《CURL技术知识教程》系列分享专栏

简介

文章

- PHP采集相关教程之一 CURL函数库
- php中通过curl模拟登陆discuz论坛的实现代码
- php中通过curl smtp发送邮件
- PHP curl 并发最佳实践代码分享
- CURL的学习和应用(附多线程实现)
- php curl模仿ftp文件上传代码
- php使用curl来获取远程图片
- PHP Curl多线程原理实例详解
- curl不使用文件存取cookie php使用curl获取cookie示例
- php使用curl抓取qq空间的访客信息示例
- PHP中CURL的CURLOPT_POSTFIELDS参数使用细节
- cURL multi批处理实现及避免cURL multi造成CPU负载过高问题
- php利用curl抓取新浪微博内容示例
- PHP函数分享之curl方式取得数据、模拟登陆、POST数据
- PHP扩展CURL的用法详解
- php采用curl访问域名返回405 method not allowed提示的解决方法
- PHP执行Curl时报错提示CURL ERROR: Recv failure: Connection reset by peer的解决方法
- PHP中使用CURL模拟登录并获取数据实例
- PHP curl实现抓取302跳转后页面的示例
- PHP使用CURL实现对带有验证码的网站进行模拟登录的方法
- php中的curl_multi系列函数使用例子
- PHP使用CURL_MULTI实现多线程采集的例子
- PHP curl 抓取AJAX异步内容示例
- php中file_get_content 和curl以及fopen 效率分析
- php之curl实现http与https请求的方法
- php之curl设置超时实例
- php的curl封装类用法实例
- php采用curl模仿登录人人网发布动态的方法
- PHP基于CURL进行POST数据上传实例
- PHP中让curl支持sock5的代码实例
- PHP中CURL的几个经典应用实例
- PHP使用CURL函数获取HTTPS网页及POST数据示例
- PHP中巧用curl 并发减少获取第三方网页内容时间
- 如何通过命令查看CURL慢在哪里？
- PHP+Curl伪造客户端获取页面方法
- php curl登陆qq后获取用户信息时证书错误
- php中curl使用指南
- PHP的cURL库简介及使用示例
- PHP curl CURLOPT_RETURNTRANSFER参数的作用使用实例
- Linux中使用curl命令访问https站点4种常见错误和解决方法
- php使用curl获取https请求的方法
- 自己写的php curl库实现整站克隆功能
- PHP CURL 内存泄露问题解决方法
- php使用curl出现Expect:100-continue解决方法
- php运行出现Call to undefined function curl_init()的解决方法

- php使用curl简单抓取远程url的方法
- php curl 上传文件代码实例
- PHP curl伪造IP地址和header信息代码实例
- php curl 获取https请求的2种方法
- php curl请求信息和返回信息设置代码实例
- PHP Curl模拟登录微信公众平台、新浪微博实例代码

PHP采集相关教程之一 CURL函数库

先写一个简单的抓取页面函数

```
<?php
function GetSources($Url,$User_Agent='',$Referer_Url='') //抓取某个指定的页面
{
//$Url 需要抓取的页面地址
//$User_Agent 需要返回的user_agent信息 如“baiduspider”或“googlebot”
$ch = curl_init();
curl_setopt ($ch, CURLOPT_URL, $Url);
curl_setopt ($ch, CURLOPT_USERAGENT, $User_Agent);
curl_setopt ($ch, CURLOPT_REFERER, $Referer_Url);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION,1);
curl_setopt ($ch, CURLOPT_RETURNTRANSFER, 1);
$MySources = curl_exec ($ch);
curl_close($ch);
return $MySources;
}
$Url = "http://www.jb51.net"; //要获取内容的也没
$User_Agent = "baiduspider+(+http://www.baidu.com/search/spider.htm)";
$Referer_Url = 'http://www.jb51.net/';
echo GetSources($Url,$User_Agent,$Referer_Url);
?>
```

PHP中的CURL函数库（Client URL Library Function）

curl_close — 关闭一个curl会话；

curl_copy_handle — 拷贝一个curl连接资源的所有内容和参数；

curl_errno — 返回一个包含当前会话错误信息的数字编号；

curl_error — 返回一个包含当前会话错误信息的字符串；

curl_exec — 执行一个curl会话；

curl_getinfo — 获取一个curl连接资源句柄的信息；

curl_init — 初始化一个curl会话；

curl_multi_add_handle — 向curl批处理会话中添加单独的curl句柄资源；

curl_multi_close — 关闭一个批处理句柄资源；

curl_multi_exec — 解析一个curl批处理句柄；

curl_multi_getcontent — 返回获取的输出的文本流；

curl_multi_info_read — 获取当前解析的curl的相关传输信息；

curl_multi_init — 初始化一个curl批处理句柄资源；

curl_multi_remove_handle — 移除curl批处理句柄资源中的某个句柄资源；

curl_multi_select — Get all the sockets associated with the cURL extension, which can then be "selected"；

curl_setopt_array — 以数组的形式为一个curl设置会话参数；

curl_setopt — 为一个curl设置会话参数；

curl_version — 获取curl相关的版本信息；

curl_init()函数的作用初始化一个curl会话，curl_init()函数唯一的一个参数是可选的，表示一个url地址；

curl_exec()函数的作用是执行一个curl会话，唯一的参数是curl_init()函数返回的句柄；

curl_close()函数的作用是关闭一个curl会话，唯一的参数是curl_init()函数返回的句柄；

PHP代码

```
<?php
$ch = curl_init("http://blog.huangchao.org/");
curl_exec($ch);
curl_close($ch);
?>
```

curl_version()函数的作用是获取curl相关的版本信息，curl_version()函数有一个参数，不清楚是做什么的；

PHP代码

```
<?php
print_r(curl_version())
?>
```

curl_getinfo()函数的作用是获取一个curl连接资源句柄的信息，curl_getinfo()函数有两个参数，第一个参数是curl的资源句柄，第二个参数是下面一些常量：

PHP代码

```
<?php
$ch = curl_init("http://blog.huangchao.org/");
print_r(curl_getinfo($ch));
?>
```

可选的常量包括：

CURLINFO_EFFECTIVE_URL：最后一个有效的url地址；

CURLINFO_HTTP_CODE：最后一个收到的HTTP代码；

CURLINFO_FILETIME：远程获取文档的时间，如果无法获取，则返回值为“-1”；

CURLINFO_TOTAL_TIME：最后一次传输所消耗的时间；

CURLINFO_NAMELOOKUP_TIME：名称解析所消耗的时间；

CURLINFO_CONNECT_TIME：建立连接所消耗的时间；

CURLINFO_PRETRANSFER_TIME：从建立连接到准备传输所使用的时间；

CURLINFO_STARTTRANSFER_TIME：从建立连接到传输开始所使用的时间；
CURLINFO_REDIRECT_TIME：在事务传输开始前重定向所使用的时间；
CURLINFO_SIZE_UPLOAD：上传数据量的总值；
CURLINFO_SIZE_DOWNLOAD：下载数据量的总值；
CURLINFO_SPEED_DOWNLOAD：平均下载速度；
CURLINFO_SPEED_UPLOAD：平均上传速度；
CURLINFO_HEADER_SIZE：header部分的大小；
CURLINFO_HEADER_OUT：发送请求的字符串；
CURLINFO_REQUEST_SIZE：在HTTP请求中有问题的请求的大小；
CURLINFO_SSL_VERIFYRESULT：Result of SSL certification verification requested by setting CURLOPT_SSL_VERIFYPEER；
CURLINFO_CONTENT_LENGTH_DOWNLOAD：从Content-Length: field中读取的下载内容长度；
CURLINFO_CONTENT_LENGTH_UPLOAD：上传内容大小的说明；
CURLINFO_CONTENT_TYPE：下载内容的“Content-type”值，NULL表示服务器没有发送有效的“Content-Type: header”；
curl_setopt()函数的作用是为一个curl设置会话参数。curl_setopt_array()函数的作用是以数组的形式为一个curl设置会话参数；
PHP代码

```
<?php
$ch = curl_init();
$fp = fopen("example_homepage.txt", "w");
curl_setopt($ch, CURLOPT_FILE, $fp);
$options = array(
CURLOPT_URL => 'http://www.baidu.com/',
CURLOPT_HEADER => false
);
curl_setopt_array($ch, $options);
curl_exec($ch);
curl_close($ch);
fclose($fp);
?>
```

可设置的参数有：
CURLOPT_AUTOREFERER：自动设置header中的referer信息；
CURLOPT_BINARYTRANSFER：在启用CURLOPT_RETURNTRANSFER时候将获取数据返回；
CURLOPT_COOKIESESSION：启用时curl会仅仅传递一个session cookie，忽略其他的cookie，默认状况下curl会将所有的cookie返回给服务端。session cookie是指那些用来判断服务器端的session是否有效而存在的cookie；
CURLOPT_CRLF：启用时将Unix的换行符转换成回车换行符；
CURLOPT_DNS_USE_GLOBAL_CACHE：启用时会启用一个全局的DNS缓存，此项为线程安全的，并且默认为true；
CURLOPT_FAILONERROR：显示HTTP状态码，默认行为是忽略编号小于等于400的HTTP信息；
CURLOPT_FILETIME：启用时会尝试修改远程文档中的信息。结果信息会通过curl_getinfo()函数的CURLINFO_FILETIME选项返回；
CURLOPT_FOLLOWLOCATION：启用时会将服务器服务器返回的“Location:”放在header中递归的返回给服务器，使用CURLOPT_MAXREDIRS可以限定递归返回的数量；
CURLOPT_FORBID_REUSE：在完成交互以后强迫断开连接，不能重用；
CURLOPT_FRESH_CONNECT：强制获取一个新的连接，替代缓存中的连接；
CURLOPT_FTP_USE_EPRT：TRUE to use EPRT (and LPRT) when doing active FTP downloads. Use FALSE to disable EPRT and LPRT and use PORT only；Added in PHP 5.0.0.
CURLOPT_FTP_USE_EPSV：TRUE to first try an EPSV command for FTP transfers before reverting back to PASV. Set to FALSE to disable EPSV；
CURLOPT_FTPAPPEND：TRUE to append to the remote file instead of overwriting it；
CURLOPT_FTPASCII：An alias of CURLOPT_TRANSFERTEXT. Use that instead；
CURLOPT_FTPLISTONLY：TRUE to only list the names of an FTP directory；
CURLOPT_HEADER：启用时会将头文件的信息作为数据流输出；
CURLOPT_HTTPGET：启用时会设置HTTP的method为GET，因为GET是默认是，所以只在被修改的情况下使用；
CURLOPT_HTTPPROXYTUNNEL：启用时会通过HTTP代理来传输；
CURLOPT_MUTE：讲curl函数中所有修改过的参数恢复默认值；
CURLOPT_NETRC：在连接建立以后，访问~/.netrc文件获取用户名和密码信息连接远程站点；
CURLOPT_NOBODY：启用时将不对HTML中的body部分进行输出；
CURLOPT_NOPROGRESS：启用时关闭curl传输的进度条，此项的默认设置为true；
CURLOPT_NOSIGNAL：启用时忽略所有的curl传递给php进行的信号。在SAPI多线程传输时此项被默认打开；
CURLOPT_POST：启用时会发送一个常规的POST请求，类型为：application/x-www-form-urlencoded，就像表单提交的一样；
CURLOPT_PUT：启用时允许HTTP发送文件，必须同时设置CURLOPT_INFILE和CURLOPT_INFILESIZE
CURLOPT_RETURNTRANSFER：将curl_exec()获取的信息以文件流的形式返回，而不是直接输出；
CURLOPT_SSL_VERIFYPEER：FALSE to stop cURL from verifying the peer's certificate. Alternate certificates to verify against can be specified with the CURLOPT_CAINFO option or a certificate directory can be specified with the CURLOPT_CAPATH option. CURLOPT_SSL_VERIFYHOST may also need to be TRUE or FALSE if CURLOPT_SSL_VERIFYPEER is disabled (it defaults to 2). TRUE by default as of cURL 7.10. Default bundle installed as of cURL 7.10；
CURLOPT_TRANSFERTEXT：TRUE to use ASCII mode for FTP transfers. For LDAP, it retrieves data in plain text instead of HTML. On Windows systems, it will not set STDOUT to binary mode；
CURLOPT_UNRESTRICTED_AUTH：在使用CURLOPT_FOLLOWLOCATION产生的header中的多个locations中持续追加用户名和密码信息，即使域名已发生改变；
CURLOPT_UPLOAD：启用时允许文件传输；
CURLOPT_VERBOSE：启用时会汇报所有的信息，存放在STDERR或指定的CURLOPT_STDERR中；
CURLOPT_BUFFERSIZE：每次获取的数据中读入缓存的大小，这个值每次都会被填满；
CURLOPT_CLOSEPOLICY：不是CURLCLOSEPOLICY_LEAST_RECENTLY_USED就是CURLCLOSEPOLICY_OLDEST，还存在另外三个，但是curl暂时还不支持；
CURLOPT_CONNECTTIMEOUT：在发起连接前等待的时间，如果设置为0，则不等待；
CURLOPT_DNS_CACHE_TIMEOUT：设置在内存中保存DNS信息的时间，默认为120秒；
CURLOPT_FTPSSLAUTH：The FTP authentication method (when is activated): CURLFTPAUTH_SSL (try SSL first), CURLFTPAUTH_TLS (try TLS first), or CURLFTPAUTH_D

EFAULT (let cURL decide)；
CURLOPT_HTTP_VERSION：设置curl使用的HTTP协议，CURL_HTTP_VERSION_NONE（让curl自己判断），CURL_HTTP_VERSION_1_0（HTTP/1.0），CURL_HTTP_VERSION_1_1（HTTP/1.1）；
CURLOPT_HTTPAUTH：使用的HTTP验证方法，可选的值有：CURLAUTH_BASIC，CURLAUTH_DIGEST，CURLAUTH_GSSNEGOTIATE，CURLAUTH_NTLM，CURLAUTH_ANY，CURLAUTH_ANYSAFE，可以使用“|”操作符分隔多个值，curl让服务器选择一个支持最好的值，CURLAUTH_ANY等价于CURLAUTH_BASIC | CURLAUTH_DIGEST | CURLAUTH_GSSNEGOTIATE | CURLAUTH_NTLM，CURLAUTH_ANYSAFE等价于CURLAUTH_DIGEST | CURLAUTH_GSSNEGOTIATE | CURLAUTH_NTLM
CURLOPT_INFILESIZE：设定上传文件的大小；
CURLOPT_LOW_SPEED_LIMIT：当传输速度小于CURLOPT_LOW_SPEED_LIMIT时，PHP会根CURLOPT_LOW_SPEED_TIME来判断是否因太慢而取消传输；
CURLOPT_LOW_SPEED_TIME：The number of seconds the transfer should be below CURLOPT_LOW_SPEED_LIMIT for PHP to consider the transfer too slow and abort；当传输速度小于CURLOPT_LOW_SPEED_LIMIT时，PHP会根据CURLOPT_LOW_SPEED_TIME来判断是否因太慢而取消传输；
CURLOPT_MAXCONNECTS：允许的最大连接数量，超过是会通过CURLOPT_CLOSEPOLICY决定应该停止哪些连接；
CURLOPT_MAXREDIRS：指定最多的HTTP重定向的数量，这个选项是和CURLOPT_FOLLOWLOCATION一起使用的；
CURLOPT_PORT：一个可选的用来指定连接端口的量；
CURLOPT_PROXYAUTH：The HTTP authentication method(s) to use for the proxy connection. Use the same bitmasks as described in CURLOPT_HTTPAUTH. For proxy authentication, only CURLAUTH_BASIC and CURLAUTH_NTLM are currently supported.
CURLOPT_PROXYPORT：The port number of the proxy to connect to. This port number can also be set in CURLOPT_PROXY.
CURLOPT_PROXYTYPE：Either CURLPROXY_HTTP (default) or CURLPROXY_SOCKS5.
CURLOPT_RESUME_FROM：在恢复传输时传递一个字节偏移量（用来断点续传）
CURLOPT_SSL_VERIFYHOST：
1 to check the existence of a common name in the SSL peer certificate.
2 to check the existence of a common name and also verify that it matches the hostname provided.
CURLOPT_SSLVERSION：The SSL version (2 or 3) to use. By default PHP will try to determine this itself, although in some cases this must be set manually.
CURLOPT_TIMECONDITION：如果在CURLOPT_TIMEVALUE指定的某个时间以后被编辑过，则使用CURL_TIMECOND_IFMODSINCE返回页面，如果没有被修改过，并且CURLOPT_HEADER为true，则返回一个"304 Not Modified"的header，CURLOPT_HEADER为false，则使用CURL_TIMECOND_ISUNMODSINCE，默认值为CURL_TIMECOND_IFMODSINCE
CURLOPT_TIMEOUT：设置curl允许执行的最长秒数
CURLOPT_TIMEVALUE：设置一个CURLOPT_TIMECONDITION使用的时间戳，在默认状态下使用的是CURL_TIMECOND_IFMODSINCE
CURLOPT_CAINFO：The name of a file holding one or more certificates to verify the peer with. This only makes sense when used in combination with CURLOPT_SSL_VERIFYPEER.
CURLOPT_CAPATH：A directory that holds multiple CA certificates. Use this option alongside CURLOPT_SSL_VERIFYPEER.
CURLOPT_COOKIE：设定HTTP请求中“Set-Cookie:”部分的内容。
CURLOPT_COOKIEFILE：包含cookie信息的文件名称，这个cookie文件可以是Netscape格式或者HTTP风格的header信息。
CURLOPT_COOKIEJAR：连接关闭以后，存放cookie信息的文件名称
CURLOPT_CUSTOMREQUEST：A custom request method to use instead of "GET" or "HEAD" when doing a HTTP request. This is useful for doing "DELETE" or other, more obscure HTTP requests. Valid values are things like "GET", "POST", "CONNECT" and so on; i.e. Do not enter a whole HTTP request line here. For instance, entering "GET /index.html HTTP/1.0\r\n\r\n" would be incorrect.
Note: Don't do this without making sure the server supports the custom request method first.
CURLOPT_EGBSOCKET：Like CURLOPT_RANDOM_FILE, except a filename to an Entropy Gathering Daemon socket.
CURLOPT_ENCODING：header中“Accept-Encoding: ”部分的内容，支持的编码格式为："identity"，"deflate"，"gzip"。如果设置为空字符串，则表示支持所有的编码格式
CURLOPT_FTPPORT：The value which will be used to get the IP address to use for the FTP "POST" instruction. The "POST" instruction tells the remote server to connect to our specified IP address. The string may be a plain IP address, a hostname, a network interface name (under Unix), or just a plain '-' to use the systems default IP address.
CURLOPT_INTERFACE：在外部网络接口中使用的名称，可以是一个接口名，IP或者主机名。
CURLOPT_KRB4LEVEL：KRB4(Kerberos 4)安全级别的设置，可以是一下几个值之一："clear"，"safe"，"confidential"，"private"。默认的值为"private"，设置为null的时候表示禁用KRB4，现在KRB4安全仅能在FTP传输中使用。
CURLOPT_POSTFIELDS：在HTTP中的“POST”操作。如果要传送一个文件，需要一个@开头的文件名
CURLOPT_PROXY：设置通过的HTTP代理服务器
CURLOPT_PROXYUSERPWD：连接到代理服务器的，格式为“[username]:[password]”的用户名和密码。
CURLOPT_RANDOM_FILE：设定存放SSL用到的随机数种子的文件名称
CURLOPT_RANGE：设置HTTP传输范围，可以用“X-Y”的形式设置一个传输区间，如果有多个HTTP传输，则使用逗号分隔多个值，形如："X-Y,N-M"。
CURLOPT_REFERER：设置header中"Referer: " 部分的值。
CURLOPT_SSL_CIPHER_LIST：A list of ciphers to use for SSL. For example, RC4-SHA and TLSv1 are valid cipher lists.
CURLOPT_SSLCERT：传递一个包含PEM格式证书的字符串。
CURLOPT_SSLCERTPASSWD：传递一个包含使用CURLOPT_SSLCERT证书必需的密码。
CURLOPT_SSLCERTTYPE：The format of the certificate. Supported formats are "PEM" (default), "DER", and "ENG".
CURLOPT_SSLENGINE：The identifier for the crypto engine of the private SSL key specified in CURLOPT_SSLKEY.
CURLOPT_SSLENGINE_DEFAULT：The identifier for the crypto engine used for asymmetric crypto operations.
CURLOPT_SSLKEY：The name of a file containing a private SSL key.
CURLOPT_SSLKEYPASSWD：The secret password needed to use the private SSL key specified in CURLOPT_SSLKEY.
Note: Since this option contains a sensitive password, remember to keep the PHP script it is contained within safe.
CURLOPT_SSLKEYTYPE：The key type of the private SSL key specified in CURLOPT_SSLKEY. Supported key types are "PEM" (default), "DER", and "ENG".
CURLOPT_URL：需要获取的URL地址，也可以在PHP的curl_init()函数中设置。
CURLOPT_USERAGENT：在HTTP请求中包含一个”user-agent”头的字符串。
CURLOPT_USERPWD：传递一个连接中需要的用户名和密码，格式为：“[username]:[password]”。
CURLOPT_HTTP200ALIASES：设置不再以error的形式来处理HTTP 200的响应，格式为一个数组。
CURLOPT_HTTPHEADER：设置一个header中传输内容的数组。
CURLOPT_POSTQUOTE：An array of FTP commands to execute on the server after the FTP request has been performed.
CURLOPT_QUOTE：An array of FTP commands to execute on the server prior to the FTP request.
CURLOPT_FILE：设置输出文件的位置，值是一个资源类型，默认为STDOUT (浏览器)。
CURLOPT_INFILE：在上传文件的时候需要读取的文件地址，值是一个资源类型。

CURLOPT_STDERR：设置一个错误输出地址，值是一个资源类型，取代默认的STDERR。
CURLOPT_WRITEHEADER：设置header部分内容的写入的文件地址，值是一个资源类型。
CURLOPT_HEADERFUNCTION：设置一个回调函数，这个函数有两个参数，第一个是curl的资源句柄，第二个是输出的header数据。header数据的输出必须依赖这个函数，返回已写入的数据大小。
CURLOPT_PASSWDFUNCTION：设置一个回调函数，有三个参数，第一个是curl的资源句柄，第二个是一个密码提示符，第三个参数是密码长度允许的最大值。返回密码的值。
CURLOPT_READFUNCTION：设置一个回调函数，有两个参数，第一个是curl的资源句柄，第二个是读取到的数据。数据读取必须依赖这个函数。返回读取数据的大小，比如0或者EOF。
CURLOPT_WRITEFUNCTION：设置一个回调函数，有两个参数，第一个是curl的资源句柄，第二个是写入的数据。数据写入必须依赖这个函数。返回精确的已写入数据的大小
curl_copy_handle()函数的作用是拷贝一个curl连接资源的所有内容和参数
PHP代码

```
<?php
$ch = curl_init("http://qzone.myqq.us/");
$another = curl_copy_handle($ch);
curl_exec($another);
curl_close($another);
?>
```

curl_error()函数的作用是返回一个包含当前会话错误信息的字符串。
curl_errno()函数的作用是返回一个包含当前会话错误信息的数字编号。
curl_multi_init()函数的作用是初始化一个curl批处理句柄资源。
curl_multi_add_handle()函数的作用是向curl批处理会话中添加单独的curl句柄资源。curl_multi_add_handle()函数有两个参数，第一个参数表示一个curl批处理句柄资源，第二个参数表示一个单独的curl句柄资源。
curl_multi_exec()函数的作用是解析一个curl批处理句柄，curl_multi_exec()函数有两个参数，第一个参数表示一个批处理句柄资源，第二个参数是一个引用值的参数，表示剩余需要处理的单个的curl句柄资源数量。
curl_multi_remove_handle()函数表示移除curl批处理句柄资源中的某个句柄资源，curl_multi_remove_handle()函数有两个参数，第一个参数表示一个curl批处理句柄资源，第二个参数表示一个单独的curl句柄资源。
curl_multi_close()函数的作用是关闭一个批处理句柄资源。
PHP代码

```
<?php
$ch1 = curl_init();
$ch2 = curl_init();
curl_setopt($ch1, CURLOPT_URL, "http://blog.huangchao.org/");
curl_setopt($ch1, CURLOPT_HEADER, 0);
curl_setopt($ch2, CURLOPT_URL, "http://test.huangchao.org/");
curl_setopt($ch2, CURLOPT_HEADER, 0);
$mh = curl_multi_init();
curl_multi_add_handle($mh,$ch1);
curl_multi_add_handle($mh,$ch2);
do {
curl_multi_exec($mh,$flag);
} while ($flag > 0);
curl_multi_remove_handle($mh,$ch1);
curl_multi_remove_handle($mh,$ch2);
curl_multi_close($mh);
?>
```

curl_multi_getcontent()函数的作用是在设置了CURLOPT_RETURNTRANSFER的情况下，返回获取的输出的文本流。
curl_multi_info_read()函数的作用是获取当前解析的curl的相关传输信息。
curl_multi_select()：Get all the sockets associated with the cURL extension, which can then be "selected"

php中通过curl模拟登陆discuz论坛的实现代码

libcurl同时也支持HTTPS认证、HTTP POST、HTTP PUT、 FTP 上传(这个也能通过PHP的FTP扩展完成)、HTTP 基于表单的上传、代理、cookies和用户名+密码的认证。php的curl真的是相当好用，网上一搜索相关文章都是关于curl模拟登陆的，很少人提供模拟discuz发贴的源码。

```
<?php
$discuz_url = 'http://127.0.0.1/discuz/';//论坛地址
$login_url = $discuz_url .'logging.php?action=login';//登录页地址

$post_fields = array();
//以下两项不需要修改
$post_fields['loginfield'] = 'username';
$post_fields['loginsubmit'] = 'true';
//用户名和密码，必须填写
$post_fields['username'] = 'tianxin';
$post_fields['password'] = '111111';
//安全提问
$post_fields['questionid'] = 0;
$post_fields['answer'] = '';
//@todo验证码
$post_fields['seccodeverify'] = '';
//获取表单FORMHASH
$ch = curl_init($login_url);
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
$contents = curl_exec($ch);
curl_close($ch);
preg_match('/<input\s*type="hidden"\s*name="formhash"\s*value="(.*?)"\s*\/>/i', $contents, $matches);
if(!empty($matches)) {
$formhash = $matches[1];
} else {
die('Not found the forumhash.');
}

//POST数据，获取COOKIE,cookie文件放在网站的temp目录下
$cookie_file = tempnam('./temp','cookie');
$ch = curl_init($login_url);
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $post_fields);
curl_setopt($ch, CURLOPT_COOKIEJAR, $cookie_file);
curl_exec($ch);
curl_close($ch);
//取到了关键的cookie文件就可以带着cookie文件去模拟发帖,fid为论坛的栏目ID
$send_url = $discuz_url."post.php?action=newthread&fid=2";

$ch = curl_init($send_url);
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_COOKIEFILE, $cookie_file);
$contents = curl_exec($ch);
curl_close($ch);
//这里的hash码和登陆窗口的hash码的正则不太一样，这里的hidden多了一个id属性
preg_match('/<input\s*type="hidden"\s*name="formhash"\s*id="formhash"\s*value="(.*?)"\s*\/>/i', $contents, $matches);
if(!empty($matches)) {
$formhash = $matches[1];
} else {
die('Not found the forumhash.');
}

$post_data = array();
//帖子标题
$post_data['subject'] = 'test2';
//帖子内容
$post_data['message'] = 'test2';
$post_data['topicsubmit'] = "yes";
$post_data['extra'] = '';
//帖子标签
$post_data['tags'] = 'test';
//帖子的hash码，这个非常关键！假如缺少这个hash码，discuz会警告你来路的页面不正确
$post_data['formhash']=$formhash;

$ch = curl_init($send_url);
curl_setopt($ch, CURLOPT_REFERER, $send_url); //伪装REFERER
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 0);
curl_setopt($ch, CURLOPT_COOKIEFILE, $cookie_file);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $post_data);
$contents = curl_exec($ch);
curl_close($ch);
//清理cookie文件
unlink($cookie_file);
?>
```

php中通过curl smtp发送邮件

先google了一下,发现很多问相关问题的但没有相关的解答,在phpclasses里也没有找到相关的类于是自己边看stmp的相关协议边开始尝试curl SMTP协议

