Nginx + VFP FastCGI开发说明

（by木瓜 2019）

# 1.为什么放弃IIS+CGI / IIS+FastCGI

众所周知IIS的性能令人诟病，早期VFP开发WEB也没更好的方案，采用了CGI的方式运行于IIS内，好处是每次WEB调用，VFP程序运行后就结束，不存在内存泄漏，变量冲突等，也不会引起宿主IIS崩溃，但坏处就是频繁的启动进程，性能不理想。但FastCGI程序不同，执行完请求后，不会立即结束，而是留在内存等待下一次请求，减少了大量进程初始化的过程，性能就大大提高，缺点就是VFP开发人员要注意，大量的全局变量，不规范的变量使用等，会“污染”下一次请求。

# 2.为什么选择Nginx

Nginx恐怕是目前性能最强悍的WEB服务器、反向代理服务器，只有几M大小，而且是开源的，支持linux,windows等多平台，只需简单的几行配置就可以完成负载均衡、url rewrite等功能 。另外，网上关于nginx的资源丰富，基本都是修改配置文件来实现不同的功能，linux界的大牛很多，你需要的配置不管是windows和linux都是相同的。

目前淘宝采用的就是Nginx服务器，看看双11产生的海量并发就可以感受这是一个多么牛的程序。

感兴趣可以搜索下看看大家的评论，几个主流服务器lighttpd,apache,iis的对比。

官方地址：<http://nginx.org/>

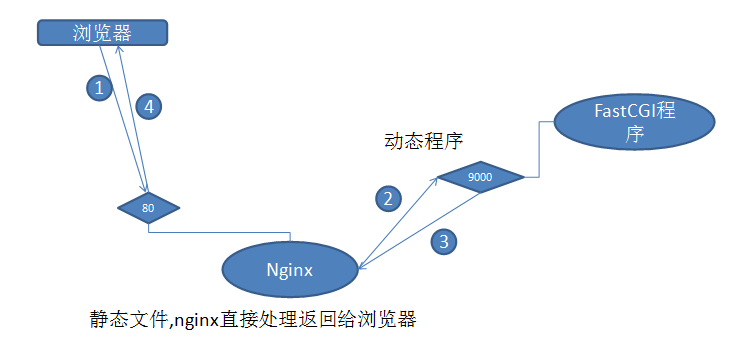
使用Nginx的缺点就是没有图型化的配置界面，需要手动修改配置文件，但是，网上这方面资源还是很多的，网上搜下，想要怎么配置都有详细的说明。

# 3.Nginx如何与FastCGI程序通信

Nginx本身只支持静态资源，比如html,jpg等，不支持php,asp,python ,lua等脚本语言，目前，如日中天的php,python都是通过fastcgi协议运行于Nginx下，可以通过一条指令fastcgi\_pass来指定转发，例如



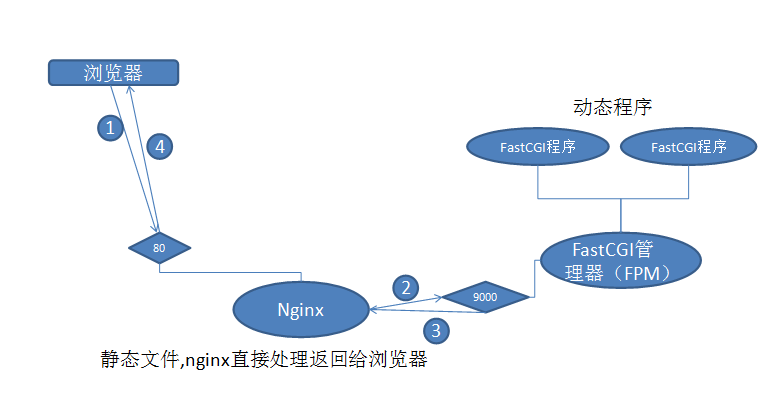
以上面配置为例，FastCGI程序通过监听9000端口，接受nginx的转发请求，处理完后返回给Nginx，Nginx再返回给浏览器，FastCGI程序并不退出进程，继续监听下一次请求，整个通信过程是这样的：



