# **[PHP运行模式](http://www.cnblogs.com/xia520pi/p/3914964.html)**

## **1、运行模式**

关于PHP目前比较常见的五大运行模式：

1）**CGI**（通用网关接口 / Common Gateway Interface）

2）**FastCGI**（常驻型CGI / Long-Live CGI）

3）**CLI**（命令行运行 / Command Line Interface）

4）**Web模块模式**（Apache等Web服务器运行的模式）

5）**ISAPI**（Internet Server Application Program Interface）

备注：在PHP5.3以后，PHP不再有ISAPI模式，安装后也不再有php5isapi.dll这个文件。要在IIS6上使用高版本PHP，必须安装FastCGI 扩展，然后使IIS6支持FastCGI。

### **1.1 CGI模式**

**CGI**即通用网关接口（Common Gateway Interface），它是一段程序，通俗的讲CGI就象是一座桥，把网页和Web服务器中的执行程序连接起来，它把HTML接收的指令传递给服务器的执行程序，再把服务器执行程序的结果返还给HTML页。CGI 的跨平台性能极佳，几乎可以在任何操作系统上实现。 **CGI已经是比较老的模式了，这几年都很少用了。**

　　每有一个用户请求，都会先要创建CGI的子进程，然后处理请求，处理完后结束这个子进程，这就是Fork-And-Execute模式。 当用户请求数量非常多时，会大量挤占系统的资源如内存，CPU时间等，造成效能低下。所以用CGI方式的服务器有多少连接请求就会有多少CGI子进程，子进程反复加载是CGI性能低下的主要原因。

　　如果不想把 PHP 嵌入到服务器端软件（如 Apache）作为一个模块安装的话，可以选择以 CGI 的模式安装。或者把 PHP 用于不同的 CGI 封装以便为代码创建安全的 chroot 和 setuid 环境。这样每个客户机请求一个PHP文件，Web服务器就调用php.exe（win下是php.exe,linux是php）去解释这个文件，然后再把解释的结果以网页的形式返回给客户机。 这种安装方式通常会把 PHP 的可执行文件安装到 web 服务器的 cgi-bin 目录。CERT 建议书 CA-96.11 建议不要把任何的解释器放到 cgi-bin 目录。

　　这种方式的好处是把Web Server和具体的程序处理独立开来，结构清晰，可控性强，同时缺点就是如果在高访问需求的情况下，CGI的进程Fork就会成为很大的服务器负担，想 象一下数百个并发请求导致服务器Fork出数百个进程就明白了。这也是为什么CGI一直背负性能低下，高资源消耗的恶名的原因。

### **1.2 FastCGI模式**

　　FastCGI是CGI的升级版本，FastCGI像是一个常驻 (long-live)型的 CGI，它可以一直执行着，只要激活后，不会每次都要花费时间去 Fork 一次 （这是 CGI 最为人诟病的 fork-and-execute 模式）。

　　FastCGI是一个可伸缩地、高速地在HTTP server和动态脚本语言间通信的接口。多数流行的HTTP server都支持FastCGI，包括Apache、Nginx和lighttpd等，同时，FastCGI也被许多脚本语言所支持，其中就有PHP。

　　FastCGI接口方式采用C/S结构，可以将HTTP服务器和脚本解析服务器分开，同时在脚本解析服务器上启动一个或者多个脚本解析守护进程。当HTTP服务器每次遇到动态程序时，可以将其直接交付给FastCGI进程来执行，然后将得到的结果返回给浏览器。这种方式可以让HTTP服务器专一地处理静态请求或者将动态脚本服务器的结果返回给客户端，这在很大程度上提高了整个应用系统的性能。

**【原理】**

1）Web Server启动时载入FastCGI进程管理器（IIS ISAPI或Apache Module)；

2）FastCGI进程管理器自身初始化，启动多个CGI解释器进程 (可见多个php-cgi.exe或php-cig)并等待来自Web Server的连接；

3）当客户端请求到达Web Server时，FastCGI进程管理器选择并连接到一个CGI解释器。Web server将CGI环境变量和标准输入发送到FastCGI子进程php-cgi；

4）FastCGI子进程完成处理后将标准输出和错误信息从同一连接返回Web Server。当FastCGI子进程关闭连接时，请求便告处理完成。FastCGI子进程接着等待并处理来自FastCGI进程管理器（运行在 WebServer中）的下一个连接。在正常的CGI模式中，php-cgi.exe在此便退出了。

　　在CGI模式中，你可以想象 CGI通常有多慢。每一个Web请求PHP都必须重新解析php.ini、重新载入全部dll扩展并重初始化全部数据结构。使用FastCGI，所有这些都只在进程启动时发生一次。一个额外的好处是，持续数据库连接（Persistent database connection）可以工作。

**备注**：PHP的FastCGI进程管理器是**PHP-FPM（PHP-FastCGI Process Manager）**

**【优点】**

1）从稳定性上看，FastCGI是以独立的进程池来运行CGI，单独一个进程死掉，系统可以很轻易的丢弃，然后重新分配新的进程来运行逻辑；

2）从安全性上看，FastCGI支持分布式运算。FastCGI和宿主的Server完全独立，FastCGI怎么down也不会把Server搞垮；

3）从性能上看，FastCGI把动态逻辑的处理从Server中分离出来，大负荷的IO处理还是留给宿主Server，这样宿主Server可以一心一意作IO，对于一个普通的动态网页来说, 逻辑处理可能只有一小部分，大量的是图片等静态。