这个在网上可以找到多相关的例子,可以自己实验一下使用telnet去连接mail服务器

```
$ telnet 邮箱SMTP服务地址 25
Trying 邮箱服务IP地址...
Connected to 邮箱SMTP服务地址.
Escape character is '^]'.
exchange邮箱服务器地址 Microsoft ESMTP MAIL Service ready at Sat, 2 Jun 2012 15:02:12 +0800
EHLO 127.0.0.1
-exchange邮箱服务器地址 Hello [邮箱服务IP地址]
-SIZE
-PIPELINING
-DSN
-ENHANCEDSTATUSCODES
-X-ANONYMOUSTLS
-AUTH NTLM LOGIN
-X-EXPS GSSAPI NTLM
-8BITMIME
-BINARYMIME
-CHUNKING
-XEXCH50
XRDST
AUTH LOGIN
VXNlcm5hbWU6
用户名(base64_encode)
UGFzc3dvcmQ6
密码(base64_encode)
2.7.0 Authentication successful
MAIL FROM:发件箱地址
2.1.0 Sender OK
RCPT TO:收件箱地址
2.1.5 Recipient OK
DATA
Start mail input; end with <CRLF>.<CRLF>
要发送的内容(这里的相关的规范有很多)
.
2.6.0 <0b476f30-3b96-4e3d-84d2-395a96d34000@exchange邮箱服务器地址> Queued mail for delivery
QUIT
2.0.0 Service closing transmission channel
Connection closed by foreign host.
```

php测试代码:

```php
<?php
header("content-type:text/html;charset=utf-8");
$smtp = array(
"url" => "邮箱SMTP服务器地址",
"port" => "邮箱SMTP服务器端口", // 一般为25
"username" => "用户名",
"password" => "密码",
"from" => "发件地址",
"to" => "收件地址",
"subject" => "测试一下标题",
"body" => "测试一下内容"
);
$CRLF = "\r\n";
$test = "";
$curl = curl_init();
curl_setopt($curl, CURLOPT_URL, $smtp['url']);
curl_setopt($curl, CURLOPT_PORT, $smtp['port']);
curl_setopt($curl, CURLOPT_TIMEOUT,10);
function inlineCode($str){
$str = trim($str);
return $str?'=?UTF-8?B?'.base64_encode($str).'?= ':'';
}
function buildHeader($headers){
$ret = '';
foreach($headers as $k=>$v){
$ret.=$k.': '.$v."\n";
}
return $ret;
}
//
$header = array(
'Return-path'=>'<'.$smtp['from'].'>',
'Date'=>date('r'),
'From'=> '<'.$smtp['from'].'>',
'MIME-Version'=>'1.0',
'Subject'=>inlineCode($smtp['subject']),
'To'=>$smtp['to'],
'Content-Type'=>'text/html; charset=UTF-8; format=flowed',
'Content-Transfer-Encoding'=>'base64'
);
$data = buildHeader($header).$CRLF.chunk_split(base64_encode($smtp['body']));
$content = "EHLO ".$smtp["url"].$CRLF; // 先得hello一下
$content .= "AUTH LOGIN".$CRLF.base64_encode($smtp["username"]).$CRLF.base64_encode($smtp["password"]).$CRLF; // 验证登陆
$content .= "MAIL FROM:".$smtp["from"].$CRLF; // 发件地址
$content .= "RCPT TO:".$smtp["to"].$CRLF; // 收件地址
$content .= "DATA".$CRLF.$data.$CRLF."".$CRLF; // 发送内容
$content .= "QUIT".$CRLF; // 退出
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true); // curl接收返回数据
curl_setopt($curl, CURLOPT_CUSTOMREQUEST, $content);
$test = curl_exec($curl);
var_dump($test);
echo "<br/>\r\n";
var_dump($content);
// 结束
curl_close($curl);
```

以上只是测试的php
包测试+修改花了近6个小时让产品的代码兼容了fsockopen和curl
以后有空写个兼容fsockopen和curl简单发送邮件的smtp类

PHP curl 并发最佳实践代码分享

本文将探讨两种具体的实现方法, 并对不同的方法做简单的性能对比.

1. 经典cURL并发机制及其存在的问题

经典的cURL实现机制在网上很容易找到, 比如参考PHP在线手册的如下实现方式:

```
function classic_curl($urls, $delay) {
$queue = curl_multi_init();
$map = array();

foreach ($urls as $url) {
// create cURL resources
$ch = curl_init();

// set URL and other appropriate options
curl_setopt($ch, CURLOPT_URL, $url);

curl_setopt($ch, CURLOPT_TIMEOUT, 1);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_NOSIGNAL, true);

// add handle
curl_multi_add_handle($queue, $ch);
$map[$url] = $ch;
}

$active = null;

// execute the handles
do {
$mrc = curl_multi_exec($queue, $active);
} while ($mrc == CURLM_CALL_MULTI_PERFORM);

while ($active > 0 && $mrc == CURLM_OK) {
if (curl_multi_select($queue, 0.5) != -1) {
do {
$mrc = curl_multi_exec($queue, $active);
} while ($mrc == CURLM_CALL_MULTI_PERFORM);
}
}

$responses = array();
foreach ($map as $url=>$ch) {
$responses[$url] = callback(curl_multi_getcontent($ch), $delay);
curl_multi_remove_handle($queue, $ch);
curl_close($ch);
}

curl_multi_close($queue);
return $responses;
}
```

首先将所有的URL压入并发队列, 然后执行并发过程, 等待所有请求接收完之后进行数据的解析等后续处理. 在实际的处理过程中, 受网络传输的影响, 部分URL的内容会优先于其他URL返回, 但是经典cURL并发必须等待最慢的那个URL返回之后才开始处理, 等待也就意味着CPU的空闲和浪费. 如果URL队列很短, 这种空闲和浪费还处在可接受的范围, 但如果队列很长, 这种等待和浪费将变得不可接受.

2. 改进的Rolling cURL并发方式

仔细分析不难发现经典cURL并发还存在优化的空间, 优化的方式时当某个URL请求完毕之后尽可能快的去处理它, 边处理边等待其他的URL返回, 而不是等待那个最慢的接口返回之后才开始处理等工作, 从而避免CPU的空闲和浪费. 闲话不多说, 下面贴上具体的实现:

```
function rolling_curl($urls, $delay) {
$queue = curl_multi_init();
$map = array();

foreach ($urls as $url) {
$ch = curl_init();

curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_TIMEOUT, 1);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_NOSIGNAL, true);

curl_multi_add_handle($queue, $ch);
$map[(string) $ch] = $url;
}

$responses = array();
do {
while (($code = curl_multi_exec($queue, $active)) == CURLM_CALL_MULTI_PERFORM) ;

if ($code != CURLM_OK) { break; }

// a request was just completed -- find out which one
while ($done = curl_multi_info_read($queue)) {

// get the info and content returned on the request
$info = curl_getinfo($done['handle']);
$error = curl_error($done['handle']);
$results = callback(curl_multi_getcontent($done['handle']), $delay);
$responses[$map[(string) $done['handle']]] = compact('info', 'error', 'results');

// remove the curl handle that just completed
curl_multi_remove_handle($queue, $done['handle']);
curl_close($done['handle']);
}

// Block for data in / output; error handling is done by curl_multi_exec
if ($active > 0) {
curl_multi_select($queue, 0.5);
}

} while ($active);

curl_multi_close($queue);
return $responses;
}
```

3. 两种并发实现的性能对比

改进前后的性能对比试验在LINUX主机上进行, 测试时使用的并发队列如下:
http://item.taobao.com/item.htm?id=14392877692 http://item.taobao.com/item.htm?id=16231676302 http://item.taobao.com/item.htm?id=17037160462 http://item.taobao.com/item.htm?id=5522416710 http://item.taobao.com/item.htm?id=16551116403 http://item.taobao.com/item.htm?id=14088310973

简要说明下实验设计的原则和性能测试结果的格式: 为保证结果的可靠, 每组实验重复20次, 在单次实验中, 给定相同的接口URL集合, 分别测量Classic(指经典的并发机制)和Rolling(指改进后的并发机制)两种并发机制的耗时(秒为单位), 耗时短者胜出(Winner), 并计算节省的时间(Excellence, 秒为单位)以及性能提升比例(Excel. %). 为了尽量贴近真实的请求而又保持实验的简单, 在对返回结果的处理上只是做了简单的正则表达式匹配, 而没有进行其他复杂的操作. 另外, 为了确定结果处理回调对性能对比测试结果的影响, 可以使用usleep模拟现实中比较负责的数据处理逻辑(如提取, 分词, 写入文件或数据库等).

性能测试中用到的回调函数为:

```
function callback($data, $delay) {
preg_match_all('/<h3>(.+)<\/h3>/iU', $data, $matches);
usleep($delay);
return compact('data', 'matches');
}
```

数据处理回调无延迟时: Rolling Curl略优, 但性能提升效果不明显.
数据处理回调延迟5毫秒: Rolling Curl完胜, 性能提升40%左右.
通过上面的性能对比, 在处理URL队列并发的应用场景中Rolling cURL应该是更加的选择, 并发量非常大(1000+)时, 可以控制并发队列的最大长度, 比如20, 每当1个URL返回并处理完毕之后立即加入1个尚未请求的URL到队列中, 这样写出来的代码会更加健壮, 不至于并发数太大而卡死或崩溃. 详细的实现请参考: http://code.google.com/p/rolling-curl/

CURL的学习和应用(附多线程实现)

curl安装：

windows下面的安装

：修改php.ini文件的设置，找到php_curl.dll

//取消下在的注释extension=php_curl.dll

linux下面安装：

```
# wget http://curl.haxx.se/download/curl-7.17.1.tar.gz # tar zxvf curl-7.17.1.tar.gz  //解压
#cd curl-7.17.1
# ./configure –prefix=/usr/local/curl
# make
# make install
```

这是安装php之前安装的方法.

**********phpinf查看是否加载成功！

使用curl的POST数据飞信接口

用curl 写了飞信接口吧，网上有很多，这里只是做个测试

```
$username = 13800138000;
$password = 123456;
$sendto = 13912345678;
$message = "测试一个试试看！";
$curlPost = 'username='.urlencode($username).'&password='.urlencode($password).'&sendto='.urlencode($sendto).'&message='.urlencode($message).";
$ch = curl_init();//初始化curl
curl_setopt($ch,CURLOPT_URL,'http://sms.api.bz/fetion.php');//抓取指定网页
curl_setopt($ch, CURLOPT_HEADER, 0);//设置header
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);//要求结果为字符串且输出到屏幕上
curl_setopt($ch, CURLOPT_POST, 1);//post提交方式
curl_setopt($ch, CURLOPT_POSTFIELDS, $curlPost);
$data = curl_exec($ch);//运行curl
curl_close($ch);
print_r($data);//输出结果
```

返回的结果是：短信已提交到发送队列！

飞信接口的地址是http://sms.api.bz/

飞信接口模式： http://sms.api.bz/fetion.php?username=您的移动飞信登录手机号 &password=您的移动飞信登录密码

&sendto=接收短信的飞信好友手机号

&message=短信内容

格式：http://sms.api.bz/fetion.php?username=13800138000&password=123456&sendto=13912345678&message=短信内容

注意要保持utf-8格式的，这点我犯错了

总结一下使用curl方法：

初始化curl

使用curl_setopt设置目标url，和其他选项，这些选项方法详细参考：http://cn2.php.net/manual/zh/ref.curl.php

curl_exec，执行curl

执行后，关闭curl

最后一步就是输出

一个最要的curl函数：curl_getinfo

curl_getinfo (resource $ch [, int $opt = 0])

```
<?php
/*curl实例
*/
$curl = curl_init();
// 设置你需要抓取的URL
curl_setopt($curl, CURLOPT_URL, 'http://www.baidu.com');
// 设置header
curl_setopt($curl, CURLOPT_HEADER, 0);
// 设置cURL 参数，要求结果保存到字符串中还是输出到屏幕上。
curl_setopt($curl, CURLOPT_RETURNTRANSFER, 1);
// 运行cURL，请求网页
$data = curl_exec($curl);
if($data === false){
 echo curl_error($curl);exit;
}
$info = curl_getinfo($curl);
// 关闭URL请求
curl_close($curl);

// 显示获得的数据
var_dump($info);
var_dump($data);
```

可以返回：

URLINFO_EFFECTIVE_URL – 最后一个有效的URL地址
CURLINFO_HTTP_CODE – 最后一个收到的HTTP代码
CURLINFO_FILETIME – 远程获取文档的时间，如果无法获取，则返回值为“-1”
CURLINFO_TOTAL_TIME – 最后一次传输所消耗的时间
CURLINFO_NAMELOOKUP_TIME – 名称解析所消耗的时间
CURLINFO_CONNECT_TIME – 建立连接所消耗的时间
CURLINFO_PRETRANSFER_TIME – 从建立连接到准备传输所使用的时间
CURLINFO_STARTTRANSFER_TIME – 从建立连接到传输开始所使用的时间
CURLINFO_REDIRECT_TIME – 在事务传输开始前重定向所使用的时间
CURLINFO_SIZE_UPLOAD – 上传数据量的总值
CURLINFO_SIZE_DOWNLOAD – 下载数据量的总值
CURLINFO_SPEED_DOWNLOAD – 平均下载速度
CURLINFO_SPEED_UPLOAD – 平均上传速度
CURLINFO_HEADER_SIZE – header部分的大小
CURLINFO_HEADER_OUT – 发送请求的字符串
CURLINFO_REQUEST_SIZE – 在HTTP请求中有问题的请求的大小
CURLINFO_SSL_VERIFYRESULT – 通过设置CURLOPT_SSL_VERIFYPEER返回的SSL证书验证请求的结果
CURLINFO_CONTENT_LENGTH_DOWNLOAD – 从Content-Length: field中读取的下载内容长度
CURLINFO_CONTENT_LENGTH_UPLOAD – 上传内容大小的说明
CURLINFO_CONTENT_TYPE – 下载内容的Content-Type:值，NULL表示服务器没有发送有效的Content-Type: header

使用curl实现多线程

curl一般用来抓取网页，第二种就是get或者post数据，第三种应用就是实现PHP的多线程任务
下面来实现多线程的

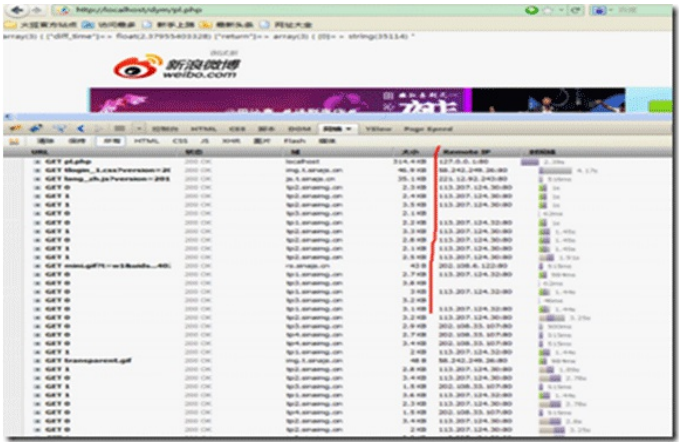
```php
<?php
/*
curl 多线程抓取
*/
 /**
     * curl 多线程
     *
     * @param array $array 并行网址
     * @param int $timeout 超时时间
     * @return array
     */
 function Curl_http($array,$timeout){
  $res = array();
  $mh = curl_multi_init();//创建多个curl语柄
 $startime = getmicrotime();
  foreach($array as $k=>$url){
   $conn[$k]=curl_init($url);

        curl_setopt($conn[$k], CURLOPT_TIMEOUT, $timeout);//设置超时时间
        curl_setopt($conn[$k], CURLOPT_USERAGENT, 'Mozilla/5.0 (compatible; MSIE 5.01; Windows NT 5.0)');
        curl_setopt($conn[$k], CURLOPT_MAXREDIRS, 7);//HTTp定向级别
        curl_setopt($conn[$k], CURLOPT_HEADER, 0);//这里不要header，加块效率
        curl_setopt($conn[$k], CURLOPT_FOLLOWLOCATION, 1); // 302 redirect
        curl_setopt($conn[$k],CURLOPT_RETURNTRANSFER,1);
        curl_multi_add_handle ($mh,$conn[$k]);
  }
  //防止死循环耗死cpu 这段是根据网上的写法
  do {
   $mrc = curl_multi_exec($mh,$active);//当无数据，active=true
  } while ($mrc == CURLM_CALL_MULTI_PERFORM);//当正在接受数据时
  while ($active and $mrc == CURLM_OK) {//当无数据时或请求暂停时，active=true
   if (curl_multi_select($mh) != -1) {
    do {
     $mrc = curl_multi_exec($mh, $active);
    } while ($mrc == CURLM_CALL_MULTI_PERFORM);
   }
  }

  foreach ($array as $k => $url) {
     curl_error($conn[$k]);
       $res[$k]=curl_multi_getcontent($conn[$k]);//获得返回信息
       $header[$k]=curl_getinfo($conn[$k]);//返回头信息
       curl_close($conn[$k]);//关闭语柄
       curl_multi_remove_handle($mh ，$conn[$k]);   //释放资源
  }

  curl_multi_close($mh);
  $endtime = getmicrotime();
  $diff_time = $endtime - $startime;

  return array('diff_time'=>$diff_time,
      'return'=>$res,
     'header'=>$header
     );

 }
 //计算当前时间
 function getmicrotime() {
     list($usec, $sec) = explode(" ",microtime());
     return ((float)$usec + (float)$sec);
 }

 //测试一下，curl 三个网址
 $array = array(
    "http://www.weibo.com/",
    "http://www.renren.com/",
    "http://www.qq.com/"
    );
 $data = Curl_http($array,'10');//调用
 var_dump($data);//输出

?>
```

关于do while的那段解释：

因为$active要等全部url数据接受完毕才变成false，所以这里用到了curl_multi_exec的返回值判断是否还有数据，
当有数 据的时候就不停调用curl_multi_exec，暂时没有数据就进入select阶段，新数据一来就可以被唤醒继续执行。
这里的好处就是CPU的无谓 消耗没有了。更详细的说明：http://hi.baidu.com/%D4%C2%D2%B9%C4%FD%ED%F8/blog/item/9dfcf4fbe6b84374024f563d.html

这个多线程的写法步骤：
第一步：调用curl_multi_init
第二步：循环调用curl_multi_add_handle
这一步需要注意的是，curl_multi_add_handle的第二个参数是由curl_init而来的子handle。
第三步：持续调用curl_multi_exec
第四步：根据需要循环调用curl_multi_getcontent获取结果
第五步：调用curl_multi_remove_handle，并为每个字handle调用curl_close
第六步：调用curl_multi_close

多线程的测试效果图：



总结：36个http请求，从执行的是时间顺序上来看，三个网站的ip交叉，说明是同时并发的！

---

linux命令下的curl

几种常见的使用方式：

下载作用：

直接下载 相当于wget

curl -o 1.jpg http://cgi2.tky.3web.ne.jp/~zzh/screen1.JPG 批量下载screen1.JPG–screen10.JPG

curl -O http://cgi2.tky.3web.ne.jp/~zzh/screen[1-10].JPG 断点下载

curl -c -O http://cgi2.tky.3wb.ne.jp/~zzh/screen1.JPG 反向代理功能

curl -x 123.45.67.89:1080 -o page.html http://www.yahoo.com 显示头文件

curl -I www.sina.com

php curl模仿ftp文件上传代码

php教程 curl模仿ftp文件上传代码

```
<body>
<form action="curlupload.php" method="post" enctype="multipart/form-data">
<div>
<label for="upload">select file</label>
<input name="upload" type="file" />
<input type="submit" name="submit" value="upload" />
</div>
</form>
</body>
</html>
<?
if (isset($_post['submit'])) {
if (!empty($_files['upload']['name'])) {
$ch = curl_init();
$localfile = $_files['upload']['tmp_name'];
$fp = fopen($localfile, 'r');
curl_setopt($ch, curlopt_url, 'ftp://ftp_login:password@ftp.domain.com/'.$_files['upload']['name']);
curl_setopt($ch, curlopt_upload, 1);
curl_setopt($ch, curlopt_infile, $fp);
curl_setopt($ch, curlopt_infilesize, filesize($localfile));
curl_exec ($ch);
$error_no = curl_errno($ch);
curl_close ($ch);
if ($error_no == 0) {
$error = 'file uploaded succesfully.';
} else {
$error = 'file upload error.';
}
} else {
$error = 'please select a file.';
}
}
?>
```

好了下面封闭成类了

```
<?php
class curl_ftp
{

private $ftpname; //ftp用户名
private $ftppaw; //ftp密码
private $urlftp; //ftp地址
private $filename; //文件名

public __construct($name, $password, $ftp)
{
$this->ftpname = $name;
$this->ftppaw = $password;
$this->urlftp = $ftp;
// $this->filename = $filename;
}

public function getftp()
{
if (isset($_post['submit']))
{
if (!empty($_files['upload']['name']))
{
$ch = curl_init();
$this->filename = $_files['upload']['tmp_name'];
$fp = fopen($this->filename, 'r');
curl_setopt($ch, curlopt_url, $this->ftp.$this->filename);
curl_setopt($ch, curlopt_userpwd, "$name:password");
curl_setopt($ch, curlopt_upload, 1);
curl_setopt($ch, curlopt_infile, $fp);
curl_setopt($ch, curlopt_infilesize, filesize($this->filename));
```

```
curl_exec ($ch);
$error_no = curl_errno($ch);
curl_close ($ch);
if ($error_no == 0)
{
$error = '文件上传成功';
}
else
{
$error = '文件上传失败';
}
}
else
{
$error = '未选择文件';
}
}
}
}
```

php使用curl来获取远程图片

本文章来介绍php使用curl来获取远程图片实现方法，有需要了解采集远程图片的朋友不防进入参考。curl要求php环境支持才行. 可以运行phpinfo()函数是否支持,一般要将php.ini中;extension=php_curl.dll前的;去掉.重新启动IIS或者APACHE就可以了.

```
/*
*@通过curl方式获取指定的图片到本地
*@ 完整的图片地址
*@ 要存储的文件名
*/
function getImg($url = "", $filename = "")
{
 //去除URL连接上面可能的引号
  //$url = preg_replace( '/(?:^["']+|["']/]+$)/', '', $url );
  $hander = curl_init();
  $fp = fopen($filename,'wb');
  curl_setopt($hander,CURLOPT_URL,$url);
  curl_setopt($hander,CURLOPT_FILE,$fp);
  curl_setopt($hander,CURLOPT_HEADER,0);
  curl_setopt($hander,CURLOPT_FOLLOWLOCATION,1);
  //curl_setopt($hander,CURLOPT_RETURNTRANSFER,false);//以数据流的方式返回数据,当为false是直接显示出来
  curl_setopt($hander,CURLOPT_TIMEOUT,60);
  curl_exec($hander);
  curl_close($hander);
  fclose($fp);
  Return true;
}
调用时，直接getImg("/logo.jpg","upload/image.jpg")
```