其中，1，4采用的是HTTP协议，2,3采用的是FastCGI协议

当然，这样会产生一个问题，这就是FastCGI程序本身需要管理来自Nginx产生的并发，增加了开发的难度。然而，办法总是有的，这就是一个叫FPM的中间层 (FastCGI Process Manager)，由FPM来管理 FastCGI程序，FastCGI程序本身只用专注于业务逻辑，不用关心具体的通信细节。

增加了中间层的结构是这样的：



那么这个FPM需要自己来开发吗？当然不需要，网上已经有大量现成熟可靠的管理工具，不需要你再造轮子。推荐两个： php-cgi-spawner 和 xxfpm，源码地址：

php-cgi-spawner

<https://github.com/deemru/php-cgi-spawner>

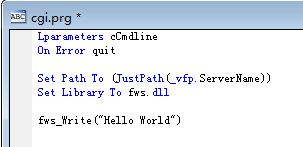
xxfpm

<http://xiaoxia.org/2011/02/01/xxfpm-wrote-a-fastcgi-process-manager/>

<https://github.com/78/xxfpm>

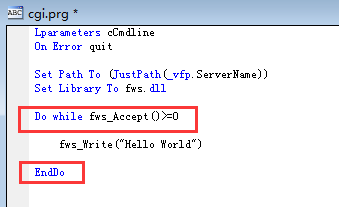
# 4.使用VFP要如何开发FastCGI程序

早期使用FWS开发CGI程序，只需要简单的一句输出就可以了：



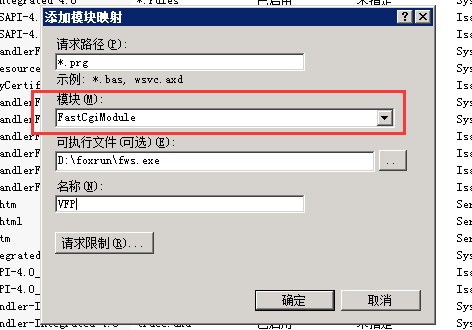
程序结束就向浏览器输出了 hello world

前面讲过，FastCGI程序每一次请求，并不会退出程序，而是等待下一次请求，那么代码是这样写的：

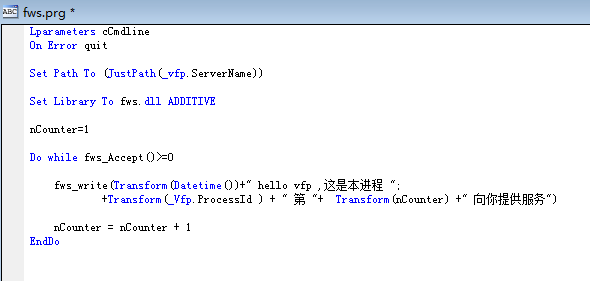


你没看错，旧的CGI程序只需要增加条循环语句，即可改造成FastCGI程序，FWS保持向后兼容，这段程序编译后，可以同时运行于CGI和FastCGI模式。

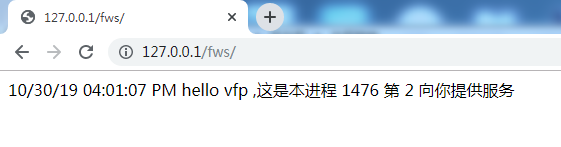
如果你想继续使用IIS，可以把模块映射修改为FastCGIModule即可。



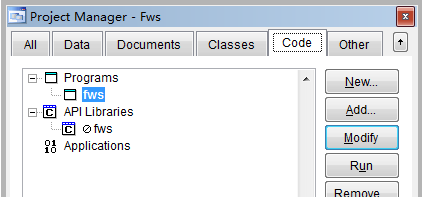
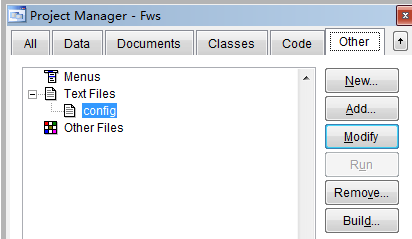
由于FASTCGI是并发运行的，为了能够在浏览器里查看是哪个进程返回的数据，我们把前面的程序修改如下：



