**php中ob函数缓冲机制深入理解**

这篇文章主要介绍了php中ob函数缓冲机制,有需要的小伙伴们可以来学习下

下面就php中ob函数缓冲机制通过文字说明加代码分析的形式给大家展示如下：

对于一个刚刚入门的php程序员来说，php缓冲区是几乎透明的。在他们心目中，一个**echo print\_r 函数**，数据便会‘嗖'的一声飞到浏览器上，显示出来。我也一直如此单纯地认为。 其实，在技术的世界里，向来都是由简单到复杂，也许那些技术开发者开始单纯如你我，但是面对残酷的现实，不得不调整策略，以期提高机器运行效率，最后想到了那些让我们赞叹的idea。

**说到缓冲，也就是buffer,这里必须要和缓存做一下比较，单纯地比较定义是无意义的，莫不如看看它们做什么。缓存解决的是如何快速查找利用数据，节省cpu消耗问题，而缓冲解决的是高速cpu与低速I/O设备不匹配的问题。**

　　再说下本文的另一个主角，ob函数，ob是**output\_buffering**的简写。既然ob函数是php扩展函数，那么ob函数主要操作的也就是**php buffer**了。

　　简单说完本文两个主角，我们还必须回归开头的主题，**echo print\_r**函数输出的数据是怎么到达浏览器让用户看到的呢？实际上的历程是这样的：

复制代码代码如下:  
echo、print\_r=>php output\_buffering=>webServer buffer=>browser buffer=>browser display

　　我们可以清楚地看到，从**echo、print\_r**函数到发送信息给客户端经历了两个缓冲区，在客户端还经历了一个浏览器缓冲区。我们本文主要讨论的是**php output\_buffering**。

　　未使用ob函数时缓冲区的使用情况

 　　我们的代码很多时候是根本不使用ob函数的，那么它们使用缓冲区了吗？这要看php设置情况。缓冲区是通过php.ini中的output\_buffering变量控制的。其默认值是off,可以设置为on来打开buffer。打来buffer后，即便程序中没有用ob函数，实际上代码也是使用了缓冲区的。另外，不管**php.ini中output\_buffering**的设置，cli模式下的php始终默认是关闭的。

　　为什么要是缓冲区呢？简单来说，高速的cpu早早处理完自己的数据，想通过线路传递给用户，但是线路太窄了，一下输送不过去。如果引入缓冲区，cpu可以将快速将生成的数据放入缓冲区，然后自己哪儿凉快儿哪儿呆着这歇着去了。缓冲区根据指令适时将数据输出。这个样就合理解决了高速cpu与低速I/O设备的矛盾了。

　　缓冲区的数据什么时候输出呢？1，当缓冲区满了的时候，缓冲是有容量大小的，到达极限则会自动输出内容。2，脚本执行完毕。很多小程序输出内容没那么多，总不能等到缓冲区满了再输出吧~这一点再自然不过。

**使用ob函数时缓冲区的使用情况**

复制代码代码如下:  
　ob\_start()

　　打开输出缓冲。这个函数是我们调用最多的一个函数之一。在output\_buffering设置为on或者x k的情况下，这个函数与其说是打开输出缓冲，还不如说将输出缓冲扩充到很大。当然在output\_buffering设置为off的条件下，ob\_start会起到打开buffer的作用。ob\_start()还可以传递一个可选参数 output\_callback 函数,php官方手册有详细说明。

复制代码代码如下:  
ob\_get\_contents()

　　只是得到输出缓冲区的内容，但不清除它。

复制代码代码如下:  
ob\_end\_clean()与ob\_clean()

　　这两个函数从字面意思上就可以看出其区别。前者清除缓冲区内容并且关闭，后者仅仅是做清除工作。需要注意的是，使用了这两个函数，在前面使用了echo、print\_r等函数不会输出内容。

　　笔者曾经试图通过print\_r打印出**ob\_get\_contents()**的内容，然后调用**ob\_clean()**清除缓冲区，以免影响后面对缓冲区的操作，屡屡失败。仔细想想，print\_r的内容再次写入缓冲区，而后面做了**ob\_clean()**的操作，自然不会有任何输出。在ob\_clean操作之前调用ob\_flush()函数便可达到预想的效果。

复制代码代码如下:  
ob\_flush()与flush()

　　ob\_flush()送出缓冲区的内容并且丢弃内容。因而在此函数之前最好采用ob\_get\_contents()获得缓冲区内容。flush()刷出服务器端缓冲，并且发往客户端。因而从流程上来说，应该是先调用ob\_flush()而后再调用flush函数。

　　另外说明下再**Apache buffer flush()**的工作原理：在**apache module**的sapi下，flush会通过调用**sapi\_module()**的flush成员函数指针，间接使用apache的api：：ap\_rflush刷新apache的输出缓冲区。当然apache其他模块比如**mod\_gzip**可能改变这个动作的结果，可能自己进行输出缓冲区，这将导致flush()函数产生的结果不会立即被送到客户端浏览器。

复制代码代码如下:  
ob\_get\_clean()

　　如果你已经熟练掌握**ob\_get\_contents()**和**ob\_clean()，**那这个函数就很简单了。因为它是前两者的结合体。它主要是得到当前缓冲区的内容并删除当前输出缓冲区。

　　ob函数还有很多，但大部分用法比较简单，理解较为容易。大家可以参照php手册 ，里面会有详细的解释。本文列出了笔者开始并不是很理解的一些函数，当然今后还会有新的问题出现，想到问题并且解决问题，生活的乐趣也许就在此处吧。

以上内容就是本文针对php中ob函数缓冲机制深入理解，希望对大家今后学习有所帮助。