**PHP OB-缓冲区**

什么是缓冲区?  
简单而言,缓冲区的作用就是,把输入或者输出的内容先放进内存,而不显示或者读取.至于为什么要有缓冲区,这是一个很广泛的问题,如果有兴趣,可以在网上找下资料.  
其实**缓冲区最本质的作用就是,协调高速CPU和相对缓慢的IO设备(磁盘等)的运作.**

PHP在执行的时候,在什么地方有用到缓冲区?  
想要了解PHP的缓冲区,就要知道执行PHP的时候,缓冲区被设置到了什么地方.  
当执行PHP的时候,如果碰到了echo print\_r之类的会输出数据的代码,PHP就会将要输出的数据放到PHP自身的缓冲区,等待输出.  
当PHP自身的缓冲区接到指令,指示要输出缓冲区的内容时,将会把缓冲区内的数据输出到apache上, apache接受到PHP输出的数据,然后再把该数据存在到apache自身的缓冲区内,等到输出  
当apache接受到指令,只是要输出缓冲区的内容时, 将会把缓冲区的内容输出,返回到浏览器.

由此可见,PHP要输出数据的时候,将会经过两个缓冲区(先是自身的,然后是apache的),再返回到浏览器.

缓冲区在PHP中起到什么作用?**【场景】**  
1.最常见的就是在使用header函数之前,就已经输出了某些数据,这样会导致某些错误,

例如 Cannot modify header information – headers already sent by;

|  |  |
| --- | --- |
| 1  2 | echo  "this is test" ;  header( "LOCATION [http://www.baidu.com](http://www.baidu.com/)" ); |

出现这个错误的原因是, 在header之前已经输出了某些数据,而输出这些数据的同时, apache将会同时发送一个响应状态到浏览器上(既然有输出,即这个请求是有效的),而其后你又再次使用header函数  
发送http头,则会返回这个错误,错误的意思是:HTTP头已经发送出去了,你不能对他再做修改.  
为什么使用缓冲区可以避免这个错误呢?  
因为header函数是不受缓冲区影响的,当一碰到header函数的时候,PHP马上执行apache发送这一个http头都浏览器.  
而输出的数据PHP打开输出缓冲区后, 这些数据将会存放在缓冲区,等待输出.这样就可以避免了之前所发生的错误.  
2.通过PHP写文件下载程序的时候.  
为了让文件下载更安全,同时提高更多的可控性,很多朋友都喜欢用PHP写文件下载页面.其原理很简单,就是通过fwrite把文件内容读出并显示,然后通过header来发送HTTP头,让浏览器知道这是一个附件,这样  
就可以达到提供下载的效果.  
如果用上面的办法提供下载页面,会碰到一个效率问题,如果一个文件很大,假设为100M,那么在不开启缓冲区输出的情况下,必须要把100M数据全部读出,然后一次返回到页面上,如果这样做,用户将会在所有数据读完  
之后才会得到响应,降低了用户体验感.  
如果开启了输出缓冲区,当PHP程序读完文件的某一段,然后马上输出到apache,然后让apache马上返回到浏览器,这样就可以减少用户等待时间.那后面的数据怎么办呢?我们可以写一个while循环,一直一段一段地读取文件  
每读一段,就马上输出,直到把文件全部输出为止,这样浏览器就可以持续地接受到数据,而不必等到所有文件读取完毕.

另外,该做法还解决了另外一个很严重的问题.例如一个文件是100M,如果不开启缓冲区的情况下,则需要把100M文件全部读入内存,然后再输出.但是,如果PHP程序做了内存限制呢?为了保证服务器的稳定,管理员通常会把PHP的执行  
内存设一个限制(通过php.ini总的memory\_limit, 其默认值是8M), 也就是每个PHP程序使用的内存不能使用超过这个值的内存. 假设该值为8M,而要读入的文件是100M,根本就没有足够的内存来读入该文件.这个时候,我们就需要用到上面的  
办法来解决这个问题,每次只读某一段,这样就可以避免了内存的限制  
3.静态文件缓存  
现在很多公司有这么一个需求, 就是某一个页面在第一次访问的时候,会执行PHP,然后把显示的内容返回到浏览器,同时需要把这次显示的内容保存到服务器上,这样下次访问的时候,就直接把保存在服务器上的文件直接显示,而不需要通过PHP来做操作  
这就是所谓的”静态页面缓存”.那怎么样才能做到把内容返回到浏览器的同时把数据保存到服务器上呢?这就要用到输出缓冲区了.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | ob\_start();  echo  'aaa' ;  $string  = ob\_get\_contents();  file\_put\_contents ( 'a.html' ,  $string );  ob\_flush();  flush (); |

