**CGI与Web开发**

**C++后台实践：古老的CGI与Web开发**

本文写给C/C++程序猿，也适合其他对历史感兴趣的程序猿

=============================================

请求行（request line）、请求头部（header）、空行和请求数据四个部分组成。



        谈到web开发，大家首先想到的PHP、JavaEE/JSP、.NET/ASP、Ruby on rails、[Python](http://lib.csdn.net/base/11)的Django等等。可谓百花齐放，你一般不会想到C++和Web开发有什么关系，但其实动态网页的开发（web开发）可是在这些动态网页语言诞生之前就存在了的。所以C/C++也是可以做web开发的，它利用的技术是——CGI。

        在天地初开，混沌未分之时，动态网页语言尚未出世，要实现动态网站依赖的就是CGI。谷歌/百度一下CGI，可能会出现很多名词：CGI脚本、CGI程序、CGI标准等等。其实这些都是站在不同角度来说的，CGI即Common Gateway Interface的缩写，直译为“通用网关接口”。第一次听这个名字，我也不知道是个什么鬼东西。归根结底 CGI就是一个接口协议。协议就是大家公认的一套标准（叫CGI标准也可以），比如网络协议。大家都遵守一套标准，就减少了沟通的难度。进行CGI开发，就是编写一个CGI可执行程序。其实各种语言都可以编写CGI，不但Java、Python、PHP、C#……可以，而且Shell也可以。当然C和C++也可以。由于早期CGI很多是由Perl（脚本语言）开发的，所以CGI程序也称CGI脚本，其实这个称呼不一定准确。因为C++编译出的可执行文件同样可以是CGI。

        在PHP和Java大行其道的今天，很多人看来用C++编写CGI是几乎淘汰的技术了（其实这到不然，只是比较小众罢了）。所以如果你对C/C++感兴趣或者对历史感兴趣都可以阅读本文。

**一次网页请求与响应**

        在进行网页浏览时，通常就是通过一个URL请求一个网页，然后服务器返回这个网页文件给浏览器。浏览器在本地解析该文件渲染成我们看到的网页。然而通常我们看到的网页不是静态网页，也就是说在服务端是没有这个网页文件，它是在网页请求的时候动态生成的，比如PHP/JSP网页。依据你请求的参数不同，所返回的内容不同。

        同理，如果是请求一个CGI程序的时候（比如在浏览器直接输入CGI程序的URL，或者提交表单的时候发送给CGI程序），CGI程序负责解析从前端传递过来的参数，理解它的意图然后返回数据，比如返回HTML、XML或JSON等。

WARNNING：Apache默认没有打开CGI的支持，需要进行CGI的配置。具体方法可以自行百度。

**预备前端知识**

        假设你是一个C++程序员，你可能对前端不熟（OK，我也不熟），在接下来的讲述之前，你要先掌握一些预备的前端知识（尽量少讲前端），你不需要知道如何渲染出一个美轮美奂的网页，但你需要知道前、后端如何交互。前端页面如何发送数据，一个普通的HTML页面通常的做法，你只需知道如下几种：

* form表单提交（html原生）
* js操纵下的表单提交
* js通过Ajax请求数据

这里知讲第一种（最简单的）：

**[html]** [view plain](http://blog.csdn.net/wm_1991/article/details/51942406) [copy](http://blog.csdn.net/wm_1991/article/details/51942406)

1. **<h1>**表单提交**</h1>**
2. **<form** action="/cgi-bin/hello.cgi" method="get"**>**
3. **<table>**
4. **<tr>**
5. **<td>**用户名：**</td>**
6. **<td><input** name="username"**/></td>**
7. **</tr>**
8. **<tr>**
9. **<td>**密码：**</td>**
10. **<td><input** name="password"**/></td>**
11. **</tr>**
12. **<tr>**
13. **<td><input** type="submit" value="OK"**/></td>**
14. **</tr>**
15. **</table>**
16. **</form>**

form标签的action属性的值表示的就是表单要提交到url，即表单提交以后要跳转的页面（Ajax可以达到无跳转拉取数据，刷新页面），这里action属性值的是cgi程序的url地址。（WARNNING：/ 对应的是网站根目录，而不是Linux文件系统根目录哦）。method属性表示数据请求方式，有两种：get和post。不赘述。

        我输入用户名jellywang，密码123456之后，点击OK按钮，即向 当前域名/cgi-bin/hello.cgi 的程序序提交了表单，并且携带参数username=jellywang。然后页面会跳转到这个cgi（就像普通网页跳转，浏览器地址栏更新一样）。

        如果是get请求。那么浏览器地址栏的URL看起来像这样：localhost:/cgi-bin/hello.cgi?username=jelly&password=123456。很显然这是一种不够安全的方式，所以我们还可以使用post请求。这样地址栏就看不到这种提交的参数了。（其实post也不够安全，不鼓励直接提交明文密码的方式，本文仅作示例，安全登录不上本文重点）

**环境变量与CGI处理**

        当前端页面通过get或post方法向cgi程序提交了数据以后，那么接下来cgi程序该如何解析呢？答案是环境变量。无论是Linux系统或Windows系统都有环境变量的概念。Linux用户在配置很多环境的时候，都不得不在系统配置文件中和环境变量打交道。CGI程序即是通过从环境变量中取值来获得参数的。这里介绍几个环境变量（更多的请自行百度）：

|  |  |
| --- | --- |
| REQUEST\_METHOD | 前端页面数据请求方式：get/post |
| QUERY\_STRING | 采用GET时所传输的信息 |
| CONTENT\_LENGTH | STDIO中的有效信息长度 |
| SCRIPT\_NAME | 所调用的CGI程序的名字 |
| SERVER\_NAME | 服务器的IP或名字 |
| SERVER\_PORT | 主机的端口号 |

