CGI介绍及使用Python来开发CGI应用示例

点击 下载pdf阅读 (如果浏览器不支持直接打开,请点击右键另存)

摘要

Contents

本文主要介绍了CGI相关的一些知识, 并使用Python来开发一个CGI的应用作为示例.

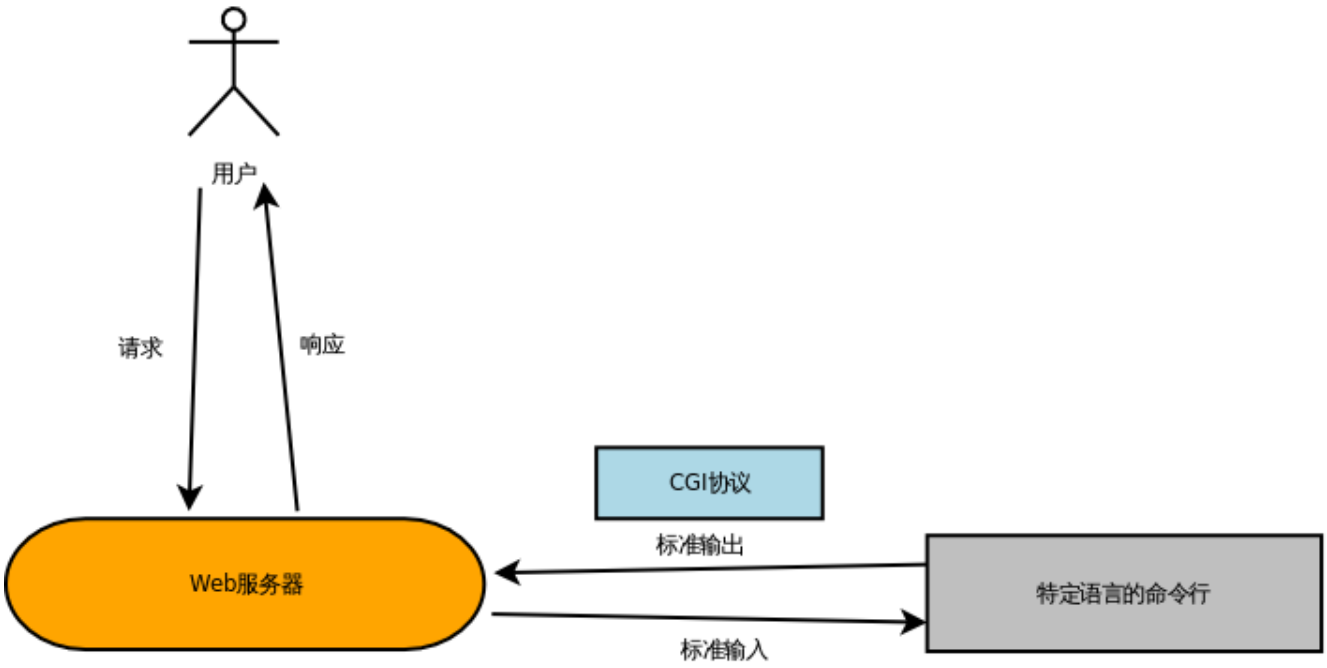
引入

我没有经历过 CGI 非常火的年代,也一直没有机会了解 CGI 开发web应用的开发过程,直到 最近听一位室友很藐视地说 **QQ**居然还在用**CGI** 来开发自己的应用, 于是心生出了想了解 CGI 的想法, 于是经过一些学习和调研,也弄清楚了一些问题,和大家分享下.

CGI的介绍

wikipedia中对 CGI 有比较详尽的描述, 基于自己的理解总结如下.

首先,CGI在web服务器处理请求中的角色关系如下图:



如上图, CGI 在其中扮演的是在web服务器和特定语言解释器之间输入输出的协议的角色, 每个来自用户的请求, web服务器都会唤起特定语言解释器的命令行(例如Python), CGI 会作为一种约定来将web服务器获得的请求数据(如url,post data)等,有选择地 作为命令行参数来输入到解释器的命令行中(标准输入), 解释器根据输入 构造出特定的html作为标准输出, 此时 CGI 又会对输出作额外的处理,如加入特定的 header(mimetype,cookie等)返回给web服务器,继而返回给用户(web服务器可能会作额外的处理).

这就是一个完整的处理流程.