实现代码2

代码如下:

```
<?php
$url = "图片绝对地址/thumbnail.jpg";
$filename = 'curl.jpg';
getImg($url, $filename);
/*
  *@通过curl方式获取制定的图片到本地
  *@ 完整的图片地址
  *@ 要存储的文件名
 */
function getImg($url = "", $filename = "") {
    if(is_dir(basename($filename))) {
        echo "The Dir was not exits";
        return false;
    }
    //去除URL连接上面可能的引号
    $url = preg_replace( '/(?:^["']+|["']/]+$)/', '', $url );
    $hander = curl_init();
    $fp = fopen($filename,'wb');
    curl_setopt($hander,CURLOPT_URL,$url);
    curl_setopt($hander,CURLOPT_FILE,$fp);
    curl_setopt($hander,CURLOPT_HEADER,0);
    curl_setopt($hander,CURLOPT_FOLLOWLOCATION,1);
    //curl_setopt($hander,CURLOPT_RETURNTRANSFER,false);//以数据流的方式返回数据,当为false是直接显示出来
    curl_setopt($hander,CURLOPT_TIMEOUT,60);
    /*$options = array(
        CURLOPT_URL=> '/thum-f3ccdd27d2000e3f9255a7e3e2c4880020110622095243.jpg',
        CURLOPT_FILE => $fp,
        CURLOPT_HEADER => 0,
        CURLOPT_FOLLOWLOCATION => 1,
        CURLOPT_TIMEOUT => 60
    );
    curl_setopt_array($hander, $options);
    */
    curl_exec($hander);
    curl_close($hander);
    fclose($fp);
    return  true;
}
 ?>
```

curl_setopt 为CURL调用设置一个选项

bool curl_setopt (int ch, string option, mixed value)

curl_setopt()函数将为一个CURL会话设置选项。option参数是你想要的设置，value是这个选项给定的值。

下列选项的值将被作为长整形使用(在option参数中指定)：

CURLOPT_INFILESIZE: 当你上传一个文件到远程站点，这个选项告诉PHP你上传文件的大小。

CURLOPT_VERBOSE: 如果你想CURL报告每一件意外的事情，设置这个选项为一个非零值。

CURLOPT_HEADER: 如果你想把一个头包含在输出中，设置这个选项为一个非零值。

CURLOPT_NOPROGRESS: 如果你不会PHP为CURL传输显示一个进程条，设置这个选项为一个非零值。

注意：PHP自动设置这个选项为非零值，你应该仅仅为了调试的目的来改变这个选项。

CURLOPT_NOBODY: 如果你不想在输出中包含body部分，设置这个选项为一个非零值。

CURLOPT_FAILONERROR: 如果你想让PHP在发生错误(HTTP代码返回大于等于300)时，不显示，设置这个选项为一人非零值。默认行为是返回一个正常页，忽略代码。

CURLOPT_UPLOAD: 如果你想让PHP为上传做准备，设置这个选项为一个非零值。

CURLOPT_POST: 如果你想PHP去做一个正规的HTTP POST，设置这个选项为一个非零值。这个POST是普通的 application/x-www-from-urlencoded 类型，多数被HTML表单使用。

CURLOPT_FTPLISTONLY: 设置这个选项为非零值，PHP将列出FTP的目录名列表。

CURLOPT_FTPAPPEND: 设置这个选项为一个非零值，PHP将应用远程文件代替覆盖它。

CURLOPT_NETRC: 设置这个选项为一个非零值，PHP将在你的 ~./netrc 文件中查找你要建立连接的远程站点的用户名及密码。
CURLOPT_FOLLOWLOCATION: 设置这个选项为一个非零值(象 “Location: ”)的头，服务器会把它当做HTTP头的一部分发送(注意这是递归的，PHP将发送形如 “Location: ”的头)。
CURLOPT_PUT: 设置这个选项为一个非零值去用HTTP上传一个文件。要上传这个文件必须设置CURLOPT_INFILE和CURLOPT_INFILESIZE选项.
CURLOPT_MUTE: 设置这个选项为一个非零值，PHP对于CURL函数将完全沉默。
CURLOPT_TIMEOUT: 设置一个长整形数，作为最大延续多少秒。
CURLOPT_LOW_SPEED_LIMIT: 设置一个长整形数，控制传送多少字节。
CURLOPT_LOW_SPEED_TIME: 设置一个长整形数，控制多少秒传送CURLOPT_LOW_SPEED_LIMIT规定的字节数。
CURLOPT_RESUME_FROM: 传递一个包含字节偏移地址的长整形参数，(你想转移到的开始表单)。
CURLOPT_SSLVERSION: 传递一个包含SSL版本的长参数。默认PHP将被它自己努力的确定，在更多的安全中你必须手工设置。
CURLOPT_TIMECONDITION: 传递一个长参数，指定怎么处理CURLOPT_TIMEVALUE参数。你可以设置这个参数为TIMECOND_IFMODSINCE 或 TIMECOND_ISUNMODSINCE。这仅用于HTTP。
CURLOPT_TIMEVALUE: 传递一个从1970-1-1开始到现在的秒数。这个时间将被CURLOPT_TIMEVALUE选项作为指定值使用，或被默认TIMECOND_IFMODSINCE使用。
下列选项的值将被作为字符串：
CURLOPT_URL: 这是你想用PHP取回的URL地址。你也可以在用curl_init()函数初始化时设置这个选项。
CURLOPT_USERPWD: 传递一个形如[username]:[password]风格的字符串,作用PHP去连接。
CURLOPT_PROXYUSERPWD: 传递一个形如[username]:[password] 格式的字符串去连接HTTP代理。
CURLOPT_RANGE: 传递一个你想指定的范围。它应该是”X-Y”格式，X或Y是被除外的。HTTP传送同样支持几个间隔，用逗句来分隔(X-Y,N-M)。
CURLOPT_POSTFIELDS: 传递一个作为HTTP “POST”操作的所有数据的字符串。
CURLOPT_REFERER: 在HTTP请求中包含一个”referer”头的字符串。
CURLOPT_USERAGENT: 在HTTP请求中包含一个”user-agent”头的字符串。
CURLOPT_FTPPORT: 传递一个包含被ftp “POST”指令使用的IP地址。这个POST指令告诉远程服务器去连接我们指定的IP地址。 这个字符串可以是一个IP地址，一个主机名，一个网络界面名(在UNIX下)，或是‘-’(使用系统默认IP地址)。
CURLOPT_COOKIE: 传递一个包含HTTP cookie的头连接。
CURLOPT_SSLCERT: 传递一个包含PEM格式证书的字符串。
CURLOPT_SSLCERTPASSWD: 传递一个包含使用CURLOPT_SSLCERT证书必需的密码。
CURLOPT_COOKIEFILE: 传递一个包含cookie数据的文件的名字的字符串。这个cookie文件可以是Netscape格式，或是堆存在文件中的HTTP风格的头。
CURLOPT_CUSTOMREQUEST: 当进行HTTP请求时，传递一个字符被GET或HEAD使用。为进行DELETE或其它操作是有益的，更Pass a string to be used instead of GET or HEAD when doing an HTTP request. This is useful for doing or another, more obscure, HTTP request.
注意: 在确认你的服务器支持命令先不要去这样做。

PHP Curl多线程原理实例详解

给各位介绍一下Curl多线程实例与原理。不对之处请指教

相信许多人对php手册中语焉不详的curl_multi一族的函数头疼不已，它们文档少，给的例子 更是简单的让你无从借鉴，我也曾经找了许多网页，都没见一个完整的应用例子。

curl_multi_add_handle

curl_multi_close

curl_multi_exec

curl_multi_getcontent

curl_multi_info_read

curl_multi_init

curl_multi_remove_handle

curl_multi_select

一般来说，想到要用这些函数时，目的显然应该是要同时请求多个url，而不是一个一个依次请求，否则不如自己循环去调curl_exec好了。

步骤总结如下：

第一步：调用curl_multi_init

第二步：循环调用curl_multi_add_handle

这一步需要注意的是，curl_multi_add_handle的第二个参数是由curl_init而来的子handle。

第三步：持续调用curl_multi_exec

第四步：根据需要循环调用curl_multi_getcontent获取结果

第五步：调用curl_multi_remove_handle，并为每个字handle调用curl_close

第六步：调用curl_multi_close

这里有PHP手册上的例子：

```
<?php
// 创建一对cURL资源
$ch1 = curl_init();
$ch2 = curl_init();

// 设置URL和相应的选项
curl_setopt($ch1, CURLOPT_URL, "http://www.jb51.net/");
curl_setopt($ch1, CURLOPT_HEADER, 0);
curl_setopt($ch2, CURLOPT_URL, "http://www.php.net/");
curl_setopt($ch2, CURLOPT_HEADER, 0);

// 创建批处理cURL句柄
$mh = curl_multi_init();

// 增加2个句柄
curl_multi_add_handle($mh,$ch1);
curl_multi_add_handle($mh,$ch2);

$active = null;
// 执行批处理句柄
do {
    $mrc = curl_multi_exec($mh, $active);
} while ($mrc == CURLM_CALL_MULTI_PERFORM);

while ($active && $mrc == CURLM_OK) {
    if (curl_multi_select($mh) != -1) {
        do {
            $mrc = curl_multi_exec($mh, $active);
        } while ($mrc == CURLM_CALL_MULTI_PERFORM);
    }
}

// 关闭全部句柄
curl_multi_remove_handle($mh, $ch1);
curl_multi_remove_handle($mh, $ch2);
curl_multi_close($mh);
?>
```

整个使用过程差不多就是这样，但是，这个简单代码有个致命弱点，就是在do循环的那段，在整个url请求期间是个死循环，它会轻易导致CPU占用100%。

现在我们来改进它，这里要用到一个几乎没有任何文档的函数curl_multi_select了，虽然C的curl库对select有说明，但是，php里的接口和用法确与C中有不同。

把上面do的那段改成下面这样：

```
do {
                $mrc = curl_multi_exec($mh,$active);
            } while ($mrc == CURLM_CALL_MULTI_PERFORM);
            while ($active and $mrc == CURLM_OK) {
                if (curl_multi_select($mh) != -1) {
                    do {
                        $mrc = curl_multi_exec($mh, $active);
                    } while ($mrc == CURLM_CALL_MULTI_PERFORM);
                }
            }
```

因为$active要等全部url数据接受完毕才变成false，所以这里用到了curl_multi_exec的返回值判断是否还有数据，当有数据的时候就不停调用curl_multi_exec，暂时没有数据就进入select阶段，新数据一来就可以被唤醒继续执行。这里的好处就是CPU的无谓消耗没有了。

另外：还有一些细节的地方可能有时候要遇到：

控制每一个请求的超时时间，在curl_multi_add_handle之前通过curl_setopt去做：

curl_setopt($ch, CURLOPT_TIMEOUT, $timeout);

判断是否超时了或者其他错误，在curl_multi_getcontent之前用：curl_error($conn[$i]);

本类的特点：
运行非常稳定。
设置一个并发就会始终以这个并发数进行工作，即使通过回调函数添加任务也不影响。
CPU占用极低，绝大部分CPU消耗在用户的回调函数上。
内存利用率高，任务数量较多（15W个任务占用内存会超过256M）可以使用回调函数添加任务，个数自定。
能够最大限度的占用带宽。
链式任务，比如一个任务需要从多个不同的地址采集数据，可以通过回调一气呵成。
能够对CURL错误进行多次尝试，次数自定（大并发一开始容易产生CURL错误，网络状况或对方服务器稳定性也有可能产生CURL错误）。
回调函数相当灵活，可以多种类型任务同时进行（比如下载文件，抓取网页，分析404可以在一个PHP进程中同时进行）。
可以非常容易的定制任务类型，比如检查404，获取redirect的最后url等。
可以设置缓存，挑战产品节操。
不足：
不能充分利用多核CPU（可以开多个进程解决，需要自己处理任务分割等逻辑）。
最大并发500（或512？），经过测试是CURL 内部限制，超过最大并发会导致总是返回失败。
目前没有断点续传功能。
目前任务是原子性的，不能对一个大文件分为几部分分别开线程下载。

curl不使用文件存取cookie php使用curl获取cookie示例

```
/*-----保存COOKIE-----*/
$url = 'www.xxx.com'; //url地址
$post = "id=user&pwd=123456"; //POST数据
$ch = curl_init($url); //初始化
curl_setopt($ch,CURLOPT_HEADER,1); //将头文件的信息作为数据流输出
curl_setopt($ch,CURLOPT_RETURNTRANSFER,1); //返回获取的输出文本流
curl_setopt($ch,CURLOPT_POSTFIELDS,$post); //发送POST数据
$content = curl_exec($ch); //执行curl并赋值给$content
preg_match('/Set-Cookie:(.*);iU',$content,$str); //正则匹配
$cookie = $str[1]; //获得COOKIE（SESSIONID）
curl_close($ch); //关闭curl

/*-----使用COOKIE-----*/
curl_setopt($ch,CURLOPT_COOKIE,$cookie);
```

php使用curl抓取qq空间的访客信息示例

config.php

```
<?php
define('APP_DIR', dirname(__FILE__));
define('COOKIE_FILE', APP_DIR . '/app.cookie.txt'); //会话记录文件
define('VISITOR_CAPTURE_INTERVAL', 3); //QQ采集间隔
define('VISITOR_DATA_UPLOAD_INTERVAL', '');
define('THIS_TIME', time());

define('REQUEST_TIMEOUT', 20); //请求超时20秒
define('END_LINE', "\n");
define('DEBUG', true); //开启调试

$login_users = array(
    array('user' => '2064556526', 'password' => '909124951'),
    array('user' => '483258700', 'password' => '909124951'),
    array('user' => '1990270522', 'password' => '909124951'),
    array('user' => '2718711637', 'password' => '909124951'),
    array('user' => '2841076562', 'password' => '909124951'),
);
```

qy.visitor.php

```
<?php
include('./config.php');
include(APP_DIR . '/qy.visitor.php');

$sessions = array();
$user = $login_users[array_rand($login_users)];

$visitor_capture = new QQVisitorCapture($user['user'], $user['password'], COOKIE_FILE, REQUEST_TIMEOUT, DEBUG, END_LINE);

$visitors = $visitor_capture->getVisitorInfo();

if (empty($visitors)) {
    $this->clearCookies(true);
} else {
    $cckf_service = new CCKFService(SECURITY_KEY,SERVICE_ID,SERVICE_ADDRESS,'', REQUEST_TIMEOUT, DEBUG, END_LINE);
}
```