        这些环境变量是从何而来，是谁定义的？是Linux吗？POSIX吗？当然不是。这里就要再次声明一下CGI是一个接口协议，这些环境变量就是属于该协议的内容，所以不论你的server所在的操作系统是Linux还是Windows，也不论你的server是Apache还是Nginx，这些变量的名称和含义都是一样的。实际就是Apache/Nginx在将这些内容填充到环境变量中，而具体填充规范则来自于CGI接口协议。

        在C语言标准中有获取环境变量值得库函数——getenv。（头文件stdlib.h）

**[cpp]** [view plain](http://blog.csdn.net/wm_1991/article/details/51942406) [copy](http://blog.csdn.net/wm_1991/article/details/51942406)

1. //比如
2. chr\* str = NULL;
3. str = getenv("QUERY\_STRING");

        对于get请求，可以从环境变量QUERY\_STRING中取出字符串 username=jelly&password=123456。然后程序自己做字符串的解析操作，解析出参数的key和value。而对于post请求，则是直接通过标注输入（STDIN）来获取这个参数字符串，比如使用scanf或cin都可以。

        在解析了请求、进行了相应的逻辑处理之后（比如检查用户名密码是否一致），CGI程序要向前端页面返回内容，这是通过标准输出（STDOUT）完成的，比如printf或cout，你可以返回xml，json，plain text或一个html网页等等。这一步完成的是就是HTTP的响应过程。所以在返回直接的数据之前，要先输出HTTP协议的首部。比如，假设你想返回一个html网页，那么你首先要输出：

**[cpp]** [view plain](http://blog.csdn.net/wm_1991/article/details/51942406) [copy](http://blog.csdn.net/wm_1991/article/details/51942406)

1. cout<<"Content-Type:text/html\n\n"<<endl;

WARNNING：这里要注意，一定要输出两个换行符（\n）。因为HTTP协议的首部和消息实体（如HTML代码）之间用空行分割。

        后面直接cout出html代码（比如输出你刚才输入的用户名成功登陆）。前端页面就会收到这些html代码，然后浏览器就渲染成网页啦。这就是一次CGI完成的动态网页操作了。

**Cgicc库**

        进行C++的CGI编程，需要手动进行字符串的解析处理，还有自行管理首部。比如资源转移了，要返回302，并且在首部用Location给出新地址。很显然，这些东西对于PHP、Python等语言都有内置的解决方案。对于C++就需要第三方库了。这里推荐一个GNU的开源库——[Cgicc](http://itfish.net/article/54314.html)。可以满足常用的各类需求，除了解析get/post请求外，还能重定向，还可以设置Cookie，还可以上传文件等等等等。

        美中不足的就是Cgicc库不支持SESSION。但是这个问题不大，我们可以很容易使用Cookie来实现SESSION功能。由于CGI本身是请求一次就创建一个进程，返回之后进程就结束（下文的FastCGI除外）。这时要在服务端维持一个SESSION的变量可选的解决方案是：用文件存储或者在Redis、Memcached等内存数据库中存储。而发给客户端的SESSIONID就用Cgicc已经支持的Cookie功能来完成，就可以了。

**CGI的痛点与FastCGI**

        CGI是一种标准，并不限定语言。所以Java、PHP、Python都可以通过这种方式来生成动态网页。但是实际上这些动态语言却很少这样用。原来是CGI有一大硬伤。那就是每次CGI请求，那么Apache都有启动一个进程去执行这个CGI程序，即颇具Unix特色的fork-and-execute。当用户请求量大的时候，这个fork-and-execute的操作会严重拖慢Server的进程。而Java的Servlet技术则是一种常驻内存的技术，不会频繁的发生进程上下文的创建和销毁操作。

        时势造英雄，FastCGI技术应运而生。简单来说，其本质就是一个常驻内存的进程池技术，由调度器负责将传递过来的CGI请求发送给处理CGI的handler进程来处理。在一个请求处理完成之后，该处理进程不销毁，继续等待下一个请求的到来。FCGI技术一出，CGI又一定程度上焕发了第二春。PHP-FPM本身是使PHP支持FCGI技术的一个Patch，现在已经被纳入PHP标准。当然，支持C++的FCGI技术也出现了，Apache有FCGI的模块可以安装，比如mod\_fcgid。

**现代CGI的编程范式**

        前面我们知道，CGI可以直接返回一个html网页。CGI程序本身也可以进行各种计算、逻辑处理任务。随着各类web前后端技术的发展，以及大数据、高并发的Server使用场景越来越多。现代的CGI的用法，在发生变化。

        现在，越来越多的任务从后端转移到前端，前端页面利用丰富的Js技术来进行更多的处理。

1. JS可以使用Ajax技术来向后台CGI发起数据请求。Ajax完成的是不需要刷新整个页面就可以加载后端数据（比如从数据库中取出）。
2. CGI一般不再用于直接返回html页面，同时将复杂的计算、IO任务下沉到后端（后端可以进一步进行路由转发，实现负载均衡）。使CGI作为前后端之间的中间层。彼时CGI的职能是完成基本的数据交换：解析前端数据请求，再转发给对应后端；然后从后端取回数据，给前端返回XML或JSON。
3. 前端JS利用XML/JSON中的数据来进行填充，绘制出丰富的页面。