与输出缓冲区有关的配置  
在PHP.INI中,有两个跟缓冲区紧密相关的配置项  
1.output\_buffering  
该配置直接影响的是php本身的缓冲区,有3种配置参数.on/off/xK(x为某个整型数值);  
on    - 开启缓冲区  
off    - 关闭缓冲区  
256k    - 开启缓冲区,而且当缓冲区的内容超过256k的时候,自动刷新缓冲区(把数据发送到apache);

2.implicit\_flush  
该配置直接影响apache的缓冲区,有2种配置参数. on/off  
on    - 自动刷新apache缓冲区,也就是,当php发送数据到apache的缓冲区的时候,不需要等待其他指令,直接就把输出返回到浏览器  
off    - 不自动刷新apache缓冲区,接受到数据后,等待刷新指令  
  
与缓冲区有关的函数  
1.ob\_implicit\_flush  
作用和implicit\_flush一样,是否自动刷新apache的缓冲区  
2.flush  
作用是发送指令到apache,让apache刷新自身的输出缓冲区.  
3.ob\_start  
打开输出缓冲区,无论php.ini的文件如何配置,如果使用该函数,即使output\_buffering设置成off,也会打开输出缓冲区  
ob\_start函数还接受一个参数,该参数是一个函数的回调,意思是,在输入缓冲区内容之前,需要使用调用传递进来的参数把缓冲区的内容处理一次,再放入缓冲区内  
4.ob\_flush  
指示php本身刷新自身的缓冲区,把数据发送到apache  
5.ob\_clean  
清除php缓冲区里面的内容  
6.ob\_end\_clean  
清除php缓冲区内的内容,并且关闭输出缓冲区  
7.ob\_end\_flush  
把php自身的缓冲区里的内容发送到apache,并把清除自身缓冲区内的内容  
8.ob\_get\_clean  
获取缓冲区的内容之后,清除缓冲区.  
9.ob\_get\_contents  
获取输出缓冲区里的内容  
10.ob\_get\_flush  
获取缓冲区里的内容,并且把这些内容发送到apache  
11.ob\_get\_length  
获取缓冲区里内容的长度  
12.ob\_list\_handlers  
获取运行ob\_start时,所回调的函数名称, 例如:  
ob\_start(‘ob\_gzhandler’);  
print\_r(ob\_list\_handlers);  
将打印出ob\_gzhandler;  
13.ob\_gzhandler  
该函数的作用是作为ob\_start的回调参数, 在缓冲区刷新之前,会调用该函数对数据进行到底gzip或者deflate压缩.这个函数需要zlib扩展的支持.

**使用缓冲区的相关内容**：  
1.ob\_flush和flush的次序关系.上面的分析可以看出,ob\_flush是和php自身相关的,而flush操作的是apache的缓冲区,所有我们在使用这两个函数的时候,需要先执行ob\_flush,  
再执行flush,因为我们需要先把数据从PHP上发送到apache,然后再由apache返回到浏览器.如果php还没有把数据刷新到apache,就调用了flush,则apache无任何数据返回到浏览器.

2.有的浏览器,如果接受到的字符太少,则不会把数据显示出来,例如老版的IE(必须要大于256k才显示).这样就会造成一个疑问, 明明在php和apache都进行了刷新缓冲区的操作,但是浏览器就是没有出现自己想要的数据,也许就是这个原因造成的.所以才测试的时候,可以在输出数据的后面加上多个空格,以填满数据,确定不会浏览器造成这类诡异的问题.

3.有些webserver,他自身的输出缓冲区会有一些限制,比如nginx,他有一个配置fastcgi\_buffer\_size 4k, 就是是表明,当自身的输出缓冲区的内容达到4K才会刷新,所以为了保证内容的数据,可以添加以下代码,保证内容长度

|  |  |
| --- | --- |
| 1  2  3 | <?php  echo  str\_repeat ( " " ,4096);  ?> |

4.在apache中,如果你开启了mod\_gzip的压缩模块,这样可能会导致你的flush函数刷新不成功,其原因是,mod\_gzip有自己的输出缓冲区,当php执行了flush函数,指示apache刷新输出缓冲区,但是内容需要压缩,apache就把内容输出到自身的mod\_gzip模块,mod\_gzip也有自身的输出 缓冲区,他也不会马上输出,所以造成了内容不能马上输出.为了改善这个情况,可以关闭mod\_gzip模块,或者在httpd.conf增加以下内容,以禁止压缩

|  |  |
| --- | --- |
| 1 | SetEnv no-gzip dont-vary |