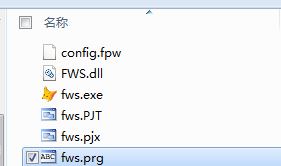
你在浏览器中看到的会是这样：

代码写好了，编译成exe即可，注意别忘了加入vfp的config.fpw文件一块编译。

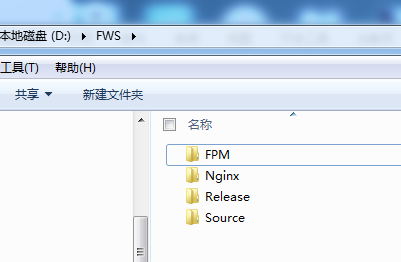
文件清单：



# 5.配置与发布

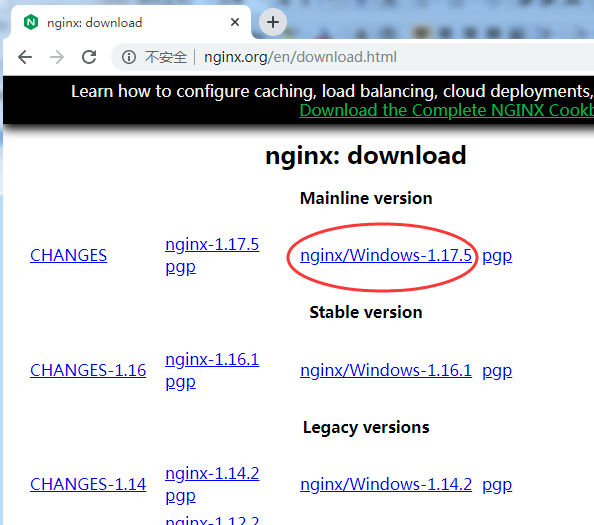
本文所有项目均放在d:\FWS文件夹，当然，可根据需求自己需要调整

1. 源码存放在Source文件夹
2. Nginx存放在 Nginx文件夹
3. FPM管理器存放在FPM文件夹
4. 发行文件存放在Release文件夹

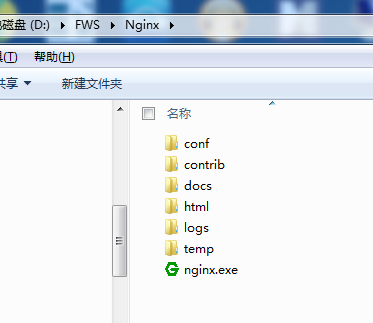


## Nginx下载与设置

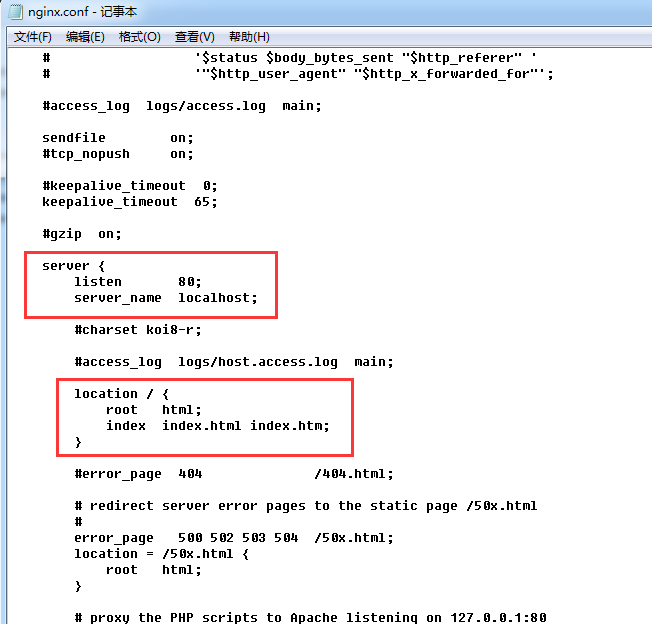
从Nginx官网下载最新版本，当前版本1.17.5



下载后是个zip压缩包，直接解压到我们的nginx文件夹就好

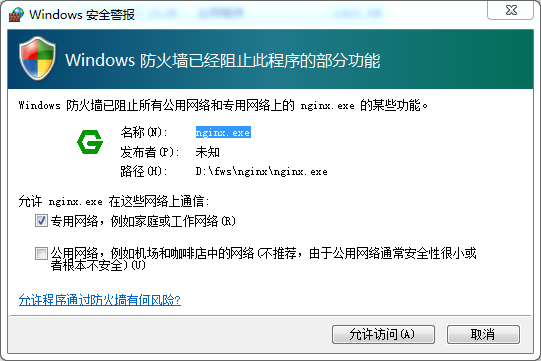


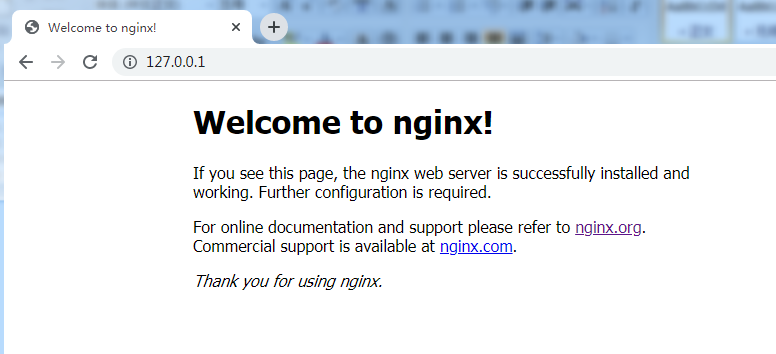