CGI的缺点

CGI 作为一种标准协议后,各种主流的web服务器都支持,如 apache, IIS 等, 那么从上面的处理流程中我们会发现其中的几个主要缺点:

1. 对于每个请求,都需要新创建一个解释器的进程,而进程的创建通常都是比较昂贵的(expensive)
2. 而且,对于脚本语言,解释器还需要一定的时间来解释生成对应的html
3. 更大机率的 code injection , 因为在cgi脚本中都是手动地处理html所以更容易引起代码注入(当然更多地取决于程序员本身)

使用Python来开发简单的CGI应用

环境如下:

1. Ubuntu 8.10
2. Apache 2.x
3. Python 2.5.2

配置环境

编辑apache配置文件(我的是/etc/apache2/apache2.conf), 加入下面一行:

AddHandler cgi-script .py

告诉apache来使用CGI协议来解释python文件.

编写**python**文件

这里是个helloworld的应用, 更复杂的可参考 Python CGI.

代码如下(假设名为test.py):

```
#! /usr/bin/env python

print "Content-Type: text/html"     # HTML is following
print                               # blank line, end of headers

print "<html><header><title>Test CGI Python</title></header><body>Hello CGI!
```

需要注意的是:

1. 一个CGI脚本由2部分组成, 第一部分是输出header, 第二部分是输出常规的html
2. 注意 *#! /usr/bin/env python* 这行代码,是说明执行本脚本所用的程序(这是shell的相关知识)

那么此时可以在浏览器里看到相应的输出结果.

引申

关于框架

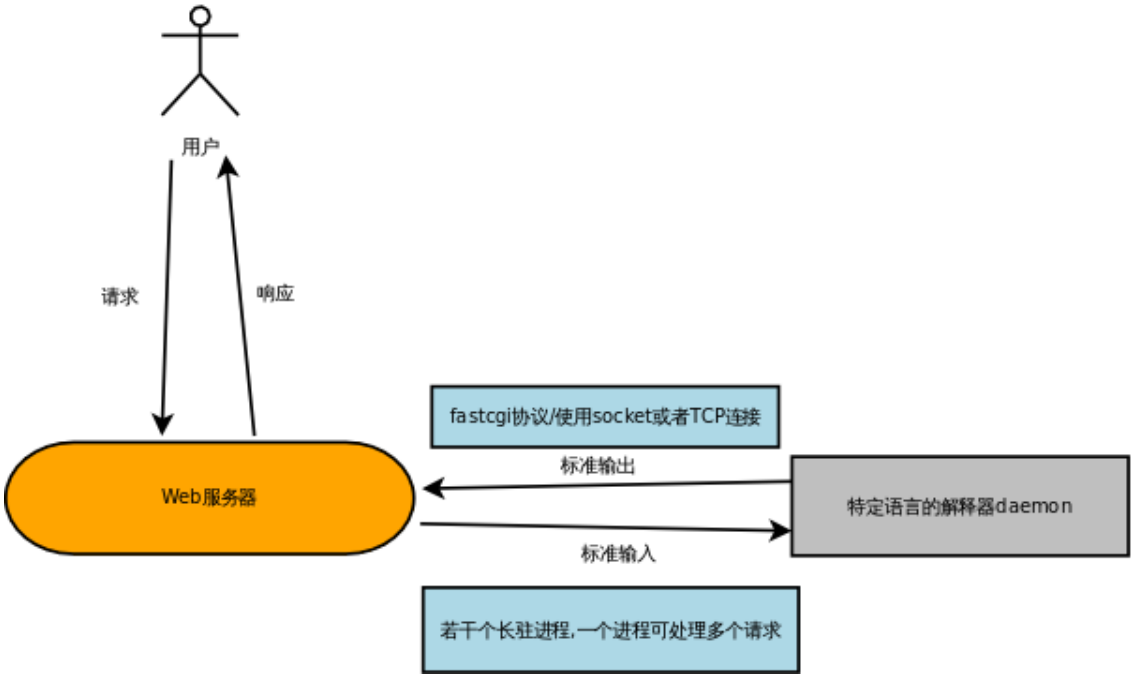
如果你弄清楚了上面的示例,你就能够明白 各种**web**框架**(django, cakephp**等**),**都只是简化和封装了一些处理的方式**,**本质还是类似于**CGI**的处理方式, 即1)header 2)html.