**【缺点】**

　　说完了好处，也来说说缺点。从我的实际使用来看，用FastCGI模式更适合生产环境的服务器。但对于开发用机器来说就不太合适。因为当使用 Zend Studio调试程序时，由于 FastCGI会认为 PHP进程超时，从而在页面返回 500错误。这一点让人非常恼火，所以我在开发机器上还是换回了 ISAPI模式。对某些服务器的新版本支持不好，对分布式负载均衡没要求的模块化安装是否是更好的选择。目前的FastCGI和Server沟通还不够智能，一个FastCGI进程如果执行时间过长会被当成是死进程杀掉重起，这样在处理长时间任务的时候很麻烦，这样做也使得FastCGI无法允许联机调试。因为是多进程，所以比CGI多线程消耗更多的服务器内存，PHP-CGI解释器每进程消耗7至25兆内存，将这个数字乘以50或100就是很大的内存数。

### **1.3 CLI模式**

　　PHP-CLI是PHP Command Line Interface的简称，如同它名字的意思，就是PHP在命令行运行的接口，区别于在Web服务器上运行的PHP环境（PHP-CGI，ISAPI等）。 也就是说，PHP不单可以写前台网页，它还可以用来写后台的程序。 PHP的CLI Shell脚本适用于所有的PHP优势，使创建要么支持脚本或系统甚至与GUI应用程序的服务端，在Windows和Linux下都是支持PHP-CLI模式的。

**【优点】**

1）使用多进程，子进程结束以后，内核会负责回收资源；

2）使用多进程，子进程异常退出不会导致整个进程Thread退出，父进程还有机会重建流程；

3）一个常驻主进程，只负责任务分发，逻辑更清楚。

　　我们在Linux下经常使用"php –m"查找PHP安装了那些扩展就是PHP命令行运行模式；有兴趣的同学可以输入"php –h"去深入研究该运行模式。

### **1.4 模块模式**

**模块模式**是以mod\_php5模块的形式集成，此时mod\_php5模块的作用是接收Apache传递过来的PHP文件请求，并处理这些请求，然后将处理后的结果返回给Apache。如果我们在Apache启动前在其配置文件中配置好了PHP模块（mod\_php5）， PHP模块通过注册apache2的ap\_hook\_post\_config挂钩，在Apache启动的时候启动此模块以接受PHP文件的请求。

　　除了这种启动时的加载方式，Apache的模块可以在运行的时候动态装载，这意味着对服务器可以进行功能扩展而不需要重新对源代码进行编译，甚至根本不需要停止服务器。我们所需要做的仅仅是给服务器发送信号HUP或者AP\_SIG\_GRACEFUL通知服务器重新载入模块。但是在动态加载之前，我们需要将模块编译成为动态链接库。此时的动态加载就是加载动态链接库。 Apache中对动态链接库的处理是通过模块mod\_so来完成的，因此mod\_so模块不能被动态加载，它只能被静态编译进Apache的核心。这意味着它是随着Apache一起启动的。

　　Apache是如何加载模块的呢？我们以前面提到的mod\_php5模块为例。首先我们需要在Apache的配置文件httpd.conf中添加一行：

LoadModule php5\_module modules/mod\_php5.so

　　这里我们使用了LoadModule命令，该命令的第一个参数是模块的名称，名称可以在模块实现的源码中找到。第二个选项是该模块所处的路径。如果需要在服务器运行时加载模块，可以通过发送信号HUP或者AP\_SIG\_GRACEFUL给服务器，一旦接受到该信号，Apache将重新装载模块，而不需要重新启动服务器。

　　该运行模式是我们以前在windows环境下使用apache服务器经常使用的，而在模块化（DLL）中，PHP是与Web服务器一起启动并运行的。（它是apache在CGI的基础上进行的一种扩展，加快PHP的运行效率）。

### **1.5 ISAPI模式**

**ISAPI**（Internet Server Application Program Interface）是微软提供的一套面向Internet服务的API接口，一个ISAPI的DLL，可以在被用户请求激活后长驻内存，等待用户的另一个请求，还可以在一个DLL里设置多个用户请求处理函数，此外，ISAPI的DLL应用程序和WWW服务器处于同一个进程中，效率要显著高于CGI。（由于微软的排他性，只能运行于windows环境）

　　PHP作为Apache模块，Apache服务器在系统启动后，预先生成多个进程副本驻留在内存中，一旦有请求出现，就立即使用这些空余的子进程进行处理，这样就不存在生成子进程造成的延迟了。这些服务器副本在处理完一次HTTP请求之后并不立即退出，而是停留在计算机中等待下次请求。对于客户浏览器的请求反应更快，性能较高。

## **2、参考文献**

    1）<http://blog.csdn.net/hguisu/article/details/7386882>

    2）<http://www.phpernote.com/news/723.html>

    3）<http://www.jb51.net/article/37759.htm>

    4）<http://blog.csdn.net/21aspnet/article/details/3280512>

    5）<http://www.williamlong.info/archives/1846.html>

    6）<http://www.phpddt.com/php/php-cli.html>

    7）<http://www.cnblogs.com/ellisonDon/archive/2012/11/19/2777526.html>