qy.class.php

```
<?php
class Trace
{
    public static function nl($num = 1)
    {
        $str = '';
        for ($i = 0; $i < $num; $i++) {
            $str .= "\n";
        }
        return $str;
    }

    public static function br($num = 1)
    {
        $str = '';
        for ($i = 0; $i < $num; $i++) {
            $str .= "<br/>";
        }
        return $str;
    }

    public static function write($content, $end_line, $title = null)
    {
        $close = '^^^^^^^^^^^^^^^^';

        if ($title) {
            $start = '--------' . $title . '---------';
        } else {
            $start = '----------------';
        }

        echo $start . $end_line;

        if (is_array($content)) {
            print_r($content);
            echo $end_line;
        } else {
            echo $content;
            echo $end_line;
        }

        if (empty($content)) {
            echo $end_line;
        } else {
```

```
            echo $close . $end_line;
        }
    }

}
class Utils
{

    public static function getMicroTime()
    {
        list($mic, $time) = explode(" ", microtime());
        return intval($time) + floatval(sprintf('%.3f', $mic));
    }

    public static function getUTCMilliseconds()
    {
        return round(rand(1, 9) / 10 * 2147483647) * round(1, 999) % 10000000000;
    }

    public static function decodeURIComponent($content)
    {
        return urldecode(preg_replace("/\\\\x([0-9a-z]{2,3})/i", "%$1", $content));
    }

    public static function  jsRandom()
    {
        list($mic, $time) = explode(" ", microtime());
        return $mic;
    }

    function loginJsTime()
    {
        list($mic, $time) = explode(" ", microtime());
        return $time . sprintf('%3d', $mic * 1000);

    }

    protected static function utf8_unicode($c)
    {
        switch (strlen($c)) {
            case 1:
                return ord($c);
            case 2:
                $n = (ord($c[0]) & 0x3f) << 6;
                $n += ord($c[1]) & 0x3f;
                return $n;
            case 3:
                $n = (ord($c[0]) & 0x1f) << 12;
                $n += (ord($c[1]) & 0x3f) << 6;
                $n += ord($c[2]) & 0x3f;
                return $n;
            case 4:
                $n = (ord($c[0]) & 0x0f) << 18;
                $n += (ord($c[1]) & 0x3f) << 12;
                $n += (ord($c[2]) & 0x3f) << 6;
                $n += ord($c[3]) & 0x3f;
                return $n;
        }
    }

    public static function  getGTK($str)
    {
        $hash = 5381;
        for ($i = 0, $len = strlen($str); $i < $len; ++$i) {
            $hash += ($hash << 5) + self::utf8_unicode($str[$i]);
        }
        return $hash & 2147483647;
    }

    protected static function hexchar2bin($str)
    {
        $arr = '';
        $temp = null;
        for ($i = 0; $i < strlen($str); $i = $i + 2) {
            $arr .= "\\x" . substr($str, $i, 2);
        }
        eval('$temp="' . $arr . '";');
        return $temp;
    }

    protected static function getUid($uid)
    {
        $temp = null;
        eval('$temp="' . $uid . '";');
        return $temp;
    }

    public static function getEncryption($password, $uin, $vcode)
    {
        $uin = self::getUid($uin);
        $str1 = self::hexchar2bin(strtoupper(md5($password)));
        $str2 = strtoupper(md5($str1 . $uin));
        return strtoupper(md5($str2 . strtoupper($vcode)));
    }

}

class CookieFileExtract
{
```

```
protected $cookie_file;
protected $cookie_list;

protected function  __construct($cookie_file)
{
    $this->cookie_file = $cookie_file;

    $this->cookie_list = $this->extractFile();
}

protected function isValidateCookieFile()
{
    if ($this->cookie_file && file_exists($this->cookie_file)) {
        return true;
    } else {
        return false;
    }
}

protected function extractFile()
{
    $cookie_list = array();

    if ($this->isValidateCookieFile($this->cookie_file)) {
        $content = file($this->cookie_file);
        if (is_array($content)) {
            foreach ($content as $line) {
                $line = trim($line);
                if (strlen($line) > 0 && $line[0] != '#') {
                    $cookie = (preg_split("/\s+/", $line));
                    if (count($cookie) == 7) {
                        $cookie_list[$cookie[5]] = $cookie[6];
                    } else {
                        $cookie_list[$cookie[5]] = '';
                    }
                }
            }
        }
    }

    return $cookie_list;
}

protected function buildCookieStr($cookies)
{
    $arr = array();

    if (is_array($cookies)) {
        foreach ($cookies as $k => $cookie) {
            $line = $cookie['domain'] . "\t" . "TRUE" . "\t" . $cookie['path'] . "\t" . "FALSE" . "\t" . $cookie['expires'] . "\t" . $k . "\t" . $cookie['value'];
            $arr[] = $line;
        }
    }
    return $arr;
}

protected function __setCookies($cookies)
{
    $new_line = array();
    if (is_array($cookies)) {
        if ($this->isValidateCookieFile($this->cookie_file)) {
            $content = file($this->cookie_file);
            if (is_array($content)) {
                foreach ($content as $line) {
                    $line = trim($line);
                    if (strlen($line) > 0 && $line[0] != '#') {
                        $cookie = (preg_split("/\s+/", $line));
                        if (!in_array($cookie[5], $cookies)) {
                            $new_line[] = $line;
                        }
                    } else {
                        $new_line[] = $line;
                    }
                }
            }
        }

        file_put_contents($this->cookie_file, implode("\n", array_merge($new_line, $this->buildCookieStr($cookies))));
    }
}

protected function __getAllCookies($refresh = false)
{
    if ($refresh) {
        $this->cookie_list = $this->extractFile();
    }
    return $this->cookie_list;
}

protected function __getCookie($cookie_name, $refresh = false)
{
    $cookie_list = $this->__getAllCookies($refresh);

    if (is_array($cookie_list) && array_key_exists($cookie_name, $cookie_list)) {
        return $cookie_list[$cookie_name];
    } else {
        return null;
    }
}
```

```
    protected function __clearAllCookies()
    {
        $this->cookie_list = null;
        @unlink($this->cookie_file);
    }
}

class BaseRequest extends CookieFileExtract
{

    protected $curl_instance;
    protected $request_timeout;
    protected $debug;
    protected $end_line;
    protected $user_agent = 'Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:26.0) Gecko/20100101 Firefox/26.0';

    protected function __construct($cookie_file, $request_timeout, $debug, $end_line)
    {
        parent::__construct($cookie_file);
        $this->request_timeout = $request_timeout;
        $this->debug = $debug;
        $this->end_line = $end_line;
        $this->initInstance();
    }

    protected function initInstance()
    {

        $this->curl_instance = curl_init();

        if ($this->isValidateCookieFile()) {
            curl_setopt($this->curl_instance, CURLOPT_COOKIEJAR, $this->cookie_file);
            curl_setopt($this->curl_instance, CURLOPT_COOKIEFILE, $this->cookie_file);
        }

        curl_setopt($this->curl_instance, CURLOPT_TIMEOUT, $this->request_timeout);
        curl_setopt($this->curl_instance, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($this->curl_instance, CURLOPT_HEADER, 1);
        curl_setopt($this->curl_instance, CURLOPT_SSL_VERIFYPEER, false);
        curl_setopt($this->curl_instance, CURLOPT_SSL_VERIFYHOST, 0);
        curl_exec($this->curl_instance);

    }

    function setCookies($cookies)
    {
        $this->closeInstance();
        $this->__setCookies($cookies);
        $this->initInstance();
    }

    protected function getAllCookies($refresh = false)
    {
        $this->closeInstance();
        $cookies = $this->__getAllCookies($refresh);
        $this->initInstance();
        return $cookies;
    }
    protected function clearAllCookies($refresh = false)
    {
        $this->closeInstance();
        $this->__clearAllCookies();

        if ($refresh) {
            $this->initInstance();
        }
    }

    protected function getCookie($name, $refresh = false)
    {
        $this->closeInstance();
        $cookie = $this->__getCookie($name, $refresh);
        $this->initInstance();
        return $cookie;
    }

    protected function getRequestInstance()
    {
        return $this->curl_instance;
    }

    protected function closeInstance()
    {
        if (is_resource($this->curl_instance)) {
            curl_close($this->curl_instance);
        }
    }

    protected function resetInstance()
    {
        $this->closeInstance();
        @unlink($this->cookie_file);
        $this->initInstance();
    }

    protected function requestExec($option)
    {

        curl_setopt_array($this->getRequestInstance(), $option);
```

```
        //if ($this->debug) {
        //    $result = curl_exec($this->getRequestInstance());
        //    Trace::write($result, $this->end_line, 'request output');
        //} else {
        return curl_exec($this->getRequestInstance());
        //}
    }
}

class QQVisitorRequest extends BaseRequest
{
    protected $user;
    protected $password;

    protected function __construct($user, $password, $cookie_file, $request_timeout, $debug, $end_line)
    {

        parent::__construct(dirname($cookie_file) . '/' . $user . '.' . basename($cookie_file), $request_timeout, $debug, $end_line);
        $this->user = $user;
        $this->password = $password;
    }
}
class ResultExtract
{

    public static function  getCoreJsInfo($content, $user)
    {
        $arr = array();
        preg_match('/cfg\s*=\s*\{(.*?)\s*\}\s*,\s*URL_PARAM_HASH/s', $content, $m);
        if (count($m) > 0) {
            $f = preg_replace('/\s*/', '', $m[1]);
            $f = preg_replace('/"\+g\_iLoginUin\+"/', $user, $f);
            $f = preg_replace('/\$j\.cookie.get\("hotfeeds_closed"\)==1\?""\:/', '', $f);

            $f = explode(',', $f);
            if (count($f) > 0) {
                foreach ($f as $t) {
                    $t = trim($t);
                    $p = strpos($t, ':');
                    $key = trim(substr($t, 0, $p), '"');
                    $value = trim(substr($t, $p + 1), '"');
                    if ($key) {
                        $arr[$key] = $value;
                    }
                }

                if (count($arr) > 0) {
                    $arr['visitor'] = $arr;
                }
            }
        }

        return $arr;
    }

    public static function  enterQzoneSuccess($content)
    {
        $arr = array();
        $arr2 = array();
        preg_match('/g_Data\s*=\s*{\s*feedsPart1\s*:\s*(.*?)\s*,\s*feedsPart2/s', $content, $m);

        if (count($m) > 0) {
            $f = preg_replace('/\s*/', '', $m[1]);
            $f = preg_replace('/([\{,])([^,]*?)(\:)/', '$1"$2"$3', $f);
            $f = preg_replace('/:\'(.*?)\'([,\}])/', ':"$1"$2', $f);
            $arr = json_decode($f, true);
            $arr = $arr['main'];
        }

        preg_match('/g_type.*?g_IZone_Flag/s', $content, $m);

        if (count($m) > 0) {
            $f = preg_replace('/\s*/', '', $m[0]);
            $f = explode(',', $f);

            foreach ($f as $t) {
                $t = trim($t);
                $p = strpos($t, '=');
                $key = trim(substr($t, 0, $p));
                $value = trim(substr($t, $p + 1), '"');
                if ($key) {
                    $arr2[$key] = $value;
                }
            }
        }

        return array_merge($arr, $arr2);

    }

    public static function getLoginJsInfo($content)
    {

        $s = preg_replace('/.*?pt\.plogin\s*=\s*\{(.*?)aqScanLink.*/s', '$1', $content);
        preg_match('/.*js_type\s*:\s*(\d+)\s*,.*/', $s, $m);

        if (count($m) > 1) {
            return array('js_type' => $m[1]);
        }
```

```
    }

    return array();
}

public static function getLoginAddress($content)
{
    preg_match('/.*?<\s*iframe\s*id\s*=\s*"login_frame"\s*name\s*=\s*"login_frame".*?src\s*=\s*"(.*?xui\.ptlogin2\.qq\.com.*?)".*?>\s*<\/iframe>.*?/', $content, $m);

    if (count($m) > 1) {
        return html_entity_decode($m[1]);
    }
    return null;
}

public static function checkLoginSuccess($content)
{

    preg_match_all('/.*?\((.*)\).*?/', $content, $match);
    if ($match[1][0]) {
        $g = explode(',', $match[1][0]);
        if (count($g) > 1) {
            if (($g[count($g) - 2]) == "'登录成功！'") {
                $url_parts = parse_url($g[2]);
                parse_str($url_parts['query'], $arr);
                if (array_key_exists('ptsig', $arr)) {
                    $ptsig = $arr['ptsig'];
                } else {
                    $ptsig = null;
                }
                return array('status' => true, 'value' => array('url' => $g[2], 'ptsig' => $ptsig));
            }
        }
    }
    return array('status' => false);
}
public static function rightFrameVisitors($content)
{
    $visitor_list = array();
    $f = preg_replace('/\s*/', '', $content);
    $f = preg_replace('/.*?_Callback\((\{.*?\})\).*?/', '$1', $f);
    $f = json_decode($f, true);

    if (is_array($f) && count($f) > 0 && array_key_exists('data', $f)
        && array_key_exists('module_3', $f['data'])
        && array_key_exists('data', $f['data']['module_3'])
        && array_key_exists('items', $f['data']['module_3']['data'])
    ) {

        $visitors = $f['data']['module_3']['data']['items'];

        foreach ($visitors as $visitor) {

            if (!array_key_exists('loc', $visitor)) {
                $visitor_list [] = array(
                    'uin' => $visitor['uin'], 'name' => $visitor['name'], 'online' => $visitor['online'], 'time' => $visitor['time'],
                    'img' => $visitor['img'], 'yellow' => $visitor['yellow'], 'supervip' => $visitor['supervip'],
                );
            }
        }
    }

    return $visitor_list;
}

public static function getVisitors($content)
{

    $f = preg_replace('/\s*/', '', $content);
    preg_match('/^.*?(\{.*?\})\);\s*$/', $f, $m);

    $visitor_list = array();

    if (is_array($m) && count($m) > 1 && strlen($m[1]) > 0) {
        $visitors = json_decode(trim($m[1]), true);

        if (array_key_exists('data', $visitors) && array_key_exists('items', $visitors['data'])) {

            foreach ($visitors['data']['items'] as $visitor) {

                if ($visitor['name']) {
                    $_ = array(
                        'uin' => $visitor['uin'], 'name' => $visitor['name'], 'time' => $visitor['time'],
                        'yellow' => $visitor['yellow'], 'supervip' => $visitor['supervip'],
                    );
                    if (array_key_exists('portraitlabel', $visitor)) {
                        $_['portraitlabel'] = $visitor['portraitlabel'];
                    }
                    $visitor_list[] = $_;

                    if (array_key_exists('uins', $visitor)) {
                        foreach ($visitor['uins'] as $uins) {
                            $_ = array(
                                'uin' => $uins['uin'], 'name' => $uins['name'], 'time' => $uins['time'],
                                'yellow' => $uins['yellow'], 'supervip' => $uins['supervip'],
                            );

                            if (array_key_exists('portraitlabel', $uins)) {
                                $_['portraitlabel'] = $uins['portraitlabel'];
                            }
```

```
                            $visitor_list[] = $_;
                        }
                    }
                }
            }
        }
        return $visitor_list;
    }

    public static function  checkVC($content)
    {
        preg_match_all('/.*?\((.*)\).*?/', $content, $match);

        if (strlen($match[1][0]) > 1) {
            $m = str_replace("'", '', $match[1][0]);
            $g = explode(',', $m);

            if (count($g) == 3) {
                return array('saltUin' => $g[2], 'verifycode' => $g[1], 'RSAKey' => $g[2] ? false : true);
            }
        }
        return array();
    }

    public static function getLoginInfo($content)
    {
        $s = preg_replace('/.*?pt\.ptui\s*=\s*\{(.*?)\}\s*;.*/s', '$1', $content);
        $e = preg_split('/,\s*/', trim($s));

        $info = array();

        foreach ($e as $t) {

            $t = trim($t);
            $p = strpos($t, ':');
            $key = trim(substr($t, 0, $p));
            $value = trim(substr($t, $p + 1));

            if (preg_match('/encodeURIComponent/', $value)) {
                $value = preg_replace('/^encodeURIComponent\s*\(\s*"(.*)?"\s*\)\s*,?$/', '$1', $value);
            } else {
                $value = trim($value, "',");
            }

            if ($key) {
                $info[$key] = urldecode($value);
            }
        }
        return $info;
    }

}
class QQVisitorCapture extends QQVisitorRequest
{

    public function __construct($user, $password, $cookie_file, $request_timeout, $debug, $end_line)
    {
        parent:: __construct($user, $password, $cookie_file, $request_timeout, $debug, $end_line);
    }

    public function trace($content, $title)
    {
        if ($this->debug = true) {
            Trace:: write($content, $this->end_line, $title);
        }
    }

    public function error($message)
    {
        $this->trace($message, 'login error ');
        return false;
    }

    public function success()
    {
        return true;
    }

    public function getGTKEncryption()
    {
        return Utils ::getGTK($this->getCookie('skey', true));
    }

    public function getCookies($refresh = false)
    {
        return $this->getAllCookies($refresh);
    }

    public function clearCookies($refresh = false)
    {
        return $this->clearAllCookies($refresh);
    }

    public function login()
    {
        $login_submit_info_url = null;
        $login_submit_info_url = $this->visitQzone();

        $this->setCookie(array(
```

```
$this->setCookies(array(
    'pgv_pvid' => array(
        'value' => Utils::getUTCMilliseconds(), 'path' => '/', 'domain' => '.qq.com', 'expires' => '0'
    ),
    'pgv_info' => array(
        'value' => 'ssid=s' . Utils::getUTCMilliseconds(), 'path' => '/', 'domain' => '.qq.com', 'expires' => '0'
    ),
    '_qz_referrer' => array(
        'value' => 'qzone.qq.com', 'path' => '/', 'domain' => '.qq.com', 'expires' => '0'
    ),
));

//log
$this->trace('', 'login begin');

//log
$this->trace($login_submit_info_url, '$login_submit_info_url===');

$login_submit_info = $this->getLoginSubmitInfo($login_submit_info_url);

//log
$this->trace($login_submit_info, '$login_submit_info===');

if (empty($login_submit_info)) {
    $this->error('err-001');
}

$this->report();

//log
$this->trace('', 'getLoginJs');
$js_arr = $this->getLoginJs();

//log
$this->trace($js_arr, '$getLoginJs===');

if (empty($js_arr)) {
    $this->error('err-002');
}

$u = $uin = $this->user;
$r = Utils::jsRandom();
$verifycode = null;
$pt_rsa = null;
$ptredirect = $login_submit_info['target'];
$h = $t = $g = $from_ui = 1;
$p = null;
$regmaster = $login_submit_info['regmaster'];
$u1 = Utils::decodeURIComponent($login_submit_info['s_url']);
$ptlang = $login_submit_info['lang'];
$action = strval(rand(5, 9)) . '-' . strval(strlen($this->user) + strlen($this->password) + rand(1, 5)) . '-' . Utils::loginJsTime();
$js_ver = $login_submit_info['ptui_version'];
$js_type = $js_arr['js_type'];
$login_sig = $login_submit_info['login_sig'];
$appid = $aid = $login_submit_info['appid'];
$pt_qzone_sig = $login_submit_info['pt_qzone_sig'];
$daid = $login_submit_info['daid'];

//log
$this->trace('', 'checkVC');
$check_data = $this->checkVC($regmaster, $appid, $js_ver, $js_type, $login_sig, $u1, $r);

if (count($check_data) !== 3) {
    $this->error('err-003');
}

//log
$this->trace($check_data, '$check_data===');

//log
$this->trace(json_encode($check_data), '$check_data===');

$verifycode = $check_data['verifycode'];

if ($check_data['RSAKey']) {
    $this->error('err-004');
} else {
    $p = Utils::getEncryption($this->password, $check_data['saltUin'], $verifycode);
    $pt_rsa = 0;
}

//log
$this->trace('', 'submitLogin');

$this->setCookies(array(
    'ptui_loginuin' => array(
        'value' => $this->user, 'path' => '/', 'domain' => '.qq.com', 'expires' => '0'
    ),
));

$login_result = $this->submitLogin($verifycode, $p,
    $pt_rsa, $ptredirect, $u1,
    $h, $t, $g, $from_ui,
    $ptlang, $action, $js_ver, $js_type,
    $login_sig, $aid, $daid, $pt_qzone_sig);

if ($login_result['status']) {
    $this->trace('登录成功', 'submitLogin');
    $this->trace($login_result, '$login_result====');
    $this->loginSuccessRedirect($login_result['value']['url']);
```

```
            $this->setCookies(array(
                'fnc' => array(
                    'value' => 1, 'path' => '/', 'domain' => '.qzone.qq.com', 'expires' => '0'
                ),
            ));
            $enterQzoneInfo = $this->enterQzone($login_result['value']['url'], $login_result['value']['ptsig']);

            //log
            $this->trace($enterQzoneInfo, '$enterQzoneInfo===');

            if ($enterQzoneInfo) {

                $this->trace('进入空间', 'enterQzone');

                //$referer = $login_result['value']['url'];
                //$scope = 0;

                //log
                $this->trace('', 'getCoreJs');
                $coreJsInfo = $this->getCoreJs();
                $this->trace($coreJsInfo, 'getCoreJs===');

                $this->setCookies(array(
                    'qzone_referer' => array(
                        'value' => $login_result['value']['url'], 'path' => '/', 'domain' => '.local.cckf123456789.com', 'expires' => '0'
                    ),
                    'qzone_visitor_param' => array(
                        'value' => implode('|', array_values($coreJsInfo['visitor'])), 'path' => '/', 'domain' => '.local.cckf123456789.com', 'expires' => '0'
                    ),
                ));

                //log
                //$this->trace('', 'rightFrameVisitor');
                //print_r($this->rightFrameVisitor($referer, implode('|', array_values($coreJsInfo['visitor']))));

                return $this->success();
            } else {
                $this->error('err-006');
            }

        } else {
            $this->error('err-005');
        }
    }

    protected function visitQzone()
    {
        $options = array(
            CURLOPT_TIMEOUT => $this->request_timeout,
            CURLOPT_HEADER => 1,
            CURLOPT_RETURNTRANSFER => 1,
            CURLOPT_URL => 'http://qzone.qq.com',
            CURLOPT_HTTPHEADER => array(
                'Referer:http://xui.ptlogin2.qq.com/cgi-bin/xlogin?proxy_url=http%3A//qzs.qq.com/qzone/v6/portal/proxy.html&daid=5&pt_qzone_sig=1&hide_title_bar=1&low_login=0&qlogin_auto_login=1
&no_verifyimg=1&link_target=blank&appid=549000912&style=22&target=self&s_url=http%3A//qzs.qq.com/qzone/v5/loginsucc.html?para=izone&pt_qr_app=%E6%89%8B%E6%9C%BAQQ%E7%A
9%BA%E9%97%B4&pt_qr_link=http%3A//z.qzone.com/download.html&self_regurl=http%3A//qzs.qq.com/qzone/v6/reg/index.html&pt_qr_help_link=http%3A//z.qzone.com/download.html',
                'User-Agent:' . $this->user_agent,
                'Host:qzone.qq.com',
                'Accept-Language:zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3',
                'Connection:keep-alive',)
        );

        return ResultExtract::getLoginAddress($this->requestExec($options));
    }

    protected function getLoginJs()
    {
        $options = array(
            CURLOPT_TIMEOUT => $this->request_timeout,
            CURLOPT_HEADER => 1,
            CURLOPT_RETURNTRANSFER => 1,
            CURLOPT_URL => 'http://imgcache.qq.com/ptlogin/ver/10067/js/c_login_old.js',
            CURLOPT_HTTPHEADER => array(
                'Referer:http://xui.ptlogin2.qq.com/cgi-bin/xlogin?proxy_url=http%3A//qzs.qq.com/qzone/v6/portal/proxy.html&daid=5&pt_qzone_sig=1&hide_title_bar=1&low_login=0&qlogin_auto_login=1
&no_verifyimg=1&link_target=blank&appid=549000912&style=22&target=self&s_url=http%3A//qzs.qq.com/qzone/v5/loginsucc.html?para=izone&pt_qr_app=%E6%89%8B%E6%9C%BAQQ%E7%A
9%BA%E9%97%B4&pt_qr_link=http%3A//z.qzone.com/download.html&self_regurl=http%3A//qzs.qq.com/qzone/v6/reg/index.html&pt_qr_help_link=http%3A//z.qzone.com/download.html',
                'User-Agent:' . $this->user_agent,
                'Host:imgcache.qq.com',
                'Connection:keep-alive',)
        );

        return ResultExtract::getLoginJsInfo($this->requestExec($options));
    }

    public function checkVC($regmaster, $appid, $js_ver, $js_type, $login_sig, $u1, $r)
    {

        $options = array(
            CURLOPT_TIMEOUT => $this->request_timeout,
            CURLOPT_HEADER => 1,
            CURLOPT_RETURNTRANSFER => 1,
            CURLOPT_URL => 'http://check.ptlogin2.qq.com/check?' . http_build_query(array(
                'regmaster' => $regmaster,
                'uin' => $this->user,
                'appid' => $appid,
                'js_ver' => $js_ver,
                'js_type' => $js_type,
                'login_sig' => $login_sig,
                'u1' => $u1,
```

```
                'r' => $r,
            )),
            CURLOPT_HTTPHEADER => array(
                'Referer:http://xui.ptlogin2.qq.com/cgi-bin/xlogin?proxy_url=http%3A//qzs.qq.com/qzone/v6/portal/proxy.html&daid=5&pt_qzone_sig=1&hide_title_bar=1&low_login=0&qlogin_auto_login=1&no_verifyimg=1&link_target=blank&appid=549000912&style=22&target=self&s_url=http%3A//qzs.qq.com/qzone/v5/loginsucc.html?para=izone&pt_qr_app=%E6%89%8B%E6%9C%BAQQ%E7%A9%BA%E9%97%B4&pt_qr_link=http%3A//z.qzone.com/download.html&self_regurl=http%3A//qzs.qq.com/qzone/v6/reg/index.html&pt_qr_help_link=http%3A//z.qzone.com/download.html',
                'User-Agent:' . $this->user_agent,
                'Host:check.ptlogin2.qq.com',
                'Accept-Language:zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3',
                'Connection:keep-alive',
            )
        );

        return ResultExtract::checkVC($this->requestExec($options));
    }

    protected function report()
    {
        $url = 'http://ui.ptlogin2.qq.com/cgi-bin/report?id=358342&t=' . Utils::jsRandom();

        $options = array(
            CURLOPT_TIMEOUT => $this->request_timeout,
            CURLOPT_HEADER => 1,
            CURLOPT_RETURNTRANSFER => 1,
            CURLOPT_URL => $url,
            CURLOPT_HTTPHEADER => array(
                'Referer:http://xui.ptlogin2.qq.com/cgi-bin/xlogin?proxy_url=http%3A//qzs.qq.com/qzone/v6/portal/proxy.html&daid=5&pt_qzone_sig=1&hide_title_bar=1&low_login=0&qlogin_auto_login=1&no_verifyimg=1&link_target=blank&appid=549000912&style=22&target=self&s_url=http%3A//qzs.qq.com/qzone/v5/loginsucc.html?para=izone&pt_qr_app=%E6%89%8B%E6%9C%BAQQ%E7%A9%BA%E9%97%B4&pt_qr_link=http%3A//z.qzone.com/download.html&self_regurl=http%3A//qzs.qq.com/qzone/v6/reg/index.html&pt_qr_help_link=http%3A//z.qzone.com/download.html',
                'User-Agent:' . $this->user_agent,
                'Host:ui.ptlogin2.qq.com',
                'Connection:keep-alive',)
        );

        $this->requestExec($options);

    }

    protected function getLoginSubmitInfo($url)
    {
        $options = array(
            CURLOPT_TIMEOUT => $this->request_timeout,
            CURLOPT_HEADER => 1,
            CURLOPT_RETURNTRANSFER => 1,
            CURLOPT_URL => $url,
            CURLOPT_HTTPHEADER => array(
                'Referer:http://qzone.qq.com/',
                'User-Agent:' . $this->user_agent,
                'Host:xui.ptlogin2.qq.com',
                'Connection:keep-alive',
            )
        );

        return ResultExtract::getLoginInfo($this->requestExec($options));
    }

    protected function submitLogin($verifycode, $p,
                                   $pt_rsa, $ptredirect, $u1,
                                   $h, $t, $g, $from_ui,
                                   $ptlang, $action, $js_ver, $js_type,
                                   $login_sig, $aid, $daid, $pt_qzone_sig)
    {
        $url = 'http://ptlogin2.qq.com/login';

        $url .= '?' . http_build_query(array(
                'u' => $this->user,
                'verifycode' => $verifycode));

        $url .= '&p=' . $p; ###

        $url .= '&' . http_build_query(array(
                'pt_rsa' => $pt_rsa,
                'ptredirect' => $ptredirect,
                'u1' => $u1,
                'h' => $h,
                't' => $t,
                'g' => $g,
                'from_ui' => $from_ui,
                'ptlang' => $ptlang,
                'action' => $action,
                'js_ver' => $js_ver,
                'js_type' => $js_type,
            ));

        $url .= '&login_sig=' . $login_sig; ###

        $url .= '&' . http_build_query(array(
                'aid' => $aid,
                'daid' => $daid,
                'pt_qzone_sig' => $pt_qzone_sig));

        $options = array(
            CURLOPT_TIMEOUT => $this->request_timeout,
            CURLOPT_HEADER => 1,
            CURLOPT_RETURNTRANSFER => 1,
            CURLOPT_URL => $url,
            CURLOPT_HTTPHEADER => array(
                'Referer:http://xui.ptlogin2.qq.com/cgi-bin/xlogin?proxy_url=http%3A//qzs.qq.com/qzone/v6/portal/proxy.html&daid=5&pt_qzone_sig=1&hide_title_bar=1&low_login=0&qlogin_auto_login=1&no_verifyimg=1&link_target=blank&appid=549000912&style=22&target=self&s_url=http%3A//qzs.qq.com/qzone/v5/loginsucc.html?para=izone&pt_qr_app=%E6%89%8B%E6%9C%BAQQ%E7%A9%BA%E9%97%B4&pt_qr_link=http%3A//z.qzone.com/download.html&self_regurl=http%3A//qzs.qq.com/qzone/v6/reg/index.html&pt_qr_help_link=http%3A//z.qzone.com/download.html',
```

```
            'User-Agent:' . $this->user_agent,
            'Host:ptlogin2.qq.com',
            'Accept-Language:zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3',
            'Connection:keep-alive',
        )
    );

    return ResultExtract::checkLoginSuccess($this->requestExec($options));
}

public function loginSuccessRedirect($url)
{
    $options = array(
        CURLOPT_TIMEOUT => $this->request_timeout,
        CURLOPT_HEADER => 1,
        CURLOPT_RETURNTRANSFER => 1,
        CURLOPT_URL => $url,

        CURLOPT_HTTPHEADER => array(
            'Referer:http://xui.ptlogin2.qq.com/cgi-bin/xlogin?proxy_url=http%3A//qzs.qq.com/qzone/v6/portal/proxy.html&daid=5&pt_qzone_sig=1&hide_title_bar=1&low_login=0&qlogin_auto_login=1
&no_verifyimg=1&link_target=blank&appid=549000912&style=22&target=self&s_url=http%3A//qzs.qq.com/qzone/v5/loginsucc.html?para=izone&pt_qr_app=%E6%89%8B%E6%9C%BAQQ%E7%A
9%BA%E9%97%B4&pt_qr_link=http%3A//z.qzone.com/download.html&self_regurl=http%3A//qzs.qq.com/qzone/v6/reg/index.html&pt_qr_help_link=http%3A//z.qzone.com/download.html',
            'User-Agent:' . $this->user_agent,
            'Host:qzs.qq.com',
            'Accept:text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
            'Connection:keep-alive',
            'DNT:1',
        )
    );
    $this->requestExec($options);
}

public function  enterQzone($referer, $ptsig)
{
    $options = array(
        CURLOPT_TIMEOUT => $this->request_timeout,
        CURLOPT_HEADER => 1,
        CURLOPT_RETURNTRANSFER => 1,
        CURLOPT_URL => 'http://user.qzone.qq.com/' . $this->user . '?ptsig=' . $ptsig,

        CURLOPT_HTTPHEADER => array(
            'Referer:' . $referer,
            'Host:user.qzone.qq.com',
            'Connection:keep-alive',
        )
    );

    return ResultExtract::enterQzoneSuccess($this->requestExec($options));
}

public function getCoreJs()
{
    $options = array(
        CURLOPT_TIMEOUT => $this->request_timeout,
        CURLOPT_HEADER => 1,
        CURLOPT_RETURNTRANSFER => 1,
        CURLOPT_URL => "http://ctc.qzonestyle.gtimg.cn/c/=/qzone/v8/engine/cpu.js,/qzone/v8/ic/qm.js,/qzone/v8/ic/tab_menu.js,/qzone/v8/ic/feeds.js,/qzone/v8/ic/tab_friend_feed.js,/qzone/v8/t
oolbar/core.js",
    );

    return ResultExtract::getCoreJsInfo($this->requestExec($options), $this->user);
}
public function getVisitorInfo($mask = 7, $page = 1, $fupdate = 1, $clear = 1)
{
    $options = array(
        CURLOPT_TIMEOUT => $this->request_timeout,
        CURLOPT_HEADER => 1,
        CURLOPT_RETURNTRANSFER => 1,
        CURLOPT_URL => 'http://g.qzone.qq.com/cgi-bin/friendshow/cgi_get_visitor_more?' . http_build_query(array(
            'uin' => $this->user,
            'mask' => $mask,
            'g_tk' => $this->getGTKEncryption(),
            'page' => $page,
            'fupdate' => $fupdate,
            'clear' => $clear,
            'sd' => Utils::jsRandom(),
        )),

        CURLOPT_HTTPHEADER => array(
            'Referer:http://ctc.qzs.qq.com/qzone/v6/friend_manage/visitors.html',
            'User-Agent:' . $this->user_agent,
            'Host:g.qzone.qq.com',
            'Connection:keep-alive',
        )
    );

    return ResultExtract::getVisitors($this->requestExec($options));
}

public function  rightFrameVisitor()
{
    $param = Utils ::getGTK($this->getCookie('qzone_visitor_param', true));
    $referver = Utils ::getGTK($this->getCookie('qzone_referer'));

    $options = array(
        CURLOPT_TIMEOUT => $this->request_timeout,
        CURLOPT_HEADER => 1,
        CURLOPT_RETURNTRANSFER => 1,
        CURLOPT_URL => 'http://r.qzone.qq.com/cgi-bin/right_frame.cgi?' . http_build_query(array(
```

```
                'uin' => $this->user,
                'param' => $param,
                'g_tk' => $this->getGTKEncryption(),
            )),

            CURLOPT_HTTPHEADER => array(
                'Referer:' . $referver,
                'User-Agent:' . $this->user_agent,
                'Host:r.qzone.qq.com',
                'Connection:keep-alive',
            )
        );

        return ResultExtract::rightFrameVisitors($this->requestExec($options));
    }

}

class CCKFServiceRequest extends BaseRequest
{
    protected $service_address;
    protected $service_id;
    protected $security_key;

    public function __construct($security_key, $service_id, $service_address, $cookie_file, $request_timeout, $debug, $end_line)
    {
        parent:: __construct($cookie_file, $request_timeout, $debug, $end_line);
        $this->service_address = $service_address;
        $this->service_id = $service_id;
        $this->security_key = $security_key;
    }
}

class CCKFService extends BaseRequest
{
    public function __construct($security_key, $service_id, $service_address, $cookie_file, $request_timeout, $debug, $end_line)
    {
        parent:: __construct($security_key, $service_id, $service_address, $cookie_file, $request_timeout, $debug, $end_line);
    }

    public function uploadData($data)
    {
        if (is_array($data) && !empty($data)) {
            $options = array(
                CURLOPT_TIMEOUT => $this->request_timeout,
                CURLOPT_HEADER => 1,
                CURLOPT_RETURNTRANSFER => 1,
                CURLOPT_URL => $this->service_address . '?' . http_build_query(array()),
            );
        }
    }
}

class BaseConfigFileUtils
{
    protected $file;

    public function __construct($file)
    {
        $this->file = $file;
    }

    public function extractFile()
    {
        $f_str = '';

        $fp = fopen($this->file, 'r');
        if (flock($fp, LOCK_SH)) {
            while (!feof($fp)) {
                $f_str .= fgets($fp);
            }
            flock($fp, LOCK_UN);
        }
        fclose($fp);
        $c = json_decode($f_str, true);
        return is_array($c) ? $c : array();
    }
}

class RunAtTimeConfig extends BaseConfigFileUtils
{
    protected $visitor_capture_interval;
    protected $config;

    public function __construct($file, $visitor_capture_interval)
    {
        parent::__construct($file);
        $this->config = $this->extractFile();
        $this->visitor_capture_interval = $visitor_capture_interval;
    }

    public function  updateConfig($arr)
    {
        $this->config = $this->extractFile();
        if ($this->isValidateRun()) {
            $fp = fopen($this->file, 'w');
            if (flock($fp, LOCK_EX)) {
                fwrite($fp, json_encode(array_merge($this->config, $arr)));
                flock($fp, LOCK_UN);
            }
```

```
            }
            fclose($fp);
            return true;
        }
        return false;
    }

    public function  getConfig($item)
    {
        if (is_array($this->config) && array_key_exists($item, $this->config)) {
            return $this->config[$item];
        }
        return null;
    }

    public function isValidateRun()
    {
        $c_time = Utils::getMicroTime();
        $run_at_time = floatval($this->getConfig('run_at_time'));
        if ($c_time - $run_at_time >= $this->visitor_capture_interval) {
            return true;
        } else {
            return false;
        }
    }
}

class VisitorList extends BaseConfigFileUtils
{
    protected $visitor_list;

    /**$visitor_list [] = array(
    'uin' => $visitor['uin'], 'name' => $visitor['name'], 'online' => $visitor['online'], 'time' => $visitor['time'],
    'img' => $visitor['img'], 'yellow' => $visitor['yellow'], 'supervip' => $visitor['supervip'],
    );**/

    public function __construct($file)
    {
        parent::__construct($file);
        $this->visitor_list = $this->extractFile();
    }

    public function updateVisitor($visitor)
    {
        if (is_array($visitor) && !empty($visitor)) {
            foreach ($visitor as $one) {
                array_unshift($this->visitor_list, $one);
            }
        }

        $fp = fopen($this->file, 'w');
        if (flock($fp, LOCK_EX)) {
            fwrite($fp, json_encode($this->visitor_list));
            flock($fp, LOCK_UN);
        }
        fclose($fp);
    }

    public function addVisitor($visitor)
    {
        $list = array();
        if (is_array($visitor) && !empty($visitor)) {
            foreach ($visitor as $one) {
                if (!$this->isVisitorExist($one['name'])) {
                    $list[] = $one;
                }
            }
            $this->updateVisitor($list);
        }

        return $list;
    }

    public function isVisitorExist($name)
    {
        foreach ($this->visitor_list as $one) {
            if ($one['name'] == $name) {
                return true;
            }
        }
        return false;
    }
}
```