想想 django 的 *return HttpResponse("<html><header></header><body>Hello</body></html>*, 和上述是完全相同的.(HttpResponse默认使用text/html)

关于**CGI**的缺陷**-Fastcgi**

既然这些缺陷是如此明显,后续的一些web服务器的设计者或者web开发者便开始着手解决这些问题, 比较重要的有:

fastcgi 的出现, fastcgi 的处理流程如下

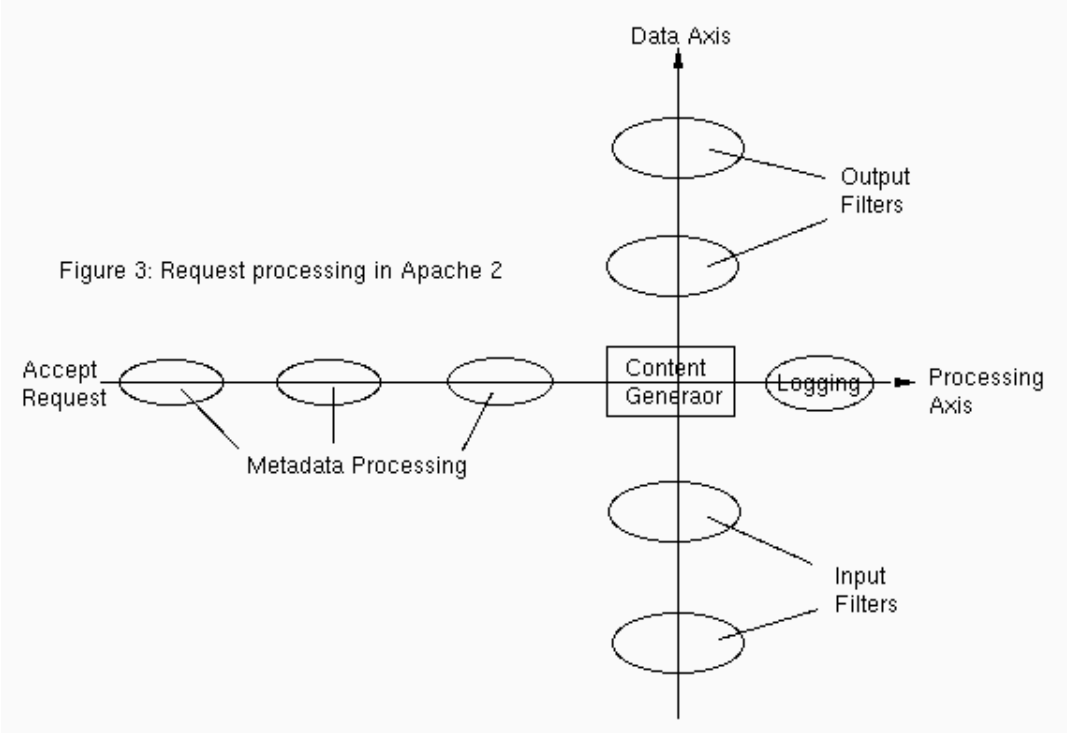


web服务器和解释器之间使用TCP或者socket来连接,在启动时会启动若干个(可以配置)长驻进程来提供请求的服务(减少建立和销毁进程的开销), 一个进程可以服务于多个请求(使用多线程或者事件驱动,参考 fastcgi spec).

所以 fastcgi 很好地解决了进程建立/销毁开销的问题.

关于CGI的缺陷-mod_xxx

另一个思路是类似于apache的mod_perl这样的解决方式.这种方式是将处理逻辑集成在web服务器之中, 如下图:



其中你可以在Data Axis(竖轴)上, *Content Generator* 之前或者之后来加入相应的filter(即module), 来进行特定的处理. 具体可以参考: Request Processing in Apache

通常而言,集成在web服务器中的module方式,会有更好的性能优势, 但是因为是集成在web服务器中, 所以mod_xxx崩溃很有可能会使得web服务器也崩溃.而 fastcgi 与web服务器之间是独立的进程 一个挂了不会影响另一个.

具体的二者的细节和优劣势,我想后面再写一篇博客来阐述.

结论

最后,我们回到我的室友提到的那个问题, **QQ**居然还在用**CGI**, 通过上面的分析,我们现在应该很明确了, 使用纯的CGI并不是一个好的办法, 因为它的诸多缺陷, 所以可以使用 fastcgi 或者module的方式.

当然,作为最初动态网页内容处理的始祖, CGI 是具有里程碑式意义的协议,到后来的 fastcgi, scgi 等 都是与 CGI 的理念相同的.

后记

好久没更新日志了,因为最近实在很忙,不过后面还会尽量来更新的.

欢迎大家讨论留言.

参考资料

1. Request Processing in Apache
2. Python CGI

3. fastcgi

本文的源码

本文的rst源码链接在 这里 .

点击 下载pdf阅读 (如果浏览器不支持直接打开,请点击右键另存)