Nginx的配置文件在conf文件夹下的nginx.conf文件，这个可以直接用记事本打开编辑就好：



可以看到，它默认的配置是监听80端口，默认的主目录是在 html文件夹，如果此时双击打开nginx.exe，在浏览器地址栏里输入127.0.0.1 就可以看到它的初始页面：

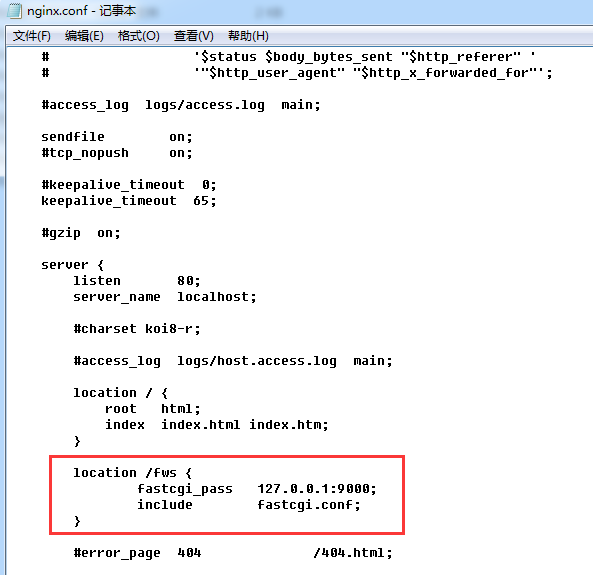
当然，防火墙会跳出来提示你，允许就好。





看到这个画面，说明nginx已经正常运行了。所有放在 nginx\html\下的静态资源都可以访问。

下面来修改配置，让 <http://127.0.0.1/fws>这个文件夹指向我们VFP开发的程序



其中：location /fws 表示当访问网站根目录 / 下在的 fws时，执行下面的配置，

比如

http://127.0.0.1/fws

http://127.0.0.1/fws/

http://127.0.0.1/fws/abc

http://127.0.0.1/fws/a.prg

http://127.0.0.1/fws/a.prg?name=test

这些web请求都会进入我们的vfp程序，而我们的VFP程序则需要判断URL或者其他参数来执行对应的操作

fastcgi\_pass 127.0.0.1:9000;