PHP中CURL的CURLOPT_POSTFIELDS参数使用细节

在通常情况下，我们使用 CURL 来提交 POST 数据的时候，我们已经习惯了这样的写法：

```
curl_setopt( $ch, CURLOPT_POSTFIELDS,$post_data);
```

但是这样的写法在有时候并不会很好用，可能会得到服务器返回的 500 错误。但是我们尝试在使用 Socket 方式向服务器提交数据的时候，我们会得到非常正确的结果。
例如我们在服务器上面存在一个如下的 PHP 文件：

```
<?php print_r($_SERVER);?>
```

当我们采用 CURL 在不注意细节的前提下向服务器发送一些数据，我们可能得到下面这样的结果，这不是我们理想中的结果：

```
[CONTENT_TYPE] => multipart/form-data; boundary=----------f924413ea122
```

但是如果我们在采用 http_build_query($post_data) 来替代 $post_data 再向这个 PHP 脚本提交数据的时候，我们就会得到和上面不同的结果，这才是我们理想中的结果：

```
[CONTENT_TYPE] => application/x-www-form-urlencoded
```

从上面这个例子中不难看出，使用 CURL 并且参数为数据时，向服务器提交数据的时候，HTTP头会发送Content_type: application/x-www-form-urlencoded。这个是正常的网页<form>提交表单时，浏览器发送的头部。而 multipart/form-data 我们知道这是用于上传文件的表单。包括了 boundary 分界符，会多出很多字节。
官方的手册上是这样说的：

```
The full data to post in a HTTP "POST" operation. To post a file, prepend a filename with @ and use the full path. This can either be passed as a urlencoded string like 'para1=val1¶2=val2&...' or as an array with the field name as key and field data as value. If value is an array, the Content-Type header will be set to multipart/form-data.
```

使用数组提供 post 数据时，CURL 组件大概是为了兼容 @filename 这种上传文件的写法，默认把 content_type 设为了 multipart/form-data。虽然对于大多数服务器并没有影响，但是还是有少部分服务器不兼容。
经过一番总结最终得出结论：在没有需要上传文件的情况下，尽量对 post 提交的数据进行 http_build_query 处理，然后再发送出去，能实现更好的兼容性，更小的请求数据包。

cURL multi批处理实现及避免cURL multi造成CPU负载过高问题

简单的cURL处理如下：

```
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, 'http://www.phpddt.com');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
$con = curl_exec($ch);
curl_close($ch);
```

cURL还提供了批量处理会话，下面是**cURL批量处理相关函数**：

curl_multi_init — 返回一个新cURL批处理句柄
curl_multi_add_handle — 向curl批处理会话中添加单独的curl句柄
curl_multi_exec — 解析一个cURL批处理句柄
curl_multi_getcontent — 如果设置了CURLOPT_RETURNTRANSFER，则返回获取的输出的文本流
curl_multi_select — 等待所有cURL批处理中的活动连接
curl_multi_info_read — 获取当前解析的cURL的相关传输信息
curl_multi_remove_handle — 移除curl批处理句柄资源中的某个句柄资源
curl_multi_close — 关闭一组cURL句柄

看下面使用**curl multi批处理的例子**：

```
<?php
/**
 * cURL multi批量处理
 *
 * @author mckee
 * @link http://www.phpddt.com  *
 */

$url_array = array(
    'http://www.phpddt.com/',
    'http://www.phpddt.com/php/627.html',
    '/article/1208556.html'
);

$handles = $contents = array();

//初始化curl multi对象
$mh = curl_multi_init();

//添加curl 批处理会话
foreach($url_array as $key => $url)
{
    $handles[$key] = curl_init($url);
    curl_setopt($handles[$key], CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($handles[$key], CURLOPT_TIMEOUT, 10);

    curl_multi_add_handle($mh, $handles[$key]);
}

//======================执行批处理句柄=================================
$active = null;
do {
    $mrc = curl_multi_exec($mh, $active);
} while ($mrc == CURLM_CALL_MULTI_PERFORM);


while ($active and $mrc == CURLM_OK) {

    if(curl_multi_select($mh) === -1){
        usleep(100);
    }
    do {
        $mrc = curl_multi_exec($mh, $active);
    } while ($mrc == CURLM_CALL_MULTI_PERFORM);

}
//====================================================================

//获取批处理内容
foreach($handles as $i => $ch)
{
    $content = curl_multi_getcontent($ch);
    $contents[$i] = curl_errno($ch) == 0 ? $content : '';
}

//移除批处理句柄
foreach($handles as $ch)
{
    curl_multi_remove_handle($mh, $ch);
}

//关闭批处理句柄
curl_multi_close($mh);

print_r($contents);
```

上面这段程序重点是执行批处理的那段，普通的处理：

```
do { $n=curl_multi_exec($mh,$active); } while ($active);
```

会造成CPU Loading过高，因为$active要等全部url数据接受完毕才变成false，所以这里用到了curl_multi_exec的返回值判断是否还有数据，当有数据的时候就不停调用curl_multi_exec，没有执行数据就会sleep，如此就会避免CPU Loading 100%了。

参考资料：

（1）PHP手册关于cURL教程

（2）curl预定义常量

php利用curl抓取新浪微博内容示例

很多人都喜欢在网站上DIY自己的微博，所以我也写了一个。
这里直接抓取了新浪微博工具中的微博秀地址。

```
<?php
  set_time_limit(0);
  $url="http://widget.weibo.com/weiboshow/index.php?language=&width=0&height=550&fansRow=2&ptype=1&speed=0&skin=1&isTitle=1&noborder=1&isWeibo=1&isFans=1&uid=1724077823&verifier=8738a0fa&dpc=1"; //微博秀地址
  $ch=curl_init();
  curl_setopt($ch,CURLOPT_HEADER,false);
  curl_setopt($ch,CURLOPT_URL,$url);
  curl_setopt($ch,CURLOPT_RETURNTRANSFER,1);
  $content=curl_exec($ch);
  curl_close($ch);
  preg_match_all('/<p class="weiboShow_mainFeed_listContent_txt">(.*)<\/p>/iUs',$content,$text);//获取文字
  preg_match_all('/<span class="weiboShow_mainFeed_listContent_actionTime">(.*)<\/span>/iUs',$content,$time);//获取时间
  $me=explode('<div class="weiboShow_developer_pic">',$content);
  $me=explode('</div>',$me[1]);
  preg_match_all("/src=\"([^\"].*)\"/iUs",$me[0],$avatar);//获取我的头像

  $a=$text[0];
  $b=$time[0];
  $result=array_combine($a, $b);//合并数组
  foreach($result as $text=>$time){
    echo "<img src='".$avatar[1][0]."' height='50'/>";
    echo strip_tags($text);
    echo strip_tags($time);
  }
?>
```

PHP函数分享之curl方式取得数据、模拟登陆、POST数据

废话不多说直接上代码

```
/********************* curl 系列 ***********************/
//直接通过curl方式取得数据(包含POST、HEADER等)
/*
 * $url: 如果非数组，则为http;如是数组，则为https
 * $header: 头文件
 * $post: post方式提交 array形式
 * $cookies: 0默认无cookie,1为设置,2为获取
 */
public function curl_allinfo($urls, $header = FALSE, $post = FALSE, $cookies = 0) {
    $url = is_array($urls) ? $urls['0'] : $urls;
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

    //带header方式提交
    if($header != FALSE){
        curl_setopt($ch, CURLOPT_HTTPHEADER, $header);
    }

    //post提交方式
    if($post != FALSE){
        curl_setopt($ch, CURLOPT_POST, 1);
        curl_setopt($ch, CURLOPT_POSTFIELDS, $post);
    }

    if($cookies == 1){
        curl_setopt($ch, CURLOPT_COOKIEJAR, "cookiefile");
    }else if($cookies == 2){
        curl_setopt($ch, CURLOPT_COOKIEFILE, "cookiefile");
    }

    if(is_array($urls)){
        curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
        curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, false);
    }

    $data = curl_exec($ch);
    curl_close($ch);
    return $data;
}
```

PHP扩展CURL的用法详解

实现的功能：

1、实现远程获取和采集内容

2、实现PHP 网页版的FTP上传下载

3、实现模拟登陆：去一个邮件系统，curl可以模拟cookies

4、实现接口对接（API），数据传输等：通过一个平台发送短信啊，抓取和传递所传输的信息。

5、实现模拟Cookie等：登陆的状态下才可以操作一些属性。

如何使用CURL功能：

默认情况加PHP是不支持CURL的，需要在php.ini中开启该功能

;extension=php_curl.dll前面的分号去掉

1 整个操作过程中第一步是用cur_init()函数进行初始化

```
$curl = curl_init('www.jb51.net')
```

2.用curl_setopt（）函数进行设置选项。

3.设置后，进行执行事务 curl_exec($curl);

4 最后关闭curl_close();

使用PHP CURL实现传输和获取功能（post传输方式）：获取远程网页数据

```
$user = "admin";
$pass = "admin";
$curlPost = "user=$user&pass=$pass";
$ch = curl_init(); //初始化一个CURL对象
curl_setopt($ch, CURLOPT_URL, "http://localhost/edu/login.php");
//设置你所需要抓取的URL
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 0);
//设置curl参数，要求结果是否输出到屏幕上，为true的时候是不返回到网页中
假设上面的0换成1的话，那么接下来的$data就需要echo一下。
curl_setopt($ch, CURLOPT_POST, 1);
//post提交
curl_setopt($ch, CURLOPT_POSTFIELDS, $curlPost);
$data = curl_exec($ch);
//运行curl,请求网页。
curl_close($ch);
```

[/code]

实现远程模拟登陆最基础部分。

curl也还是需要配置用户名和密码的，只不过是被浏览器隐藏了。

==========

curl模拟登陆

模拟登陆：就是不登陆到php100的论坛，也能查看到相应的信息。

分析登陆字段--->登陆后保留cookie状-->读取cookie并跳转到相关页-->抓取数

1、模拟登陆后创建一个文件保存cookie内容

2、通过读取生成的cookie内容模拟用户登陆状态

3、到相关页面获取所需内容

tempname创建一个临时文件

tempnam() 函数创建一个具有唯一文件名的临时文件。 若成功，则该函数返回新的临时文件名。若失败，则返回 false。

tempnam(dir,prefix)

参数 描述

dir 必需。规定创建临时文件的目录。

prefix 必需。规定文件名的开头。

相当于，fopen fwirte fclose

它可以返回一个布尔值。使用第三方来登陆你的QQ、msn是很危险的，因为它可以记录你的登录状态，抓取你的用户名和密码。

使用CURL模拟登陆到PHP100论坛

1、分析登陆所需input框字段名和所需字段数量

2、保存cookie 模拟登陆后获取会员金币数量

代码：

```
//初始化一个 cURL 对象
$curl = curl_init();
//设置你需要抓取的URL
curl_setopt($curl, CURLOPT_URL, " http://www.baidu.com ");
//设置cURL 参数，要求结果保存到字符串中还是输出到屏幕上。
curl_setopt($curl, CURLOPT_RETURNTRANSFER, 0);
//运行cURL，请求网页
$data = curl_exec($curl);
//关闭URL请求
curl_close($curl);
$user = "admin";
$pass = "admin100";
$curlPost = "user=$user&pass=$pass";
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, " http://localhost/curl/login.php ");
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 0);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $curlPost);
$data = curl_exec($ch);
curl_close($ch);
?>
if($_POST['user']=="admin"){
 echo "";
}else{
 echo "";
}
//print_r($_POST);
?>
```

php采用curl访问域名返回405 method not allowed提示的解决方法

```
/**
 * http测试
 * 注：PHP版本5.2以上才支持CURL_IPRESOLVE_V4
 * @param $url 网站域名
 * @param $type 网站访问协议
 * @param $ipresolve 解析方式
 */
public function web_http($url,$type,$ipresolve) {
    //设置Header头
    $header[] = "Accept: application/json";
     $header[] = "Accept-Encoding: gzip";
    $httptype = function_exists('curl_init');
    if (!$httptype) {
      $html = file_get_contents($url);
    } else {
      $ch = curl_init();
      curl_setopt($ch, CURLOPT_URL, $url);
      //输出头信息
      curl_setopt($ch, CURLOPT_HEADER, 1);
      //递归访问location跳转的链接，直到返回200OK
      curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
      //不对HTML中的BODY部分进行输出
      curl_setopt($ch, CURLOPT_NOBODY, 1);
      //将结果以文件流的方式返回，不是直接输出
      curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
      //以IPv4/IPv6的方式访问
      if($ipresolve=='ipv6') {
        curl_setopt($ch,CURLOPT_IPRESOLVE,CURL_IPRESOLVE_V6);
      }else{
        curl_setopt($ch,CURLOPT_IPRESOLVE,CURL_IPRESOLVE_V4);
      }
      //添加HTTP header头采用压缩和GET方式请求
      curl_setopt( $ch, CURLOPT_HTTPHEADER, $header );
      curl_setopt($ch,CURLOPT_ENCODING , "gzip");
      curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'GET');
      //清除DNS缓存
      curl_setopt($ch,CURLOPT_DNS_CACHE_TIMEOUT,0);
      //设置连接超时时间
      curl_setopt($ch,CURLOPT_CONNECTTIMEOUT,15);
      //设置访问超时
      curl_setopt($ch,CURLOPT_TIMEOUT,50);
      //设置User-agent
      curl_setopt($ch,CURLOPT_USERAGENT,'Mozilla/5.0 (Windows NT 6.1) AppleWebKit/536.11 (KHTML, like Gecko) Chrome/20.0.1132.47 Safari/536.11');
      if($type=="https") {
          //不对认证证书来源的检查
          curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
          //从证书中检查SSL加密算法是否存在
          curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, true);
      }
      //执行Curl操作
      $html = curl_exec($ch);
      //获取一个cURL连接资源句柄的信息（获取最后一次传输的相关信息）
      $info = curl_getinfo($ch);
      curl_close($ch);
    }
    return $info;
  }
```

以上为一个基本curl访问的方法，由于这里需要通过使用IPv6的方式，所以加了相应的选项，相信大家能看的明白，平时经常用到的选项上面都有出现，大家根据需要取舍。

状态码提示405/Method Not Allowed表示不支持请求的方法，这个错误并不常见。

导致这个错误是要是**由于curl默认是采用post方式进行提交访问的，post方式在此类域名下是没有权限的**，比如在测试www.amazon.cn的时候就出现了这类问题，而修改为get的方式，并且增加了header头后，即可正常访问，个人推测，或许是亚马逊那边基本上都是采用get的方式，才会被认为是人为的点击，对post做了相应屏蔽。

对此增加了如下代码：

```
//设置Header头
$header[] = "Accept: application/json";
$header[] = "Accept-Encoding: gzip";
//添加HTTP header头采用压缩和GET方式请求
curl_setopt( $ch, CURLOPT_HTTPHEADER, $header );
curl_setopt($ch,CURLOPT_ENCODING , "gzip");
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'GET');
```

命令行的形式为：

```
curl -v www.amazon.cn
```

PHP执行Curl时报错提示CURL ERROR: Recv failure: Connection reset by peer的解决方法

最近在使用curl中遇到CURL ERROR: Recv failure: Connection reset by peer的报错提示，现把解决方法与大家共享，希望对大家有所帮助。

我们经常用curl来访问web站点，web站点目前主要分为http和https两种协议，众所周知https类型的网站都是通过ssl协议+http协议的，是目前最安全的网站协议，访问此类网站的时候，会走ssl协议，验证访问者的证书，检测是否安全。

通过curl访问此类网站也是如此流程，但是curl中需要添加相应的参数，**绕过ssl证书的验证，才可以正常访问**，如**出现此错误的一般原因是没有加此参数**（如下所示）。

```
curl_setopt($c, CURLOPT_SSL_VERIFYPEER, false);
curl_setopt($c, CURLOPT_SSL_VERIFYHOST, false);
```

PHP中使用CURL模拟登录并获取数据实例

cURL 是一个功能强大的PHP库，使用PHP的cURL库可以简单和有效地抓取网页并采集内容，设置cookie完成模拟登录网页，curl提供了丰富的函数，开发者可以从PHP手册中获取更多关于cURL信息。本文以模拟登录开源中国(oschina)为例，和大家分享cURL的使用。

PHP的curl()在抓取网页的效率方面是比较高的，而且支持多线程，而file_get_contents()效率就要稍低些，当然，使用curl时需要开启下curl扩展。
代码实战

先来看登录部分的代码：

```
//模拟登录
function login_post($url, $cookie, $post) {
    $curl = curl_init();//初始化curl模块
    curl_setopt($curl, CURLOPT_URL, $url);//登录提交的地址
    curl_setopt($curl, CURLOPT_HEADER, 0);//是否显示头信息
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, 0);//是否自动显示返回的信息
    curl_setopt($curl, CURLOPT_COOKIEJAR, $cookie); //设置Cookie信息保存在指定的文件中
    curl_setopt($curl, CURLOPT_POST, 1);//post方式提交
    curl_setopt($curl, CURLOPT_POSTFIELDS, http_build_query($post));//要提交的信息
    curl_exec($curl);//执行cURL
    curl_close($curl);//关闭cURL资源，并且释放系统资源
}
```

函数login_post()首先初始化curl_init()，然后使用curl_setopt()设置相关选项信息，包括要提交的url地址，保存的cookie文件，post的数据（用户名和密码等信息），是否返回信息等等，然后curl_exec执行curl，最后curl_close()释放资源。注意PHP自带的http_build_query()可以将数组转换成相连接的字符串。

接下来如果登录成功后，我们要获取登录成功后的页面信息。

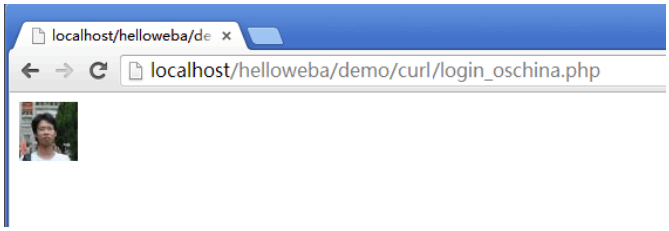
```
//登录成功后获取数据
function get_content($url, $cookie) {
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_HEADER, 0);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($ch, CURLOPT_COOKIEFILE, $cookie); //读取cookie
    $rs = curl_exec($ch); //执行cURL抓取页面内容
    curl_close($ch);
    return $rs;
}
```

函数get_content()中也是先初始化curl，然后设置相关选项，执行curl，释放资源。其中我们设置CURLOPT_RETURNTRANSFER为1即自动返回信息，而CURLOPT_COOKIEFILE可以读取到登录时保存的cookie信息，最后将页面内容返回。

我们的最终目的是要获取到模拟登录后的信息，也就是只有正常登录成功后才能获取的有用信息。接下来我们以登录开源中国的移动版为例，看看如何抓取到登录成功后的信息。

```
//设置post的数据
$post = array (
    'email' => 'oschina账户',
    'pwd' => 'oschina密码',
    'goto_page' => '/my',
    'error_page' => '/login',
    'save_login' => '1',
    'submit' => '现在登录'
);

//登录地址
$url = "http://m.jb51.net/action/user/login";
//设置cookie保存路径
$cookie = dirname(__FILE__) . '/cookie_jb51.txt';
//登录后要获取信息的地址
$url2 = "http://m.jb51.net/my";
//模拟登录
login_post($url, $cookie, $post);
//获取登录页的信息
$content = get_content($url2, $cookie);
//删除cookie文件
@ unlink($cookie);
//匹配页面信息
$preg = "/<td class='portrait'>(.*)<\/td>/i";
preg_match_all($preg, $content, $arr);
$str = $arr[1][0];
//输出内容
echo $str;
```

运行上述代码后，我们会看到最终获取到登录用户的头像图片。



使用总结：

1、初始化curl；

2、使用curl_setopt设置目标url，和其他选项；

3、curl_exec，执行curl；

4、执行后，关闭curl；

5、输出数据。

PHP curl实现抓取302跳转后页面的示例

PHP的CURL正常抓取页面程序如下：

```
$url = 'http://www.baidu.com';

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_VERBOSE, true);
curl_setopt($ch, CURLOPT_HEADER, true);
curl_setopt($ch, CURLOPT_NOBODY, true);
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'GET');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_TIMEOUT, 20);
curl_setopt($ch, CURLOPT_AUTOREFERER, true);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, true);
$ret = curl_exec($ch);
$info = curl_getinfo($ch);
curl_close($ch);
```

如果你抓取到的是302状态，是因为再抓取的过程中，有的跳转需要给下一个链接传递参数，而下一个链接同时也设置了如果没接收到相应的参数是为非法访问。

```
curl_setopt($curl, CURLOPT_CUSTOMREQUEST, 'GET');
```

显示就应该正常了。

上面用来抓取功能，几乎应该没问题的。你可以查一下CURLOPT_CUSTOMREQUEST相关资料。

使用一个自定义的请求信息来代替"GET"或"HEAD"作为HTTP请求。这对于执行"DELETE" 或者其他更隐蔽的HTTP请求。有效值如"GET"，"POST"，"CONNECT"等等。也就是说，不要在这里输入整个HTTP请求。例如输入"GET /index.html HTTP/1.0\r\n\r\n"是不正确的。

PHP使用CURL实现对带有验证码的网站进行模拟登录的方法

网上的很多模拟登录程序，大都是通过服务程序apache之类的运行，获取到验证码之后显示在网页上，然后填上再POST出去，这样虽然看起来很友好，但是既然模拟登录，登录后所干的事情就不一定是短时间完成的，所以这就要受到php最大执行时间的限制，而且有些操作还有可能权限不足。

本文提供了一个程序实例，思路就是获取到验证码之后把验证码存储为一个图片，然后程序休眠20秒，在20秒之后由用户手动查看图片，并把验证码填写到code.txt文件中，20秒休眠完成后，程序会读code.txt的验证码，这样再带着验证码进行登录操作。具体代码如下：

```
/**
 * 模拟登录
 */

//初始化变量
$cookie_file = "tmp.cookie";
$login_url = "http://xxx.com/logon.php";
$verify_code_url = "http://xxx.com/verifyCode.php";

echo "正在获取COOKIE...\n";
$curlj = curl_init();
$timeout = 5;
curl_setopt($curl, CURLOPT_URL, $login_url);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl, CURLOPT_CONNECTTIMEOUT, $timeout);
curl_setopt($curl,CURLOPT_COOKIEJAR,$cookie_file); //获取COOKIE并存储
$contents = curl_exec($curl);
curl_close($curl);

echo "COOKIE获取完成，正在取验证码...\n";
//取出验证码
$curl = curl_init();
curl_setopt($curl, CURLOPT_URL, $verify_code_url);
curl_setopt($curl, CURLOPT_COOKIEFILE, $cookie_file);
curl_setopt($curl, CURLOPT_HEADER, 0);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, 1);
$img = curl_exec($curl);
curl_close($curl);

$fp = fopen("verifyCode.jpg","w");
fwrite($fp,$img);
fclose($fp);
echo "验证码取出完成，正在休眠，20秒内请把验证码填入code.txt并保存\n";
//停止运行20秒
sleep(20);

echo "休眠完成，开始取验证码...\n";
$code = file_get_contents("code.txt");
echo "验证码成功取出：$code\n";
echo "正在准备模拟登录...\n";

$post = "username=maben&pwd=hahahaha&verifycode=$code";
$curl = curl_init();
curl_setopt($curl, CURLOPT_URL, $url);
curl_setopt($curl, CURLOPT_HEADER, false);
curl_setopt($curl, CURLOPT_RETURNTRANSFER,1);
curl_setopt($curl, CURLOPT_POSTFIELDS, $post);
curl_setopt($curl, CURLOPT_COOKIEFILE, $cookie_file);
$result=curl_exec($curl);
curl_close($curl);

//这一块根据自己抓包获取到的网站上的数据来做判断
if(substr_count($result,"登录成功")){
 echo "登录成功\n";
}else{
 echo "登录失败\n";
 exit;
}

//OK，开始做你想做的事吧。。。。。
```

php中的curl_multi系列函数使用例子

相信许多人对php手册中语焉不详的curl_multi一族的函数头疼不已，它们文档少，给的例子 更是简单的让你无从借鉴，我也曾经找了许多网页，都没见一个完整的应用例子。
curl_multi_add_handle
curl_multi_close
curl_multi_exec
curl_multi_getcontent
curl_multi_info_read
curl_multi_init
curl_multi_remove_handle
curl_multi_select

