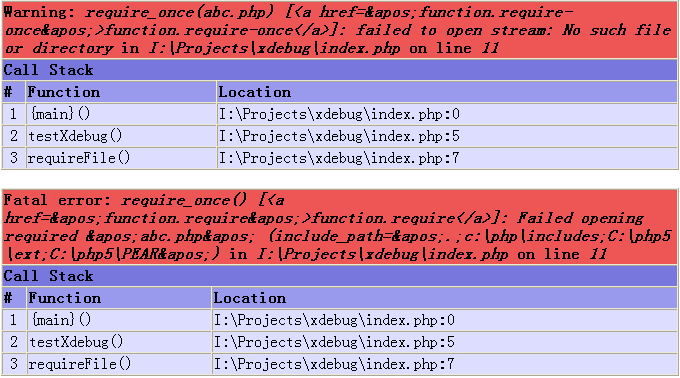
[](http://files.jb51.net/file_images/article/201306/201306200852135.png)

发现了什么？　Xdebug跟踪代码的执行，找到了出错的函数testXdebug()。  
我们把代码再写得复杂一些：　  
**testXdebug3.php**

复制代码 代码如下:

<?php  
testXdebug();  
function testXdebug() {  
       requireFile();      
}  
function requireFile() {  
       require\_once('abc.php');  
}  
?>

输出信息：

[](http://files.jb51.net/file_images/article/201306/201306200852137.png)

也就是说Xdebug具有类似于Java的Exception的“跟踪回溯”的功能，可以根据程序的执行一步步跟踪到出错的具体位置，哪怕程序中的调用很复杂，我们也可以通过这个功能来理清代码关系，迅速定位，快速排错。

其实PHP函数debug\_backtrace()也有类似的功能，但是要注意debug\_backtrace()函数只在PHP4.3.0之后版本及PHP5中才生效。这个函数是PHP开发团队在PHP5中新增的函数，然后又反向移植到PHP4.3中。

**如何利用Xdebug测试脚本执行时间**测试某段脚本的执行时间，通常我们都需要用到microtime()函数来确定当前时间。例如PHP手册上的例子：

复制代码 代码如下:

<?php  
/\*\*  
\* Simple function to replicate PHP 5 behaviour  
\*/  
function microtime\_float()  
{  
      list($usec, $sec) = explode(" ", microtime());  
return ((float)$usec + (float)$sec);  
}  
$time\_start = microtime\_float();  
// Sleep for a while  
usleep(100);  
$time\_end = microtime\_float();  
$time = $time\_end - $time\_start;  
echo "Did nothing in $time seconds\n";  
?>

但是microtime()返回的值是微秒数及绝对时间戳（例如“0.03520000 1153122275”），没有可读性。所以如上程序，我们需要另外写一个函数microtime\_float()，来将两者相加。  
Xdebug自带了一个函数xdebug\_time\_index()来显示时间。  
**如何测定脚本占用的内存？**  
有时候我们想知道程序执行到某个特定阶段时到底占用了多大内存,为此PHP提供了函数memory\_get\_usage()。这个函数只有当PHP编译时使用了-enable-memory-limit参数时才有效。　  
  
Xdebug同样提供了一个函数xdebug\_memory\_usage()来实现这样的功能，另外xdebug还提供了一个xdebug\_peak\_memory\_usage()函数来查看内存占用的峰值。

**如何检测代码中的不足？**

有时候代码没有明显的编写错误，没有显示任何错误信息（如error、warning、notice等），但是这不表明代码就是正确无误的。有时候可能某段代码执行时间过长，占用内存过多以致于影响整个系统的效率，我们没有办法直接看出来是哪部份代码出了问题。这时候我们希望把代码的每个阶段的运行情况都监控起来，写到日志文件中去，运行一段时间后再进行分析，找到问题所在。  
回忆一下，之前我们编辑php.ini文件  
**加入**[Xdebug]  
xdebug.profiler\_enable=on  
xdebug.trace\_output\_dir="I:\Projects\xdebug"  
xdebug.profiler\_output\_dir="I:\Projects\xdebug"

这几行，目的就在于把执行情况的分析文件写入到”../Projects/xdebug”目录中去（你可以替换成任何你想设定的目录）。如果你执行某段程序后，再打开相应的目录，可以发现生成了一堆文件，例如cachegrind.out.1169585776这种格式命名的文件。这些就是Xdebug生成的分析文件。用编辑器打开你可以看到很多程序运行的相关细节信息，

**最后：**

Xdebug提供了各种自带的函数，并对已有的某些PHP函数进行覆写，可以方便地用于调试排错；Xdebug还可以跟踪程序的运行，通过对日志文件的分析，我们可以迅速找到程序运行的瓶颈所在，提高程序效率，从而提高整个系统的性能。