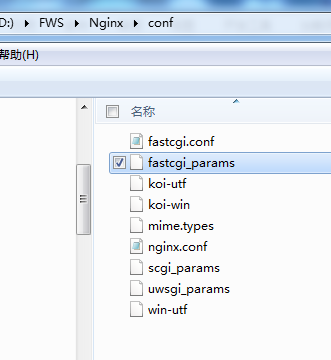
这句意思是，按照fastcgi协议，转发给9000这个端口即可。

当然，本机，可以直接使用127.0.0.1这个IP，如果是局域网的其他机器，这里换成其他IP也是可以的。

所以，web服务可以运行于一台独立的服务器（不仅限于windows），我们的VFP程序，可以运行于另一台独立的服务器，可以更好的进行处理并发。

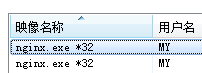
include fastcgi.conf;

这句是加入fastcgi的配置参数一并转发给fastcgi程序，include相当于vfp的宏#include，可以包含一个文件进来，有兴趣可以看下fastcgi.conf这个文件内容是什么，位于nginx.conf相同的文件夹



要注意下，由于历史原因，你可能会看到有人使用 include fastcgi\_params这个写法，这是历史原因，不需要再使用。

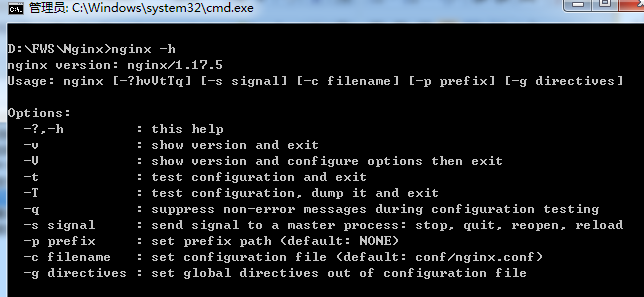
注：如果修改了配置，需要从任务管理器终止nginx进程，它有两个进程，都要终止



当然，也可以用命令重新加载配置：

nginx -s reload

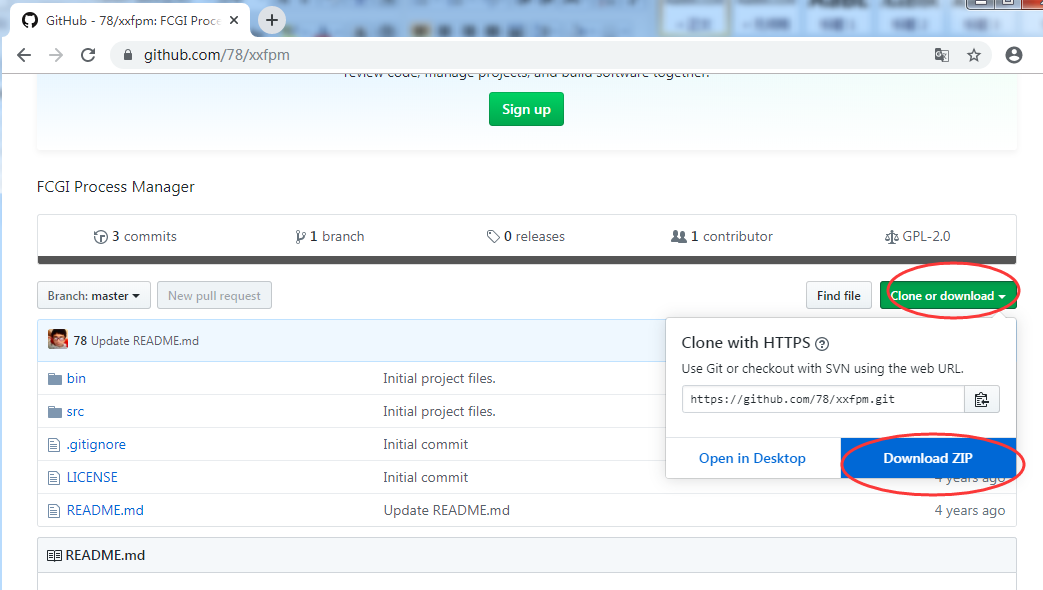
你可以使用-h参数查看帮助



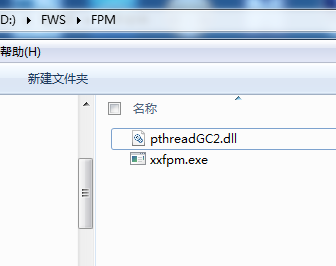
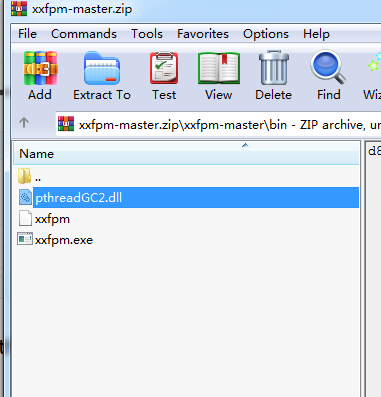
## FPM下载与用法

xxfpm是有编译好的二进制程序，php-cgi-spawner下载到的是源码，这里以xxfpm为例：

<https://github.com/78/xxfpm>



我们只需要压缩包里的xxfpm.exe和pthreadGC2.dll两个文件，把它解压到fpm文件夹



如果你有安装C编译器，可以自行编译 src\main.c文件。

xxfpm的语法