一般来说，想到要用这些函数时，目的显然应该是要同时请求多个url，而不是一个一个依次请求，否则不如自己循环去调curl_exec好了。

步骤总结如下：

第一步：调用curl_multi_init

第二步：循环调用curl_multi_add_handle

这一步需要注意的是，curl_multi_add_handle的第二个参数是由curl_init而来的子handle。

第三步：持续调用curl_multi_exec

第四步：根据需要循环调用curl_multi_getcontent获取结果

第五步：调用curl_multi_remove_handle，并为每个字handle调用curl_close

第六步：调用curl_multi_close
这里有一个网上找的简单例子，其作者称为dirty的例子，（稍后我会说明为何dirty）：

```
/*

Here's a quick and dirty example for curl-multi from PHP, tested on PHP 5.0.0RC1 CLI / FreeBSD 5.2.1

*/
$connomains = array(

"http://www.baidu.com/",

"http://www.google.com/",

"http://www.jb51.net/"

);

$mh = curl_multi_init();
foreach ($connomains as $i => $url) {

    $conn[$i]=curl_init($url);

     curl_setopt($conn[$i],CURLOPT_RETURNTRANSFER,1);

     curl_multi_add_handle ($mh,$conn[$i]);

}
do { $n=curl_multi_exec($mh,$active); } while ($active);

foreach ($connomains as $i => $url) {

     $res[$i]=curl_multi_getcontent($conn[$i]);

     curl_close($conn[$i]);

}
print_r($res);
```

整个使用过程差不多就是这样，但是，这个简单代码有个致命弱点，就是在do循环的那段，在整个url请求期间是个死循环，它会轻易导致CPU占用100%。

现在我们来改进它，这里要用到一个几乎没有任何文档的函数curl_multi_select了，虽然C的curl库对select有说明，但是，php里的接口和用法确与C中有不同。

把上面do的那段改成下面这样：

```
do {
        $mrc = curl_multi_exec($mh,$active);
    } while ($mrc == CURLM_CALL_MULTI_PERFORM);
    while ($active and $mrc == CURLM_OK) {
        if (curl_multi_select($mh) != -1) {
            do {
                $mrc = curl_multi_exec($mh, $active);
            } while ($mrc == CURLM_CALL_MULTI_PERFORM);
        }
    }
```

因为$active要等全部url数据接受完毕才变成false，所以这里用到了curl_multi_exec的返回值判断是否还有数据，当有数据的时候就不停调用curl_multi_exec，暂时没有数据就进入select阶段，新数据一来就可以被唤醒继续执行。这里的好处就是CPU的无谓消耗没有了。

另外：还有一些细节的地方可能有时候要遇到：

控制每一个请求的超时时间，在curl_multi_add_handle之前通过curl_setopt去做：

```
curl_setopt($ch, CURLOPT_TIMEOUT, $timeout);
```

判断是否超时了或者其他错误，在curl_multi_getcontent之前用：curl_error($conn[$i]);

PHP使用CURL_MULTI实现多线程采集的例子

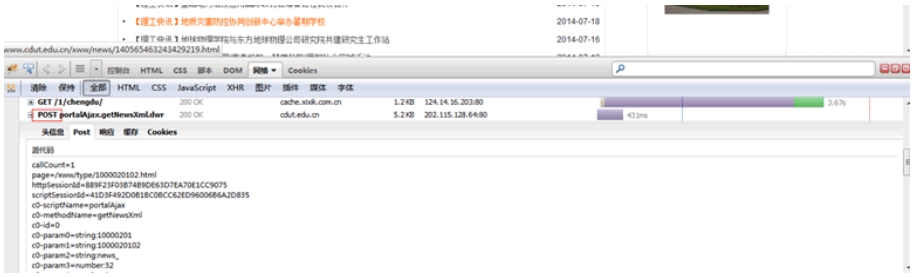
这两天有一客户定制了一个免登录发布模块，因为在模块中需要涉及到很多图片下载的问题，考虑到性能问题，所以特别写了一个CURL_MULTI远程采集网页的函数，以方便以后使用，估计以后都不会使用原来的单线程curl函数去foreach了，其性能对比很明显的。同样获取我的博客的十个不同网页，curl_multi:4.5246081352234，file_get_contents:33.001797914505，将近8倍的效率，可想而知，如果在附件更多的情况下，性能差异就越明显了，希望对您有所帮助！

```
<?php

$text = remote(array('http://www.jb51.net/','http://www.baidu.com/'));

print_r($text);
function remote($urls) {

    if (!is_array($urls) or count($urls) == 0) {

        return false;

    }
    $curl = $text = array();

    $handle = curl_multi_init();

    foreach($urls as $k => $v) {

        $nurl[$k]= preg_replace('~([^:\/\.]+)~ei', "rawurlencode('\\1')", $v);

        $curl[$k] = curl_init($nurl[$k]);

        curl_setopt($curl[$k], CURLOPT_RETURNTRANSFER, 1);

        curl_setopt($curl[$k], CURLOPT_HEADER, 0);

        curl_multi_add_handle ($handle, $curl[$k]);

    }
    $active = null;

    do {

        $mrc = curl_multi_exec($handle, $active);

    } while ($mrc == CURLM_CALL_MULTI_PERFORM);
    while ($active && $mrc == CURLM_OK) {

        if (curl_multi_select($handle) != -1) {

            do {

                $mrc = curl_multi_exec($handle, $active);

            } while ($mrc == CURLM_CALL_MULTI_PERFORM);

        }

    }
    foreach ($curl as $k => $v) {

        if (curl_error($curl[$k]) == "") {

        $text[$k] = (string) curl_multi_getcontent($curl[$k]);

        }

        curl_multi_remove_handle($handle, $curl[$k]);

        curl_close($curl[$k]);

    }

    curl_multi_close($handle);

    return $text;

}
```

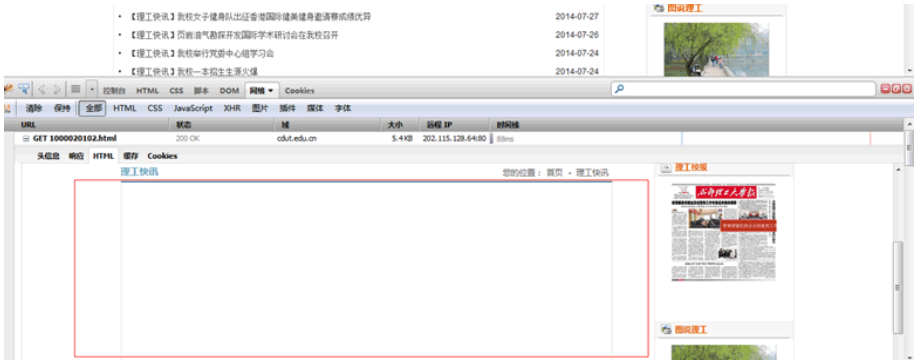
PHP curl 抓取AJAX异步内容示例

其实抓ajax异步内容的页面和抓普通的页面区别不大。ajax只不过是做了一次异步的http请求，只要使用firebug类似的工具，找到请求的后端服务url和传值的参数，然后对该url传递参数进行抓取即可。

利用Firebug的网络工具



如果抓去的是页面，则内容中没有显示的数据，是一堆JS代码。



Code

```
$cookie_file=tempnam('./temp','cookie');
$ch = curl_init();
$url1 = "http://www.cdut.edu.cn/default.html";
curl_setopt($ch,CURLOPT_URL,$url1);
curl_setopt($ch,CURLOPT_HTTP_VERSION,CURL_HTTP_VERSION_1_1);
curl_setopt($ch,CURLOPT_HEADER,0);
curl_setopt($ch,CURLOPT_RETURNTRANSFER,1);
curl_setopt($ch,CURLOPT_FOLLOWLOCATION,1);
curl_setopt($ch, CURLOPT_ENCODING ,'gzip'); //加入gzip解析
//设置连接结束后保存cookie信息的文件
curl_setopt($ch,CURLOPT_COOKIEJAR,$cookie_file);
$content=curl_exec($ch);

curl_close($ch);

$ch3 = curl_init();
$url3 = "http://www.cdut.edu.cn/xww/dwr/call/plaincall/portalAjax.getNewsXml.dwr";
$curlPost = "callCount=1&page=/xww/type/1000020118.html&httpSessionId=12A9B726E6A2D4D3B09DE7952B2F282C&scriptSessionId=295315B4B4141B09DA888D3A3ADB8FAA658&c0-scriptName=portalAjax&c0-methodName=getNewsXml&c0-id=0&c0-param0=string:10000201&c0-param1=string:1000020118&c0-param2=string:news_&c0-param3=number:5969&c0-param4=number:1&c0-param5=null:null&c0-param6=null:null&batchId=0";
curl_setopt($ch3,CURLOPT_URL,$url3);
curl_setopt($ch3,CURLOPT_POST,1);
curl_setopt($ch3,CURLOPT_POSTFIELDS,$curlPost);

//设置连接结束后保存cookie信息的文件
curl_setopt($ch3,CURLOPT_COOKIEFILE,$cookie_file);
$content1=curl_exec($ch3);
curl_close($ch3);
```

php中file_get_content 和curl以及fopen 效率分析

三个函数虽然都是读取资源的函数，但各自的应用场景不同。

curl多用于互联网网页之间的抓取，fopen多用于读取文件，而file_get_contents多用于获取静态页面的内容。

1. fopen /file_get_contents 每次请求都会重新做DNS查询，并不对DNS信息进行缓存。但是CURL会自动对DNS信息进行缓存。对同一域名下的网页或者图片的请求只需要一次DNS查询。这大大减少了DNS查询的次数。所以CURL的性能比fopen /file_get_contents 好很多。

2. fopen /file_get_contents在请求HTTP时，使用的是http_fopen_wrapper，不会keeplive。而curl却可以。这样在多次请求多个链接时，curl效率会好一些。

3. curl可以模拟多种请求，例如：POST数据，表单提交等，用户可以按照自己的需求来定制请求。而fopen / file_get_contents只能使用get方式获取数据。

php之curl实现http与https请求的方法

本文实例讲述了php之curl实现http与https请求的方法，分享给大家供大家参考。具体如下：

通常来说，php的curl函数组可以帮助我们把机器伪装***的行为来抓取网站，下面来分享两个例子，一个是访问http网页，一个访问https网页，一起来看一下。

每次要使用curl的时候，总要查一堆资料。

现在将常用的几句保存下来，省的每次都去谷歌。

常规curl请求：

```
$url = 'http://www.jb51.net';

$curl = curl_init();

curl_setopt($curl, CURLOPT_URL, $url);

curl_setopt($curl, CURLOPT_HEADER, 1);

curl_setopt($curl, CURLOPT_RETURNTRANSFER, 1);

$data = curl_exec($curl);

curl_close($curl);

var_dump($data);
```

使用curl请求HTTPS：

```
$url = 'https://www.jb51.net';

$curl = curl_init();

curl_setopt($curl, CURLOPT_URL, $url);

curl_setopt($curl, CURLOPT_HEADER, 1);

curl_setopt($curl, CURLOPT_RETURNTRANSFER, 1);

curl_setopt($curl, CURLOPT_SSL_VERIFYPEER, false);//这个是重点。

$data = curl_exec($curl);

curl_close($curl);

var_dump($data);
```

注意

当请求https的数据时，会要求证书，这时候，加上下面这两个参数，规避ssl的证书检查

```
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE); // https请求 不验证证书和hosts

curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, FALSE);
```

希望本文所述对大家的PHP程序设计有所帮助。

php之curl设置超时实例

本文实例讲述了php中curl超时设置方法。分享给大家供大家参考。具体实现方法如下：

访问HTTP方式很多，可以使用curl, socket, file_get_contents() 等方法。

在访问http时，需要考虑超时的问题。
CURL访问HTTP：

CURL 是常用的访问HTTP协议接口的lib库，性能高，还有一些并发支持的功能等。

curl_setopt($ch, opt) 可以设置一些超时的设置，主要包括：

① (重要) CURLOPT_TIMEOUT 设置cURL允许执行的最长秒数。

② (重要) CURLOPT_TIMEOUT_MS 设置cURL允许执行的最长毫秒数。

（在cURL 7.16.2中被加入。从PHP 5.2.3起可使用）

③ CURLOPT_CONNECTTIMEOUT 在发起连接前等待的时间，如果设置为0，则无限等待。

④ CURLOPT_CONNECTTIMEOUT_MS 尝试连接等待的时间，以毫秒为单位。如果设置为0，则无限等待。 （在cURL 7.16.2中被加入。从PHP 5.2.3开始可用）

⑤ CURLOPT_DNS_CACHE_TIMEOUT 设置在内存中保存DNS信息的时间，默认为120秒。

1. curl普通秒级超时：

```
$ch = curl_init();
 curl_setopt($ch, CURLOPT_URL,$url);
 curl_setopt($ch, CURLOPT_RETURNTRANSFER,1);
curl_setopt($ch, CURLOPT_TIMEOUT,60);   //只需要设置一个秒的数量就可以
 curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
 curl_setopt($ch, CURLOPT_USERAGENT, $defined_vars['HTTP_USER_AGENT']);
```

2. curl普通秒级超时使用：

```
curl_setopt($ch, CURLOPT_TIMEOUT,60);
```

3. curl如果需要进行毫秒超时，需要增加：

```
curl_easy_setopt(curl, CURLOPT_NOSIGNAL,1L);
 //或者
 curl_setopt ( $ch,  CURLOPT_NOSIGNAL,true);//支持毫秒级别超时设置
```

希望本文所述对大家的PHP程序设计有所帮助。

php的curl封装类用法实例

本文实例讲述了两个php curl封装类的用法实例，这两个函数可以让我们非常的方便的使用php curl相关函数。分享给大家供大家参考。具体如下：

使用函数之前我们要需要把php curl模块打开(libeay32.dll, ssleay32.dll, php5ts.dll, php_curl.dll)

开启php curl函数库的步骤

1).去掉windows/php.ini 文件里;extension=php_curl.dll前面的; /*用 echo phpinfo();查看php.ini的路径*/

2).把php5/libeay32.dll，ssleay32.dll复制到系统目录windows/下

3).重启apache
代码如下：

```
<?php
include_once('curl.class.php');
$aa =new Curl('');
 $curlOptions = array(
 CURLOPT_URL => "http://www.xx.com/addTicket.jsp", //访问URL
 CURLOPT_RETURNTRANSFER => true, //获取结果作为字符串返回
 CURLOPT_REFERER => "ww.ww.ww/zw2",
 CURLOPT_HTTPHEADER => array('X-FORWARDED-FOR:139.197.14.19', 'CLIENT-IP:127.0.0.1','Proxy-Client-IP:139.197.14.19','WL-Proxy-Client-IP:139.197.14.19' ),
 CURLOPT_HEADER => 1, //获取返回头信息
 //CURLOPT_SSL_VERIFYPEER => false, //支持SSL加密
 CURLOPT_POST => true, //发送时带有POST参数
 CURLOPT_POSTFIELDS => 'ids=897&Submit=%E6%8A%95%E7%A5%A8', //请求的POST参数字符串
 CURLOPT_TIMEOUT => $aa->timeout //等待响应的时间
 );
 echo $aa->getResponseText($curlOptions);
```

cul处理类：

```
<?php
class Curl
{
public $cookieFile;
public $timeout = 160;
Public function __construct($dir){
$this->cookieFile = $this->getTemporaryCookieFileName($dir);
}
/**
* 设置CURL参数并发送请求，获取响应内容
* @access private
* @param $curlOptions array curl设置参数数组
* @return string|false 访问成功，按字符串形式返回获取的信息；否则返回false
*/
public function getResponseText($curlOptions) {
/* 设置CURLOPT_RETURNTRANSFER为true */
if(!isset($curlOptions[CURLOPT_RETURNTRANSFER]) || $curlOptions[CURLOPT_RETURNTRANSFER] == false) {
$curlOptions[CURLOPT_RETURNTRANSFER] = true;
}
/* 初始化curl模块 */
```

```
$curl = curl_init();
/* 设置curl选项 */
curl_setopt_array($curl, $curlOptions);
/* 发送请求并获取响应信息 */
$responseText = '';
try {
$responseText = curl_exec($curl);
if(($errno = curl_errno($curl)) != CURLM_OK) {
$errmsg = curl_error($curl);
throw new Exception($errmsg, $errno);
}
} catch (Exception $e) {
//exceptionDisposeFunction($e);
//print_r($e);
$responseText = false;
}
/* 关闭curl模块 */
curl_close($curl);
/* 返回结果 */
return $responseText;
}
/**
* 将Unicode字符串(u0000)转化为utf-8字符串，工具函数
* @access private
* @static
* @param $string string Unicode字符串
* @return string utf-8字符串
*/
public function unicodeToUtf8($string) {
$string = str_replace('u', '', strtolower($string));
$length = strlen($string) / 4;
$stringResult = '';
for($i = 0; $i < $length; $i++) {
$charUnicodeHex = substr($string, $i * 4, 4);
$unicodeCode = hexdec($charUnicodeHex);
$utf8Code = '';
if($unicodeCode < 128) {
$utf8Code = chr($unicodeCode);
} else if($unicodeCode < 2048) {
$utf8Code .= chr(192 + (($unicodeCode - ($unicodeCode % 64)) / 64));
$utf8Code .= chr(128 + ($unicodeCode % 64));
} else {
$utf8Code .= chr(224 + (($unicodeCode - ($unicodeCode % 4096)) / 4096));
$utf8Code .= chr(128 + ((($unicodeCode % 4096) - ($unicodeCode % 64)) / 64));
$utf8Code .= chr(128 + ($unicodeCode % 64));
}
$stringResult .= $utf8Code;
}
return $stringResult;
}
```

```
private function getTemporaryCookieFileName($dir='.') {
return (str_replace("", '/', tempnam($dir, 'tmp')));
}
}
```

例子2

```
<?php
//curl类
class Curl
{
    function Curl(){
        return true;
    }

    function execute($method, $url, $fields='', $userAgent='', $httpHeaders='', $username='', $password=''){
        $ch = Curl::create();
        if(false === $ch){
            return false;
        }
        if(is_string($url) && strlen($url)){
            $ret = curl_setopt($ch, CURLOPT_URL, $url);
        }else{
            return false;
        }
        //是否显示头部信息
        curl_setopt($ch, CURLOPT_HEADER, false);
        //
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
        if($username != ''){
            curl_setopt($ch, CURLOPT_USERPWD, $username . ':' . $password);
        }
        $method = strtolower($method);
        if('post' == $method){
            curl_setopt($ch, CURLOPT_POST, true);
            if(is_array($fields)){
                $sets = array();
                foreach ($fields AS $key => $val){
                    $sets[] = $key . '=' . urlencode($val);
                }
                $fields = implode('&',$sets);
            }
            curl_setopt($ch, CURLOPT_POSTFIELDS, $fields);
        }else if('put' == $method){
            curl_setopt($ch, CURLOPT_PUT, true);
        }
        //curl_setopt($ch, CURLOPT_PROGRESS, true);
        //curl_setopt($ch, CURLOPT_VERBOSE, true);
        //curl_setopt($ch, CURLOPT_MUTE, false);
```

```
        curl_setopt($ch, CURLOPT_TIMEOUT, 10);//设置curl超时秒数
        if(strlen($userAgent)){
            curl_setopt($ch, CURLOPT_USERAGENT, $userAgent);
        }
        if(is_array($httpHeaders)){
            curl_setopt($ch, CURLOPT_HTTPHEADER, $httpHeaders);
        }
        $ret = curl_exec($ch);
        if(curl_errno($ch)){
            curl_close($ch);
            return array(curl_error($ch), curl_errno($ch));
        }else{
            curl_close($ch);
            if(!is_string($ret) || !strlen($ret)){
                return false;
            }
            return $ret;
        }
    }

    function post($url, $fields, $userAgent = '', $httpHeaders = '', $username = '', $password = ''){
        $ret = Curl::execute('POST', $url, $fields, $userAgent, $httpHeaders, $username, $password);
        if(false === $ret){
            return false;
        }
        if(is_array($ret)){
            return false;
        }
        return $ret;
    }

    function get($url, $userAgent = '', $httpHeaders = '', $username = '', $password = ''){
        $ret = Curl::execute('GET', $url, '', $userAgent, $httpHeaders, $username, $password);
        if(false === $ret){
            return false;
        }
        if(is_array($ret)){
            return false;
        }
        return $ret;
    }

    function create(){
        $ch = null;
        if(!function_exists('curl_init')){
            return false;
        }
        $ch = curl_init();
        if(!is_resource($ch)){
            return false;
```

```
            return false;
        }
        return $ch;
    }
}
?>
```

用法
GET用法：

```
$curl = new Curl();
$curl->get('http://www.jb51.net/');
```

POST用法：

```
$curl = new Curl();
$curl->get('http://www.jb51.net/', 'p=1&time=0´);
```

希望本文所述对大家的PHP程序设计有所帮助。

php采用curl模仿登录人人网发布动态的方法

本文实例讲述了php采用curl模仿登录人人网发布动态的方法。分享给大家供大家参考。具体实现方法如下：

说到php中模仿登录很多人第一时间会想到curl函数系列了，这个没错本例子也是使用curl模仿登录之后再进行动态发布，原理也简单我们只要抓取人人网的登录信息，然后再由curl post登录数据上去就可以了。

具体代码如下：

```
$rconfig = pdo_fetch("SELECT * FROM ".tablename("eduTwo_renren")." WHERE weid = :weid",array(':weid'=>$_W['weid']));

$cookie_file = dirname(__FILE__)."/renren.cookie";

$login_url = 'http://passport.renren.com/PLogin.do';

$post_fields['email'] = $rconfig['rusername'];

$post_fields['password'] = $rconfig['rpassword'];

$post_fields['origURL'] = 'http%3A%2F%2Fhome.renren.com%2FHome.do';

$post_fields['domain'] = 'renren.com';
$ch = curl_init($login_url);

curl_setopt($ch, CURLOPT_USERAGENT, 'Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN; rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5');

curl_setopt($ch, CURLOPT_HEADER, 0);

curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

curl_setopt($ch, CURLOPT_MAXREDIRS, 1);

curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);

curl_setopt($ch, CURLOPT_AUTOREFERER, 1);

curl_setopt($ch, CURLOPT_POST, 1);

curl_setopt($ch, CURLOPT_POSTFIELDS, $post_fields);

curl_setopt($ch, CURLOPT_COOKIEJAR, $cookie_file);

$content = curl_exec($ch);

$info = curl_getinfo($ch);

curl_close($ch);

//var_dump($info);exit;

//匹配用户的ID

$send_url='http://www.renren.com/home';

$ch = curl_init($send_url);

curl_setopt($ch, CURLOPT_HEADER, 0);

curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

curl_setopt($ch, CURLOPT_COOKIEFILE, $cookie_file);

curl_exec($ch);

$info = curl_getinfo($ch);

curl_close($ch);
//$uid = "305115027";

//获取token和rtk

$send_url=$info['redirect_url'];

$ch = curl_init($send_url);

curl_setopt($ch, CURLOPT_HEADER, 0);

curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

curl_setopt($ch, CURLOPT_COOKIEFILE, $cookie_file);

$tmp = curl_exec($ch);

curl_close($ch);

preg_match_all("/get_check:'(.*?)',get_check_x:'(.*?)',/is",$tmp,$arr);

preg_match_all("/'ruid':'(.*?)',/is",$tmp,$utmp);

//var_dump($utmp);exit;

$token = $arr[1][0];//1121558104

$rtk = $arr[2][0];//e9a9cb2
```

```
$uid = $utmp[1][0];
//echo $token;exit;
//发布信息
$poststr['content'] = $_GPC['content'].$rconfig['tail'];
$poststr['withInfo'] = '{"wpath":[]}';
$poststr['hostid:'] = $uid;
$poststr['privacyParams'] = '{"sourceControl": 99}';
$poststr['requestToken'] = $token;
$poststr['_rtk'] = $rtk;
$poststr['channel'] = "renren";
$head = array(
 'Referer:http://shell.renren.com/ajaxproxy.htm',
 'X-Requested-With:XMLHttpRequest',
);
$ch = curl_init("http://shell.renren.com/{$uid}/status");
curl_setopt($ch, CURLOPT_USERAGENT, 'Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN; rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5');
curl_setopt($ch,CURLOPT_HTTPHEADER,$head);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_MAXREDIRS, 1);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
curl_setopt($ch, CURLOPT_AUTOREFERER, 1);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $poststr);
curl_setopt($ch, CURLOPT_COOKIEFILE, $cookie_file);
$content = curl_exec($ch);
curl_close($ch);
//echo $content;exit;
$data = json_decode($content,true);
if($data["code"] == "0"){
 echo "发布成功！";
}else{
 echo "shit !!!";
}
```

最后就发布成功了，当然前面的数据库需要自己写一个吧，非常的简单的一个记录库也是你要发布的信息。录数据上去就可以了。

希望本文所述对大家的PHP程序设计有所帮助。

PHP基于CURL进行POST数据上传实例

本文实例讲述了PHP基于CURL进行POST数据上传的方法。分享给大家供大家参考。具体实现方法如下：

```
////二维码
$QRCode_URL="https://api.weixin.qq.com/cgi-bin/qrcode/create?access_token=".$ACC_TOKEN;


$data ='{"expire_seconds": 1800, "action_name": "QR_SCENE", "action_info": {"scene": {"scene_id": 123}}} ';
/*
$ch = curl_init($MENU_URL);
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");
curl_setopt($ch, CURLOPT_POSTFIELDS, $data);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-Type: application/json','Content-Length:'.strlen($data)));
$info = curl_exec($ch);
*/
function post($url, $params = false, $header = array()){
$ch = curl_init();
$cookieFile = 'sdadsd_cookiejar.txt';


curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, 60);
curl_setopt($ch, CURLOPT_COOKIEJAR, $cookieFile);
curl_setopt($ch, CURLOPT_COOKIEFILE,$cookieFile);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER,FALSE);
curl_setopt($ch, CURLOPT_HTTPGET, true);
curl_setopt($ch, CURLOPT_TIMEOUT, 30);
if($params !== false){ curl_setopt($ch, CURLOPT_POSTFIELDS , $params);}
curl_setopt($ch, CURLOPT_USERAGENT,'Mozilla/5.0 (Windows NT 5.1; rv:21.0) Gecko/20100101 Firefox/21.0');
curl_setopt($ch, CURLOPT_URL,$url);
curl_setopt($ch, CURLOPT_HTTPHEADER, $header);
$result = curl_exec($ch);
curl_close($ch);


return $result;
}
$result = post($QRCode_URL,$data);
```