Usage: xxfpm path [-n number] [-i ip] [-p port]

Manage FastCGI processes.

-n, –number number of processes to keep

-i, –ip ip address to bind

-p, –port port to bind, default is 8000

-u, –user start processes using specified linux user

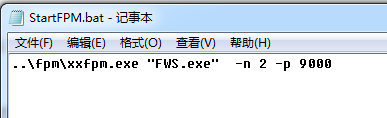
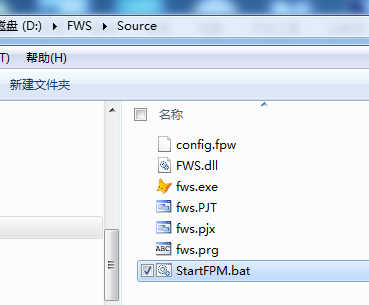
-g, –group start processes using specified linux group

-r, –root change root direcotry for the processes

-h, –help output usage information and exit

-v, –version output version information and exit

我们可以在source文件夹建一个bat文件来执行xxfpm



命令如下：

..\fpm\xxfpm.exe "FWS.exe" -n 2 -p 9000

注意，这里都使用了相对路径，如果你的文件夹不一致，请使用绝对路径 如

D:\fws\fpm\xxfpm.exe "d:\fws\source\FWS.exe" -n 2 -p 9000

其中参数 –n 2 是指立即启动两个进程常驻内存

参数 -p 9000 是指监听9000端口，要与前面nginx里指定的端口一致。

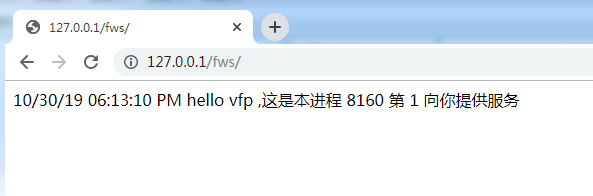
php-cgi-spawner

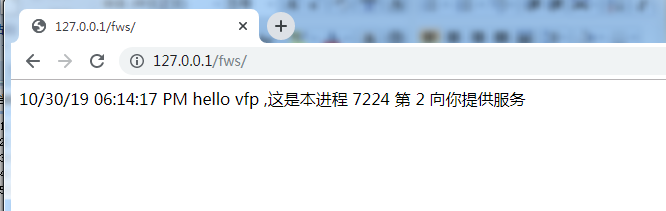
这个程序没有提供编译好的程序，有兴趣可以自己编译下，我在附件中提供这个编译好的程序，它的语法是：