希望本文所述对大家的PHP程序设计有所帮助。

PHP中让curl支持sock5的代码实例

```
//最近需要用到curl测试代理是否可用，代理是sock5非http的 所以需要在curl中增加几句。


curl_setopt($ch, CURLOPT_PROXYTYPE, CURLPROXY_SOCKS5);

curl_setopt($ch, CURLOPT_PROXY, "0.0.0.0:8080");

curl_setopt($ch,CURLOPT_PROXYUSERPWD, "username:pwd");


//测试ok 速度很快 哈哈
```

PHP中CURL的几个经典应用实例

1、cURL请求的基本步骤：

（1）初始化

（2）设置选项，包括URL

（3）执行并获取HTML文档内容

（4）释放cURL句柄

```
<?php

    //1、初始化

    $ch = curl_init();
    //2、设置选项，包括URL

    curl_setopt($ch, CURLOPT_URL, "http://www.cnblogs.com/it-cen/");
    //将curl_exec()获取的信息以文件流的形式返回，而不是直接输出

    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    //启动时会将头文件的信息作为数据流输出

    curl_setopt($ch, CURLOPT_HEADER, 1);
    //3、执行并获取HTML文档内容

    curl_exec($ch);
    //4、释放句柄

    curl_close($ch);
    echo $ch;

?>
```

注意：第二步最重要，也就是curl_setopt()函数

我们可以加一段检查错误的语句，这里要注意用的是"===false",这是为了区分空输出和布尔值false

```
$output = curl_exec($ch);



    if ($output === false) {

        echo "cURL Error:".curl_error($ch);

    }
```

curl_getinfo()函数返回cURL执行后这一请求相关的信息，这对调试和排错很有用：

```
curl_exec($ch);

    $info = curl_getinfo($ch);

    echo '<pre>';

    print_r($info);

    echo '</pre>';
```

返回的数据

```
Array
(
    [url] => http://www.cnblogs.com/it-cen/
    [content_type] => text/html; charset=utf-8
    [http_code] => 200
    [header_size] => 312
    [request_size] => 61
    [filetime] => -1
    [ssl_verify_result] => 0
    [redirect_count] => 0
    [total_time] => 0.172
    [namelookup_time] => 0.016
    [connect_time] => 0.063
    [pretransfer_time] => 0.063
    [size_upload] => 0
    [size_download] => 14658                <span style="color: #ff0000;">    //请求的数据大小</span>
    [speed_download] => 85220
    [speed_upload] => 0
    [download_content_length] => 14658
    [upload_content_length] => 0
    [starttransfer_time] => 0.125
    [redirect_time] => 0
    [certinfo] => Array
        (
        )

    [redirect_url] =>
)
```

2、这些信息在调试很有用，例如在cURL抓取的时候，可能由于网络等原因，时常出现抓取数据不完整的情况，这是我们可以通过所获取的数据计算filesize，然后和curl_getinfo()获取的进行比较，如果大小相等，就认定下载正确，否则进行重复尝试。

下面我们看一个抓取图片的例子：

```php
<?php

    header("Content-Type: image/png");
    //1、初始化

    $ch = curl_init();
    //2、设置选项，包括URL

    curl_setopt($ch, , CURLOPT_URL, "http://img04.taobaocdn.com/tfscom/TB1omaTHXXXXXajXVXXtKXbFXXX.png");

    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

    curl_setopt($ch, CURLOPT_HEADER, 1) ;
    //3、执行并获取内容

    $res = curl_exec($ch);
    //获取信息

    $info = curl_getinfo($ch);
    //4、释放资源

    curl_close($ch);
    file_put_contents("d:/aa.png", $res);

    $size = filesize("d:/aa.png");

    if ($size != $info['size_download']) {

        echo "下载的数据不完整，请重新下载";

    } else {

        echo "下载数据完整";

    }
?>
```

3、在cURL中用POST方法发送数据

```php
<?php

    $ch = curl_init();
    $data = array('name'=>'kelly', 'age'=>27, 'sex'=>1);

    curl_setopt($ch, CURLOPT_URL, "http://localhost.post.php");

    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    //设置为post

    curl_setopt($ch, CURLOPT_POST, 1);

    //把post的变量加上

    curl_setopt($ch, CURLOPT_POSTFIELDS,$data);
    $res = curl_exec($ch);

    curl_close($ch);

    echo $res;
?>
```

用此方法可以模拟留言，或者可以坐灌水机器人，思路都是一样的

4、用cURL上传文件

```php
<?php

    //索要上传的数据

    $data = array('name'=>'beauty', "upload"=>"@a.zip");
    $ch = curl_init();

    curl_setopt($ch, CURLOPT_URL, "http://127.0.0.1/Socket/upload_file.php");

    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($ch, CURLOPT_POST, 1);

    curl_setopt($ch, CURLOPT_POSTFIELDS, $data);
    $res = curl_exec($ch);

    curl_close($ch);

    echo $res;

?>
```

注意：要发送文件时，要在文件名前面加上 @ 前缀并使用完整路径

5、cURL设置项

其实，cURL有许多配置选项，这些选项才是cURL的灵魂，通过setopt()设置，下面总结几个比较常见且重要的配置项，希望在对读者在以后用到cURL时有一定的帮助：
CURLOPT_AUTOREFERER:当根据location:重定向时，自动设置header中的Referer:信息
CURLOPT_COOKIESESSION:启用时cURL会紧紧传递一个sessioncookie，忽略其他cookie

CURLOPT_HEADER:将头文件的信息作为数据流输出

CURLOPT_INFILESIZE:设置上传文件的大小，单位为字节

CURLOPT_MAXCONNECTS:允许最大连接数量

CURLOPT_MAXREDIRS:指定HTTP重定向的最多数量

CURLOPT_COOKIE:设置HTTP请求中“cookie:”部分的内容，多个cookie用分号跟个，分号后带一个空格

CURLOPT_POSTFIELDS:全部数据用HTTP协议中的“POST”操作发送要发送文件，在文件名前面加上@前缀并使用完整路径

.....

具体更多配置项请参考PHP手册

cURL功能很强大，它是一个通用的库，并非PHP独有。

希望读者通过本博文的几个经典cURL例子的学习能有所收获。

PHP使用CURL函数获取HTTPS网页及POST数据示例

```
function vpost($url,$data){ // 模拟提交数据函数
    $curl = curl_init(); // 启动一个CURL会话
    curl_setopt($curl, CURLOPT_URL, $url); // 要访问的地址
    curl_setopt($curl, CURLOPT_SSL_VERIFYPEER, 0); // 对认证证书来源的检查
    curl_setopt($curl, CURLOPT_SSL_VERIFYHOST, 1); // 从证书中检查SSL加密算法是否存在
    curl_setopt($curl, CURLOPT_USERAGENT, $_SERVER['HTTP_USER_AGENT']); // 模拟用户使用的浏览器
    curl_setopt($curl, CURLOPT_FOLLOWLOCATION, 1); // 使用自动跳转
    curl_setopt($curl, CURLOPT_AUTOREFERER, 1); // 自动设置Referer
    curl_setopt($curl, CURLOPT_POST, 1); // 发送一个常规的Post请求
    curl_setopt($curl, CURLOPT_POSTFIELDS, $data); // Post提交的数据包
    curl_setopt($curl, CURLOPT_TIMEOUT, 30); // 设置超时限制防止死循环
    curl_setopt($curl, CURLOPT_HEADER, 0); // 显示返回的Header区域内容
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, 1); // 获取的信息以文件流的形式返回
    $tmpInfo = curl_exec($curl); // 执行操作
    if (curl_errno($curl)) {
       echo 'Errno'.curl_error($curl);//捕抓异常
    }
    curl_close($curl); // 关闭CURL会话
    return $tmpInfo; // 返回数据
}
```

PHP官方文档：CURL库函数大全

PHP中巧用curl 并发减少获取第三方网页内容时间

前言：

在我们平时的程序中难免出现同时访问几个接口的情况，平时我们用curl进行访问的时候，一般都是单个、顺序访问，假如有3个接口，每个接口耗时500毫秒那么我们三个接口就要花费1500毫秒了，这个问题太头疼了严重影响了页面访问速度，有没有可能并发访问来提高速度呢？今天就简单的说一下，利用curl并发来提高页面访问速度， 希望大家多指导。

1、老的curl访问方式以及耗时统计

```
<?php
function curl_fetch($url, $timeout=3){
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_TIMEOUT, $timeout);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    $data = curl_exec($ch);
    $errno = curl_errno($ch);
    if ($errno>0) {
        $data = false;
    }
    curl_close($ch);
    return $data;
}
function microtime_float()
{
   list($usec, $sec) = explode(" ", microtime());
   return ((float)$usec + (float)$sec);
}
$url_arr=array(
     "taobao"=>"http://www.taobao.com",
     "sohu"=>"http://www.sohu.com",
     "sina"=>"http://www.sina.com.cn",
);
$time_start = microtime_float();
$data=array();
foreach ($url_arr as $key=>$val)
{
     $data[$key]=curl_fetch($val);
}
$time_end = microtime_float();
$time = $time_end - $time_start;
echo "耗时:{$time}";
?>
```

耗时:0.614秒

2、curl并发访问方式以及耗时统计

```php
<?php
function curl_multi_fetch($urlarr=array()){
    $result=$res=$ch=array();
    $nch = 0;
    $mh = curl_multi_init();
    foreach ($urlarr as $nk => $url) {
        $timeout=2;
        $ch[$nch] = curl_init();
        curl_setopt_array($ch[$nch], array(
            CURLOPT_URL => $url,
            CURLOPT_HEADER => false,
            CURLOPT_RETURNTRANSFER => true,
            CURLOPT_TIMEOUT => $timeout,
        ));
        curl_multi_add_handle($mh, $ch[$nch]);
        ++$nch;
    }
    /* wait for performing request */
    do {
        $mrc = curl_multi_exec($mh, $running);
    } while (CURLM_CALL_MULTI_PERFORM == $mrc);

    while ($running && $mrc == CURLM_OK) {
        // wait for network
        if (curl_multi_select($mh, 0.5) > -1) {
            // pull in new data;
            do {
                $mrc = curl_multi_exec($mh, $running);
            } while (CURLM_CALL_MULTI_PERFORM == $mrc);
        }
    }

    if ($mrc != CURLM_OK) {
        error_log("CURL Data Error");
    }

    /* get data */
    $nch = 0;
    foreach ($urlarr as $moudle=>$node) {
        if (($err = curl_error($ch[$nch])) == '') {
            $res[$nch]=curl_multi_getcontent($ch[$nch]);
            $result[$moudle]=$res[$nch];
        }
        else
        {
            error_log("curl error");
        }
        curl_multi_remove_handle($mh,$ch[$nch]);
        curl_close($ch[$nch]);
        ++$nch;
    }
    curl_multi_close($mh);
    return  $result;
}
$url_arr=array(
     "taobao"=>"http://www.taobao.com",
     "sohu"=>"http://www.sohu.com",
     "sina"=>"http://www.sina.com.cn",
     );
function microtime_float()
{
   list($usec, $sec) = explode(" ", microtime());
   return ((float)$usec + (float)$sec);
}
$time_start = microtime_float();
$data=curl_multi_fetch($url_arr);
$time_end = microtime_float();
$time = $time_end - $time_start;
 echo "耗时:{$time}";
?>
```

耗时:0.316秒 帅气吧整个页面访问后端接口的时间节省了一半

原文：http://www.searchtb.com/2010/12/using-multicurl-to-improve-performance.html

如何通过命令查看CURL慢在哪里？

很多时候使用CURL发现响应不够快，想优化。但是不知道到底是慢在哪里。下面这条命令就能帮你找到哪里慢，请自行把URL替换成需要测试的URL。

curl -o /dev/null -s -w %{time_connect}:%{time_starttransfer}:%{time_total}
结果：

0.223:1.110:1.772
计时器 描述：

time_connect 建立到服务器的 TCP 连接所用的时间

time_starttransfer 在发出请求之后，Web 服务器返回数据的第一个字节所用的时间

time_total 完成请求所用的时间

这些计时器都相对于事务的起始时间，甚至要先于 Domain Name Service（DNS）查询。因此，在发出请求之后，Web 服务器处理请求并开始发回数据所用的时间是 1.110 – 0.223 = 0.887 秒。客户机从服务器下载数据所用的时间是 1.772 – 1.110 = 0.662 秒。

PHP+Curl伪造客户端获取页面方法

```
public function getUrlContent($url){
        // 初始化一个curl会话
        $ch = curl_init();
        curl_setopt( $ch, CURLOPT_URL, $url);
        curl_setopt( $ch, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt( $ch, CURLOPT_CONNECTTIMEOUT, 5);
        curl_setopt( $ch, CURLOPT_SSL_VERIFYPEER, false);
        curl_setopt( $ch, CURLOPT_REFERER, 'http://www.phpxs.com');
        curl_setopt( $ch, CURLOPT_USERAGENT, 'Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.2) Gecko/20090729 Firefox/3.5.2 GTB5' );
        curl_setopt( $ch, CURLOPT_POST, 1); //设置为POST方式
        curl_setopt( $ch, CURLOPT_POSTFIELDS, array()); //数据传输
        curl_setopt( $ch, CURLOPT_FOLLOWLOCATION, 1 ); //解决重定向问题
        curl_setopt( $ch, CURLOPT_COOKIE, 'redirectLogin=3;t=1766da7fa03df9fdb66af1ebaa160ecc;');
        // 执行一个curl会话
        $contents = curl_exec($ch);
        // 返回一个保护当前会话最近一次错误的字符串
        $error = curl_error($ch);
        if($error){
            echo 'Error: '.$error;
        }
        // 关闭一个curl会话
        curl_close( $ch );
        return $contents;
    }
```

php curl登陆qq后获取用户信息时证书错误

今晚开放ecmall商城的QQ登陆功能，在回调时产生错误，file_get_contents函数执行时，没有抓取到正确的信息，于是改用curl，但是提示证书错误。

在网上找到了解决方法，就是去掉证书认证。

```
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);

curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, FALSE);
```

该种方法可行。

网上还有另外一种说法，就是使用

```
curl_setopt($ch, CURLOPT_HTTPHEADER, array('Expect:'));
```

但是测试结果是错误的。

非常简单的解决办法，不过解决起来却花了不少的功夫查资料，这里记录下来，分享给大家。

php中curl使用指南

许多同学在第一次使用curl的时候感觉一个头两个大（包括我在内），看着这一条条的curl_setopt函数完全摸不着头脑，不过在你花10分钟看了我的介绍后相信你以后也能轻松戏耍php的curl了

首先，请看一个curl代码（花10秒钟，略看一遍，然后跳到后文）

```
<?php
$data = "<soap:Envelope>[...]</soap:Envelope>";
$tuCurl = curl_init();
curl_setopt($tuCurl, CURLOPT_URL, "https://example.com/path/for/soap/url/");
curl_setopt($tuCurl, CURLOPT_PORT , 443);
curl_setopt($tuCurl, CURLOPT_VERBOSE, 0);
curl_setopt($tuCurl, CURLOPT_HEADER, 0);
curl_setopt($tuCurl, CURLOPT_SSLVERSION, 3);
curl_setopt($tuCurl, CURLOPT_SSLCERT, getcwd() . "/client.pem");
curl_setopt($tuCurl, CURLOPT_SSLKEY, getcwd() . "/keyout.pem");
curl_setopt($tuCurl, CURLOPT_CAINFO, getcwd() . "/ca.pem");
curl_setopt($tuCurl, CURLOPT_POST, 1);
curl_setopt($tuCurl, CURLOPT_SSL_VERIFYPEER, 1);
curl_setopt($tuCurl, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($tuCurl, CURLOPT_POSTFIELDS, $data);
curl_setopt($tuCurl, CURLOPT_HTTPHEADER, array("Content-Type: text/xml","SOAPAction: \"/soap/action/query\"", "Content-length: ".strlen($data)));
$tuData = curl_exec($tuCurl);
if(!curl_errno($tuCurl)){
  $info = curl_getinfo($tuCurl);
  echo 'Took ' . $info['total_time'] . ' seconds to send a request to ' . $info['url'];
} else {
  echo 'Curl error: ' . curl_error($tuCurl);
}
curl_close($tuCurl);
echo $tuData;
?>
```

WTF，这到底是在做什么？

想要学会这种“高端”的用法吗？

首先，相信你肯定知道网址大部分是由http开头的，那是因为他们需用通过http（超文本传送协议 HTTP-Hypertext transfer protocol)来进行数据传输，但是传输数据不是简单的将一句"Hello"传到服务器上就搞定的事情，发送者为了方便接受者理解发送者的实际意图以及知道发送人到底是何许人也，发送者往往要将许多额外信息一并发给接受者，就像寄信人需要在信件外套一个信封一样，信封上写着各种发信人的信息。所有的这些最终合并成了一个叫做报文(message)的玩意，也就构成了整个互联网的基础。

curl的工作就是通过http协议发送这些message (php的libcurl目前还支持https、ftp、telnet等其他协议)

现在再看代码，实际上代码只做了五件事情

curl_init()初始化curl

curl_setopt() 设置传输数据和参数

curl_exec()执行传输并获取返回数据

curl_errono()返回错误码

curl_close()关闭curl

下面给出使用GET和POST方法如何抓取和提交任意页面的数据

```
<?php
    //初始化
    $curl = curl_init();
    //设置url
    curl_setopt($curl, CURLOPT_URL, 'http://www.baidu.com');
    //设置返回获取的输出为文本流
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
    //执行命令
    $data = curl_exec($curl);
    //关闭URL请求
    curl_close($curl);
    //显示获得的数据
    print_r($data);
?>
<?php
    //初始化
    $curl = curl_init();
    //设置url
    curl_setopt($curl, CURLOPT_URL, 'http://www.baidu.com');
    //设置返回获取的输出为文本流
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
    //设置post方式提交
    curl_setopt($curl, CURLOPT_POST, 1);
    //设置post数据
    curl_setopt($curl, CURLOPT_POSTFIELDS, array("data"=>"value");
    //执行命令
    $data = curl_exec($curl);
    //关闭URL请求
    curl_close($curl);
    //打印数据
    print_r($data);
?>
```

感兴趣的同学还可以参考php官方文档，学习更多curl用法

PHP的cURL库简介及使用示例

使用PHP的cURL库可以简单和有效地去抓网页。你只需要运行一个脚本，然后分析一下你所抓取的网页，然后就可以以程序的方式得到你想要的数据了。无论是你想从从一个链接上取部分数据，或是取一个XML文件并把其导入数据库，那怕就是简单的获取网页内容，cURL 是一个功能强大的PHP库。

PHP中的CURL函数库（Client URL Library Function）

```
curl_close — 关闭一个curl会话

curl_copy_handle — 拷贝一个curl连接资源的所有内容和参数

curl_errno — 返回一个包含当前会话错误信息的数字编号

curl_error — 返回一个包含当前会话错误信息的字符串

curl_exec — 执行一个curl会话

curl_getinfo — 获取一个curl连接资源句柄的信息

curl_init — 初始化一个curl会话

curl_multi_add_handle — 向curl批处理会话中添加单独的curl句柄资源

curl_multi_close — 关闭一个批处理句柄资源

curl_multi_exec — 解析一个curl批处理句柄

curl_multi_getcontent — 返回获取的输出的文本流

curl_multi_info_read — 获取当前解析的curl的相关传输信息

curl_multi_init — 初始化一个curl批处理句柄资源

curl_multi_remove_handle — 移除curl批处理句柄资源中的某个句柄资源

curl_multi_select — Get all the sockets associated with the cURL extension, which can then be "selected"

curl_setopt_array — 以数组的形式为一个curl设置会话参数

curl_setopt — 为一个curl设置会话参数

curl_version — 获取curl相关的版本信息

curl_init()函数的作用初始化一个curl会话，curl_init()函数唯一的一个参数是可选的，表示一个url地址。

curl_exec()函数的作用是执行一个curl会话，唯一的参数是curl_init()函数返回的句柄。

curl_close()函数的作用是关闭一个curl会话，唯一的参数是curl_init()函数返回的句柄。
```

例子一: 基本例子

```
< ?php

// 初始化一个 cURL 对象

$curl = curl_init();

// 设置你需要抓取的URL

curl_setopt($curl, CURLOPT_URL, 'http://www.cmx8.cn');

// 设置header

curl_setopt($curl, CURLOPT_HEADER, 1);

// 设置cURL 参数，要求结果保存到字符串中还是输出到屏幕上。

curl_setopt($curl, CURLOPT_RETURNTRANSFER, 1);

// 运行cURL，请求网页

$data = curl_exec($curl);

// 关闭URL请求

curl_close($curl);

// 显示获得的数据

var_dump($data);

?>
```

例子二: POST数据

sendSMS.php，其可以接受两个表单域，一个是电话号码，一个是短信内容。

```
< ?php
$phoneNumber = '13812345678';
$message = 'This message was generated by curl and php';
$curlPost = 'pNUMBER=' . urlencode($phoneNumber) . '&MESSAGE=' . urlencode($message) . '&SUBMIT=Send';
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, 'http://www.lxvoip.com/sendSMS.php');
curl_setopt($ch, CURLOPT_HEADER, 1);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $curlPost);
$data = curl_exec();
curl_close($ch);
? >
```

例子三:使用代理服务器

```
< ?php
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, 'http://www.cmx8.cn');
curl_setopt($ch, CURLOPT_HEADER, 1);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_HTTPPROXYTUNNEL, 1);
curl_setopt($ch, CURLOPT_PROXY, 'proxy.lxvoip.com:1080');
curl_setopt($ch, CURLOPT_PROXYUSERPWD, 'user:password');
$data = curl_exec();
curl_close($ch);
? >
```

例子四: 模拟登录

Curl 模拟登录 discuz 程序,适合DZ7.0,将username改成你的用户名,userpass改成你的密码就可以了.

```
<?php
/**
* Curl 模拟登录 discuz 程序
* 尚未实现开启验证码的的论坛登录功能
*/
!extension_loaded('curl') && die('The curl extension is not loaded.');
$discuz_url = 'http://www.lxvoip.com';//论坛地址
$login_url = $discuz_url .'/logging.php?action=login';//登录页地址
$get_url = $discuz_url .'/my.php?item=threads'; //我的帖子
$post_fields = array();
//以下两项不需要修改
$post_fields['loginfield'] = 'username';
$post_fields['loginsubmit'] = 'true';
//用户名和密码，必须填写
$post_fields['username'] = 'lxvoip';
$post_fields['password'] = '88888888';
//安全提问
$post_fields['questionid'] = 0;
$post_fields['answer'] = '';
```

```
//@todo验证码
$post_fields['seccodeverify'] = '';
//获取表单FORMHASH
$ch = curl_init($login_url);
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
$contents = curl_exec($ch);
curl_close($ch);
preg_match('/<input\s*type="hidden"\s*name="formhash"\s*value="(.*?)"\s*\/>/i', $contents, $matches);
if(!empty($matches)) {
    $formhash = $matches[1];
} else {
    die('Not found the forumhash.');
}
//POST数据，获取COOKIE
$cookie_file = dirname(__FILE__) . '/cookie.txt';
//$cookie_file = tempnam('/tmp');
$ch = curl_init($login_url);
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $post_fields);
curl_setopt($ch, CURLOPT_COOKIEJAR, $cookie_file);
curl_exec($ch);
curl_close($ch);
//带着上面得到的COOKIE获取需要登录后才能查看的页面内容
$ch = curl_init($get_url);
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 0);
curl_setopt($ch, CURLOPT_COOKIEFILE, $cookie_file);
$contents = curl_exec($ch);
curl_close($ch);
var_dump($contents);
?>
```

以上就是本文的全部内容了，希望大家能够喜欢。

PHP curl CURLOPT_RETURNTRANSFER参数的作用使用实例

获取页面内容，不直接输出到页面，CURLOPT_RETURNTRANSFER参数设置

使用PHP curl获取页面内容或提交数据，有时候希望返回的内容作为变量储存，而不是直接输出。这个时候就必需设置curl的CURLOPT_RETURNTRANSFER选项为1或true。

1、curl获取页面内容, 直接输出例子：

```
<?php

$url = 'http://www.lai18.com/';
$ch = curl_init();

curl_setopt($ch, CURLOPT_URL, $url);

curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);

curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, false);

curl_exec($ch);

curl_close($ch);

?>
```

2、curl获取页面内容, 不直接输出例子：

```
<?php

$url = 'http://www.lai18.com/';
$ch = curl_init();

curl_setopt($ch, CURLOPT_URL, $url);

curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);

curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, false);

curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

$response = curl_exec($ch); // 已经获取到内容，没有输出到页面上。

curl_close($ch);

echo $response;

?>
```

Linux中使用curl命令访问https站点4种常见错误和解决方法

每一种客户端在处理https的连接时都会使用不同的证书库。IE浏览器和FireFox浏览器都可以在本浏览器的控制面板中找到证书管理器。在证书管理器中可以自由添加、删除根证书。

而Linux的curl使用的证书库在文件"/etc/pki/tls/certs/ca-bundle.crt"中。（CentOS）

以下是curl在访问https站点时常见的报错信息

1.Peer's Certificate issuer is not recognized

```
[root@ip-172-31-32-208 Nginx]# curl https://m.ipcpu.com curl: (60) Peer's Certificate issuer is not recognized.
more details here: http://curl.haxx.se/docs/sslcerts.html
```