..\fpm\fcgi-spawn.exe "FWS.exe" 9000 4+16

其中，9000是监听端口，4+16意思是指，常驻内存4个进程，上限16个

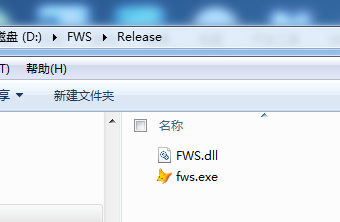
至此，fastcgi程序准备好了，nginx启动好，fpm启动好，工作完成，打开浏览器试试吧：





## VFP程序发布

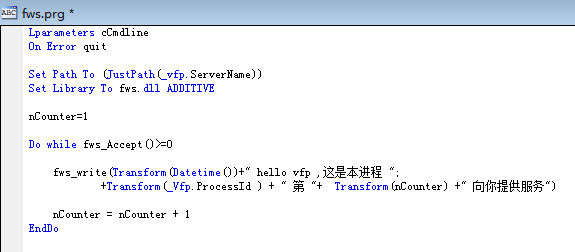
Vfp程序发布比较简单，只需要你的EXE+fws.dll和VFP运行库，以及你使用的其他数据、扩展库等，本例中只需要fws.exe和fws.dll即可，发布服务器要注意修改fpm启动的参数。



基本原理和配置工作完成，可以专注于业务逻辑的开发了。

# 6.VFP FastCGI程序详解

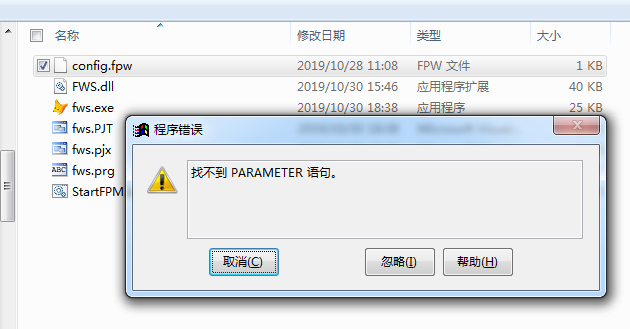
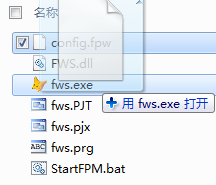
代码比较简单，但想想还是要详细解释下，因为VFP里有太多似是而非的东西。



第一行：Lparameters cCmdline

我们都知道，所有程序都有命令行参数，但VFP程序一般都是有界面的，很少人以命令方式去运行，这行代码就是接受命令行的参数，但有人会问：我又不需要处理命令行，这句有必要吗？

做个试验，在文件夹中拖动个文件到你的EXE图标上，你的程序立马报错：



看到了吗？你的程序还没有运行就报错了，连你的错误处理程序on error都没运行就报错了，而加了这行代码，你的程序就不会出错了。

第二行 On Error quit

在运行时，我们不希望产生错误而让这个程序直接挂在服务器上，产生错误要立即退出程序。因为这个时候，fws还没加载，你自己的一些设置可能都还没准备好，不能在这个时候产生错误。

当然，我真正需要捕捉错误应该怎么做呢？那是应该要等到FWS加载成功后，可以向浏览器输出信息时，再设置 On Error 程序。

第三行 Set Path To (JustPath(\_vfp.ServerName))

我们的VFP程序运行的当前路径，都是WEB服务器设定文件夹（当前NGINX设置的文件夹是 d:\fws\nginx\html），但是我们运行时需要的dll,数据，配置等信息一般跟EXE放在相同的位置，如果此时使用Set Library To fws.dll ADDITIVE 打开扩展库，vfp会先从当前目录、system32目录搜索，如果找不到，就报错了。

当我们用Set Path To设置一个文件夹时，就继续会到这个文件夹去搜索，这样才能保证第四句不会出错。

当然，你可以直接切换到程序所在文件夹，你可以把这行代码改成：

Set Default To (JustPath(\_vfp.ServerName))

程序的当前目录就直接切换到你的exe所在位置了，但开发环境不要这样，因为它会切换到vfp9.exe所在位置，这是因为\_vfp.ServerName 这个全局对像的属性，在开发环境下是 c:\program files\…\vfp9.exe，而运行环境下就是你的exe的完整路径。

另外，千万不要使用 Set Default To Sys(5)+Sys(2003)设置当前工作路径，太多人被这个错误的用法误导，Sys(5)+Sys(2003)是用来获取当前位置，既然是当前路径还要设为默认值，这句不是多余吗？

另外，或许你注意到了 JustPath()外面又套了层括号，但我建议你这样做，因为你难保证你什么时候就写出了这样的代码：

str="d:\fws"

Set Default To str

这句在VFP里是报错的，也许你想不明白，直接用 Set Default To "d:\fws" 就可以，为什么改成变量却报错了呢？这也许是VFP的历史遗留问题，你直接写Set Default To d:\fws 也不会报错，后面这个d:\fws也不是字符串，算什么呢？

所以这样修改下，就不会报错了：

Set Default To (str)

当然，你也可以使用 &，但我不建议这么做。

后面的几行需要连在一起解释：

Do while fws\_Accept()>=0

……

EndDo

这是个死循环，就是等待一次浏览器的请求，处理请求，继续下一次等待。那么我们的程序如何结束呢？只有通过任务管理器强制终止了。

Fws\_Accept()函数是个阻塞函数，等待客户端连接，如果有连接进来，会返回一个大于等于0的值，如果返回了负值，则说明FPM管理器通知你要结束了。当运行于CGI模式时，这个函数只会第一次返回大于等于0的值，第二次执行就返回了负值，这样进程就自动结束了，从而保证你的程序兼容CGI模式。

当然，为了保证程序的稳定，运行一段时间最好重启一下，就可以用计数器来控制下，比如PHP中常用的手法是执行5000次后这个进程自动退出，我们就可以修改成这样：

nCounter=1

Do while fws\_Accept()>=0

If nCounter>5000

Exit

EndIf

……

nCounter = nCounter + 1

EndDo

第5000次调用后，退出循环，进程自动终止。这时FPM管理器会监视到内存中的进程少了一个，会自动再启动一个进程，起到了类似刷新的机制。

也许你会奇怪为什么没有 Read EVENTS语句，这是我们传统VFP程序必需要写的一句，否则就会“一闪而过”，这是因为它只适用于桌面程序，需要此语句来激活Windows消息处理循环，响应键盘、鼠标的控制消息。我们的WEB程序是没有界面的，而且也不依赖Windows消息循环。当你在最后加上这句会怎样？实际上是不会出错的，但这个程序就成了僵尸程序，因为收不到桌面的通知，一直运行后台等待有人通知他，你只能通过任务管理器来终止此进程。

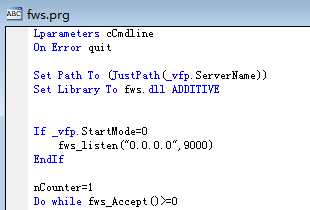
此演示程序仅仅是调用 fws\_write()输出了字符串，实际应用，你需要调用其他函数来获取浏览器发过来的所有信息进行分析处理，这不在本文解释范围。

# 7.如何调试VFP FastCGI程序

早期CGI程序调试是非常痛苦的，只有编译成EXE后，设计复杂的日志来调试，当然也有各位前辈设计了SOCKET调试服务器来实现，但新的FWS利用FASTCGI的通信特点，增加了监听函数，这样可以直接接受nginx的连接（IIS暂时难以使用此方法）：

fws\_listen(IP,端口)

当然，这个函数仅用于开发模式，不依赖FPM，运行模式时不需要，那么我们的代码需要改成这样：



If \_vfp.StartMode=0 此句检测是否是开发环境，如果是开发环境，则执行。这样就可以直接设置断点，监控整个通信过程了。

我们可以直接在循环体内部设置断点，运行程序后，从浏览器访问，则激活VFP的调试：