此种情况多发生在自签名的证书，报错含义是签发证书机构未经认证，无法识别。

解决办法是将签发该证书的私有CA公钥cacert.pem文件内容，追加到/etc/pki/tls/certs/ca-bundle.crt。

我们在访问12306.cn订票网站时也报了类似的错误。

```
[root@ip-172-31-32-208 ~]# curl https://kyfw.12306.cn/ curl: (60) Peer's certificate issuer has been marked as not trusted by the user.
More details here: http://curl.haxx.se/docs/sslcerts.html
```

2.SSL routines:SSL3_GET_SERVER_CERTIFICATE:certificate verify failed

```
[root@GO-EMAIL-1 aa]# curl https://github.com/ curl: (60) SSL certificate problem, verify that the CA cert is OK. Details:
error:14090086:SSL routines:SSL3_GET_SERVER_CERTIFICATE:certificate verify failed
More details here: http://curl.haxx.se/docs/sslcerts.html
```

此问题多是由于本地CA证书库过旧，导致新签发证书无法识别。

经排查，github.com证书是由GTE CyberTrust Root签发,现行证书时间是：

1.不早于(1998/8/13 0:29:00 GMT)
2.不晚于(2018/8/13 23:59:00 GMT)

而在我们的Redhat5.3系统中ca-bundle.crt文件发现，GTE CyberTrust Root的时间已经过期。

```
Issuer: C=US, O=GTE Corporation, CN=GTE CyberTrust Root
Validity
Not Before: Feb 23 23:01:00 1996 GMT
Not After : Feb 23 23:59:00 2006 GMT
```

解决办法是更新本地CA证书库。

方法一：

下载http://curl.haxx.se/ca/cacert.pem 替换/etc/pki/tls/certs/ca-bundle.crt

方法二：

使用update-ca-trust 更新CA证书库。（CentOS6，属于ca-certificates包）

3.unknown message digest algorithm

```
[root@WEB_YF_2.7 ~]#curl https://www.alipay.com curl: (35) error:0D0C50A1:asn1 encoding routines:ASN1_item_verify:unknown message digest algorithm
```

此问题多由证书本地openssl不能识别SSL证书签名算法所致。www.alipay.com 使用了SHA-256 RSA 加密算法。而openssl在OpenSSL 0.9.8o才加入此算法。

解决办法是升级本地openssl。

在我的操作系统RedHat5.3中,yum 升级openssl到openssl-0.9.8e-22.el5 就可以识别SHA-256算法。原因是Redhat每次都是给0.9.8e打补丁，而不是直接更换版本。在srpm包中我找到了这个补丁。

```
Summary: The OpenSSL toolkit
Name: openssl
Version: 0.9.8e
...
Patch89: openssl-fips-0.9.8e-ssl-sha256.patch
```

4.J*A和PHP的问题**

java和php都可以编程来访问https网站。例如httpclient等。

其调用的CA根证书库并不和操作系统一致。

J***A的CA根证书库是在 JRE的$J***A_HOME/jre/lib/security/cacerts，该文件会随着JRE版本的升级而升级。可以使用keytool工具进行管理。

PHP这边我没有进行测试，从php安装curl组件的过程来看，极有可能就是直接采用的操作系统curl一直的数据。

当然PHP也提供了 curl.cainfo 参数(php.ini)来指定CA根证书库的位置。

php使用curl获取https请求的方法

本文实例讲述了php使用curl获取https请求的方法。分享给大家供大家参考。具体分析如下：

今日在做一个项目，需要curl获取第三方的API，对方的API是https方式的。

之前使用curl能获取http请求，但今天获取https请求时，出现了以下的错误提示：证书验证失败。
SSL certificate problem, verify that the CA cert is OK. Details: error:14090086:SSL routines:SSL3_GET_SERVER_CERTIFICATE:certificate verify failed

解决方法为在curl请求时，加入：

```
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false); // 跳过证书检查

curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, true);  // 从证书中检查SSL加密算法是否存在
```

curl https请求代码

```php
<?php
/** curl 获取 https 请求
* @param String $url          请求的url
* @param Array  $data         要發送的數據
* @param Array  $header       请求时发送的header
* @param int    $timeout      超时时间，默认30s
*/
function curl_https($url, $data=array(), $header=array(), $timeout=30){
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false); // 跳过证书检查
    curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, true);  // 从证书中检查SSL加密算法是否存在
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_HTTPHEADER, $header);
    curl_setopt($ch, CURLOPT_POST, true);
    curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($data));
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_TIMEOUT, $timeout);

    $response = curl_exec($ch);

    if($error=curl_error($ch)){
        die($error);
    }

    curl_close($ch);

    return $response;

}

// 调用
$url = 'https://www.example.com/api/message.php';
$data = array('name'=>'fdipzone');
$header = array();

$response = curl_https($url, $data, $header, 5);

echo $response;
?>
```

希望本文所述对大家的php程序设计有所帮助。

自己写的php curl库实现整站克隆功能

有时候经常会用到一些在线手册，比如国内或国外的，有些是访问速度慢，有些是作者直接吧网站关闭了，有些是服务器总是宕机，所以还是全盘克隆到自己服务器比较爽。

库特点：

给定一初始连接，初始链接以下的层级所有文件会拷贝到本地。

多次克隆可以配置是否覆盖。

可以配置是否下载图片。

所有链接替换为相对链接，所以可以随便rewrite。

绝对不会出现文件覆盖等问题。

最NB的特点是，没有比这更NB的库了。
SVN：http://svn.phpdr.net/repos/ares/php/library/trunk/lib/CurlMulti/MyCurl/Clone.php

脚本之家下载：http://xiazai.jb51.net/201502/other/CurlMulti.rar

克隆结果展示(这个克隆操作几秒钟就完成了)：

克隆源网站：http://www.laruence.com/manual/

克隆结果：http://manual.phpdr.net/yaf/
Demo代码：

```
<?php
class Controller_Spider extends MyYaf_Controller_Base{
 function init(){
  parent::init();
  if(!$this->getRequest()->isCli()){
      Ares_Http::error403();
  }
  include 'CurlMulti/CurlMulti.php';
  include 'CurlMulti/MyCurl.php';
  include 'phpQuery.php';
 }
}
```

```php
<?php
class YafdocController extends Controller_Spider {
 function init() {
  parent::init ();
  include 'CurlMulti/MyCurl/Clone.php';
 }
 function indexAction() {
  $url = 'http://www.laruence.com/manual';
  $dir = Yaf_Application::app ()->getAppDirectory () . '/data/manual';
  $cacheDir = $this->getBaseDir () . '/cache/curl';
  if (! is_dir ( $cacheDir )) {
   mkdir ( $cacheDir );
  }
  $curl = new CurlMulti ();
  $curl->maxThread = 10;
  $curl->cache ['enable'] = true;
  $curl->cache ['enableDownload'] = true;
  $curl->cache ['dir'] = $cacheDir;
  $curl->cache ['compress'] = true;
  $clone = new MyCurl_Clone ( $curl, $url, $dir );
  $clone->overwrite = true;
  $clone->start ();
  return false;
 }
}
```

PHP CURL 内存泄露问题解决方法

phpcurl使用privoxy代理访问https://www.google.com/search?q=xxx

curl配置平淡无奇，长时间运行发现一个严重问题，内存泄露！不论用单线程和多线程都无法避免！是curl访问https站点的时候有bug！

内存泄露可以通过linux的top命令发现，使用php函数memory_get_usage()不会发现。
经过反复调试找到解决办法，curl配置添加如下几项解决问题：

```
[CURLOPT_HTTPPROXYTUNNEL] = true;

[CURLOPT_SSL_VERIFYPEER] = false;

[CURLOPT_SSL_VERIFYHOST] = false;
```

CURLOPT_HTTPPROXYTUNNEL具体说明stackoverflow上有，直接贴原文：

Without CURLOPT_HTTPPROXYTUNNEL

Without CURLOPT_HTTPPROXYTUNNEL : You just use the proxy address/port as a destination of your HTTP request. The proxy will read the HTTP headers of your query, forward your request to the destination (with your HTTP headers) and then write the response to you.

Example steps :

1）HTTP GET /index.html sent to 1.1.1.1 (proxy)

2）1.1.1.1 receive request and parse header for getting the final destination of your HTTP request.

3）1.1.1.1 forward your query and headers to www.site.com (destination in request headers).

4）1.1.1.1 write back to you the response receive from www.site.com
With CURLOPT_HTTPPROXYTUNNEL

With CURLOPT_HTTPPROXYTUNNEL : You ask the proxy to open a direct binary connection (like HTTPS, called a TCP Tunnel) directly to your destination by doing a CONNECT HTTP request. When the tunnel is ok, the proxy write you back a HTTP/1.1 200 Connection established. When it received your browser start to query the destination directly : The proxy does not parse HTTP headers and theoretically does not read tunnel datas, it just forward it, thats why it is called a tunnel !

Example steps :

1）HTTP CONNECT sent to 1.1.1.1

2）1.1.1.1 receive HTTP CONNECT and get the ip/port of your final destination (header field of HTTP CONNECT).

3）1.1.1.1 open a TCP Socket by doing a TCP handshake to your destination 2.22.63.73:80 (ip/port of www.site.com).

4）1.1.1.1 Make a tunnel by piping your TCP Socket to the TCP Socket opened to 2.22.63.73:80and then write you back HTTP/1.1 200 Connection established witch means that your client can now make your query throw the TCP Tunnel (TCP datas received will be transmited directly to server and vice versa). http://stackoverflow.com/questions/12288956/what-is-the-curl-option-curlopt-httpproxytunnel-means

php使用curl出现Expect:100-continue解决方法

本文实例讲述了php使用curl出现Expect:100-continue解决方法。分享给大家供大家参考。具体如下：

使用curl POST数据时，如果POST的数据大于1024字节，curl并不会直接就发起POST请求。而是会分两步。

1.发送一个请求，header中包含一个Expect:100-continue，询问Server是否愿意接受数据。

2.接受到Server返回的100-continue回应后，才把数据POST到Server。
这个是libcurl定义的，具体可以查看相关描述：http://www.w3.org/Protocols/rfc2616/rfc2616-sec8.html#sec8.2.3

于是这样就会出现一个问题。并不是所有的Server都会回应100-continue的。例如lighttpd，会返回"417 Expectation Fail"，会造成逻辑错误。

解决方法如下，就是发送请求时，header中包含一个空的Expect。

```
curl_setopt($ch, CURLOPT_HTTPHEADER, array("Expect:"));
```

希望本文所述对大家的php程序设计有所帮助。

php运行出现Call to undefined function curl_init()的解决方法

在测试微信平台时，运行时出现了如标题的问题，没有定义的函数，也就是php还没打开对curl_init函数的支持，后来google了一下，解决方法如下：

系统环境，WIN2003 IIS6，PHP版本5.2.12
在装好PHP后，执行类似$ch = curl_init();这样的语句，出现Call to undefined function curl_init()的错误提示。解决方法如下：

1、在php.ini中找到extension=php_curl.dll，去掉前面的，php.ini一般在c:\windows下面。

2、在php.ini中找到extension_dir = “ext”，去掉前面的;，改为extension_dir = “C:\php5\ext”。

“C:\php5\ext”只是示例，即扩展指向的路径要对
3、php_curl.dll、libeay32.dll、ssleay32.dll、php5ts.dll都拷到system32下面去。

4、然后重启电脑，故障解决。

注意：在PHP的5.2.8版本中不知道什么原因，用这方法无法解决，换成了5.2.12才解决掉。

以windows下的php+apache为例。

首先，打开php.ini，找到“extension=php_curl.dll”，然后去掉前面的“;”注释，重启apache即可。
如果还出现此类问题，先检查php.ini的extension_dir值是哪个目录，在那个目录下检查有无php_curl.dll，没有的话请下载php_curl.dll，再把php目录中的libeay32.dll和ssleay32.dll拷到c:\windows\system32里面，重启apache，OK！

在Ubuntu 下运行php，总是提示Call to undefined function curl_init()，原因没有安转：php5-curl

与curl相关的内容见：http://packages.ubuntu.com/zh-cn/intrepid/php5-curl

CURL is a library for getting files from FTP, GOPHER, HTTP server.

PHP5 is an HTML-embedded scripting language. Much of its syntax is borrowed from C, Java and Perl with a couple of unique PHP-specific features thrown in. The goal of the lan guage is to allow web developers to write dinamically generated pages quickly. This version of PHP5 was built with the Suhosin patch.

H1>

(PHP 4 >= 4.0.2)

curl_init — 初始化一个CURL会话

描述

int curl_init ([string url])

curl_init()函数将初始化一个新的会话，返回一个CURL句柄供curl_setopt(), curl_exec(),和 curl_close() 函数使用。如果可选参数被提供，那么CURLOPT_URL选项将被设置成这个参数的值。你可以使用curl_setopt()函数人工设置。

例 1. 初始化一个新的CURL会话，且取回一个网页

```
<?php
$ch = curl_init();
curl_setopt ($ch, CURLOPT_URL, "http://www.zend.com/");
curl_setopt ($ch, CURLOPT_HEADER, 0);
curl_exec ($ch);
curl_close ($ch);
?>
```

php使用curl简单抓取远程url的方法

本文实例讲述了php使用curl抓取远程url的方法。分享给大家供大家参考。具体如下：

cURL是一个非常有用的php库，可以用来连接不通类型的服务器和协议，下面是一个最基本的范例用来抓取远程网页

```
<?php
$c = curl_init('http://www.w3mentor.com/robots.txt');
curl_setopt($c, CURLOPT_RETURNTRANSFER, true);
$page = curl_exec($c);
curl_close($c);
?>
```

希望本文所述对大家的php程序设计有所帮助。

php curl 上传文件代码实例

假设server端上传文件处理脚本upload.php：

```
<?php


print_r($_POST);

print_r($_FILES);
```

1、使用 CURL 默认的方法

```
//如果php文件是utf8编码，系统是GBK编码，那么就需要转下编码，要不然Php在系统中找不到这个文件

$file = realpath(mb_convert_encoding('测试图片.JPG','GBK','utf8'));


$file = realpath('temp.jpg'); //要上传的文件

$fields['f'] = '@'.$file; // 前面加@符表示上传图片


$ch =curl_init();




curl_setopt($ch,CURLOPT_URL,'http://localhost/upload.php');


curl_setopt($ch,CURLOPT_POST,true);

curl_setopt($ch, CURLOPT_POSTFIELDS, $fields);

curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);



$content = curl_exec($ch);


echo $content;
```

2、另类的做法，有时我们需要将动态产生的内容当做文件上传到远程服务器，却又不想在本地服务器中构建临时文件。这样就有了这个另类的写法

```
$contents =<<< 'TEXT'
这里是文件内容，也可以是图片二进制，图片需要修改上传文件类型
TEXT;

$varname = 'my';//上传到$_FILES数组中的 key
$name = '3.txt';//文件名
$type = 'text/plain';//文件类型

$key = "$varname\"; filename=\"$name\r\nContent-Type: $type\r\n";
$fields[$key] = $contents;



$ch =curl_init();



curl_setopt($ch,CURLOPT_URL,'http://localhost/upload.php');

curl_setopt($ch,CURLOPT_POST,true);
curl_setopt($ch, CURLOPT_POSTFIELDS, $fields);
curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);



$content = curl_exec($ch);

echo $content;
```

PHP curl伪造IP地址和header信息代码实例

curl虽然功能强大，但是只能伪造$_SERVER["HTTP_X_FORWARDED_FOR"]，对于大多数IP地址检测程序来说，$_SERVER["REMOTE_ADDR"]很难被伪造：

首先是client.php的代码

```
$headers['CLIENT-IP'] = '202.103.229.40';
$headers['X-FORWARDED-FOR'] = '202.103.229.40';

$headerArr = array();
foreach( $headers as $n => $v ) {
    $headerArr[] = $n .':' . $v;
}

ob_start();
$ch = curl_init();
curl_setopt ($ch, CURLOPT_URL, "http://localhost/curl/server.php");
curl_setopt ($ch, CURLOPT_HTTPHEADER , $headerArr );  //构造IP
curl_setopt ($ch, CURLOPT_REFERER, "http://www.163.com/ ");   //构造来路
curl_setopt( $ch, CURLOPT_HEADER, 1);

curl_exec($ch);
curl_close ($ch);
$out = ob_get_contents();
ob_clean();

echo $out;
```

然后是server.php

```
function GetIP(){
    if(!emptyempty($_SERVER["HTTP_CLIENT_IP"]))
        $cip = $_SERVER["HTTP_CLIENT_IP"];
    else if(!emptyempty($_SERVER["HTTP_X_FORWARDED_FOR"]))
        $cip = $_SERVER["HTTP_X_FORWARDED_FOR"];
    else if(!emptyempty($_SERVER["REMOTE_ADDR"]))
        $cip = $_SERVER["REMOTE_ADDR"];
    else
    $cip = "无法获取！";
    return $cip;
}
echo "<br>访问IP: ".GetIP()."<br>";
echo "<br>访问来路: ".$_SERVER["HTTP_REFERER"];
```

php curl 获取https请求的2种方法

今天一个同事反映，使用curl发起https请求的时候报错：“SSL certificate problem, verify that the CA cert is OK. Details: error:14090086:SSL routines:SSL3_GET_SERVER_CERTIFICATE:certificate verify failed”

很明显，验证证书的时候出现了问题。

使用curl如果想发起的https请求正常的话有2种做法：

方法一、设定为不验证证书和host。

在执行curl_exec()之前。设置option

```
$ch = curl_init();
.....

curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);

curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, FALSE);
```

方法二、设定一个正确的证书。

本地ssl判别证书太旧，导致链接报错ssl证书不正确。

我们需要下载新的ssl 本地判别文件

http://curl.haxx.se/ca/cacert.pem

放到 程序文件目录

curl 增加下面的配置

```
curl_setopt($ch,CURLOPT_SSL_VERIFYPEER,true); ;

  curl_setopt($ch,CURLOPT_CAINFO,dirname(__FILE__).'/cacert.pem');
```

大功告成
（本人验证未通过。。。报错信息为：SSL certificate problem, verify that the CA cert is OK. Details: error:14090086:SSL routines:SSL3_GET_SERVER_CERTIFICATE:certificate verify failed）

如果对此感兴趣的话可以参看国外一大神文章。http://unitstep.net/blog/2009/05/05/using-curl-in-php-to-access-https-ssltls-protected-sites/

php curl请求信息和返回信息设置代码实例

在用curl抓取网页内容的时候，经常要知道，网页返回的请求头信息，和请求的相关信息，特别是在请求过程中存在重定向的时候获取请求返回头信息对分析请求内容很有帮助

下面就是一个请求中存在重定向的例子，我们的目的是要获取最终实际请求的url地址

```
$url='http://www.appchina.com/market/r/489267/com.appshare.android.ilisten.vapk?c=aplus.direct&uid=gAJ9cQEu1TlyZxsXN-aB4RaanvFL6t6Bj-vj0rIBs&p=aplus.detail&m=redirect';

$ch=curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
//curl_setopt($ch, CURLOPT_POST, 1);
//curl_setopt($ch, CURLOPT_POSTFIELDS, $params);
curl_setopt($ch, CURLOPT_HEADER, 1);//返回response头部信息
curl_setopt($ch, CURLOPT_NOBODY, 1);//不返回response body内容
//curl_setopt($ch, CURLOPT_MAXREDIRS, 1);//设置请求最多重定向的次数
curl_setopt($ch,CURLOPT_RETURNTRANSFER,1);//不直接输出response
curl_setopt($ch, CURLOPT_FOLLOWLOCATION,1);//如果返回的response 头部中存在Location值，就会递归请求
$content=curl_exec($ch);
$rinfo=curl_getinfo($ch);

echo $content,"</br>";
echo "<hr>";
print_r($rinfo);
```

下面是输出的结果

```
HTTP/1.1 200 OKServer: nginxDate: Sat, 22 Dec 2012 06:17:44 GMTContent-Type: application/vnd.android.package-archiveConnection: closeLast-Modified: Mon, 03 Dec 2012 16:00:00 GMTExpire
s: Tue, 03 Dec 2013 16:00:00 GMTCache-Control: max-age=31536000Content-Length: 2142149
Array( [url] => http://www.d.appchina.com/McDonald/r/489267/com.appshare.android.ilisten.vapk?c=aplus.direct&uid=gAJ9cQEu1TlyZxsXN-aB4RaanvFL6t6Bj-vj0rIBs&p=aplus.detail&m=redirect [c
ontent_type] => application/vnd.android.package-archive [http_code] => 200 [header_size] => 289 [request_size] => 196 [filetime] => -1 [ssl_verify_result] => 0 [redirect_count] => 0 [total_time] =>
0.171621 [namelookup_time] => 0.135256 [connect_time] => 0.152913 [pretransfer_time] => 0.152916 [size_upload] => 0 [size_download] => 0 [speed_download] => 0 [speed_upload] => 0 [downlo
ad_content_length] => 2142149 [upload_content_length] => 0 [starttransfer_time] => 0.171582 [redirect_time] => 0 [certinfo] => Array ( ))
```

可以看到，经过递归请求后最终得到一个200的response,但是这中方式不能得到最后一次请求的url，也就是最终实际请求的url，要想得到这个url就需要递归的分析每次请求返回的response

下面是我写的一个获取最后一次请求url的递归函数

```
$url='http://www.appchina.com/market/r/489267/com.appshare.android.ilisten.vapk?c=aplus.direct&uid=gAJ9cQEu1TlyZxsXN-aB4RaanvFL6t6Bj-vj0rIBs&p=aplus.detail&m=redirect';
[php] view plaincopy
$realUrl=getRedirectLocation($url);

echo "</br>--->",$realUrl;

function getRedirectLocation($url){

  $realUrl=$url;
  echo $url,"</br>";
  $ch=curl_init();
  curl_setopt($ch, CURLOPT_URL, $url);
  curl_setopt($ch, CURLOPT_HEADER, 1);curl_setopt($ch, CURLOPT_TIMEOUT, 3);//设置curl执行时间不超过3秒
  //curl_setopt($ch, CURLOPT_NOBODY, 1);//这行不能要，如果添上，那么在遇到302重定向的时候就会得不到真正的请求url
  curl_setopt($ch,CURLOPT_RETURNTRANSFER,1);
  $content=curl_exec($ch);
  //echo $content;
  $rinfo=curl_getinfo($ch);
  $matches=array();
  if(preg_match('/Location:\s+?(.+?)\s+?/', $content,$matches)){
    //echo $matches[1],"</br>";
    unset($content);
    $realUrl=getRedirectLocation($matches[1]);
  }
  if(isset($content)){
    unset($content);
  }
  return $realUrl;
}
```

PHP Curl模拟登录微信公众平台、新浪微博实例代码

使用curl之前先打开curl配置，具体方式百度一下就知道，开启curl扩展。密码用md5加密，这是经过测试成功的，把用户跟密码改成你的就行了。

下面一段代码给大家介绍php使用curl模拟登录微信公众平台，具体代码如下所示：

```
<?php
//模拟微信登入
$cookie_file = tempnam('./temp','cookie');
$login_url = 'https://mp.weixin.qq.com/cgi-bin/login';
$pwd = md5("*********");
$data = "f=json&imgcode=&pwd=$pwd&username=*****@***.com";
$ch = curl_init($login_url);
curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
curl_setopt($ch,CURLOPT_POST,1);
curl_setopt($ch,CURLOPT_COOKIEJAR,$cookie_file);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
curl_setopt($ch,CURLOPT_SSL_VERIFYHOST,false);
curl_setopt($ch,CURLOPT_REFERER,'https://mp.weixin.qq.com');
curl_setopt($ch,CURLOPT_POSTFIELDS,$data);
$content = curl_exec($ch);
curl_close($ch);
$newurl = json_decode($content,1);
//var_dump($newurl);
//exit;
$newurl = $newurl['redirect_url'];
//获取登入后页面的源码
$go_url = 'https://mp.weixin.qq.com'.$newurl;
$ch = curl_init($go_url);
curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
curl_setopt($ch,CURLOPT_COOKIEFILE,$cookie_file);
curl_setopt($ch,CURLOPT_CONNECTTIMEOUT,0);
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
$content = curl_exec($ch);
//var_dump(curl_error($ch));
print_r($content);
curl_close($ch);
?>
```

使用 PHP CURL 模拟登录新浪微博

有时候我们获取一些新浪微博的数据,但又不想使用API,只好使用模拟登录了.

发现以前可以使用的CURL模拟登录代码失效了,Google一下,发现有很多人碰到这个问题.但是没有找到解决方法,所以就自己研究了一下,发现了原因.

可能是因为新浪限制了不允许模拟登录,同样的登录参数,用网页登录一切正常,用CURL登录,返回的COOKIES竟然是临时的.

所以看起来是登录成功了,并且获取到了用户信息,但是再次访问还是未登录状态.我的解决方法比较简单,直接修改COOKIES的时效这样就行了.

附上我自己测试通过的PHP代码如下,希望有对有同样问题的朋友有用,如果你有更好的方案欢迎分享一下.

发现只要不设置CURLOPT_COOKIESESSION参数就行了,不需要修改COOKIE_FILE.

```
<?php
class sina
{
/*
一个简单的新浪微搏curl模拟登录类. 来源: http://chenall.net/post/sina_curl_login/ 使用方法:
http函数是一个简单的curl封装函数,需要自己去实现,
http函数原型如下:
http($url,$post_data = null)
返回网页内容.
第一个参数$url,就是要访问的url地址,$post_data是post数据,如果为空,则代表GET访问.
1.使用加密后密码登录 加密方法: sha1(sha1($pass))
$sina = new sina($username,$sha1pass)
2.直接使用原始密码登录
$sina = new sina($username,$sha1pass,0)
执行之后如果$sina->status非空,则登录成功,否则登录失败.
登录成功之后,你就可以直接继续使用http函数来访问其它内容.
使用 unset($sina) 会自动注销登录.
*/
public $status;
function __construct($su,$sp,$flags = 1) {
$this->status = $this->login($su,$sp,$flags);
}
function __destruct()
{
//注销登录
$this->logout();
}
function logout()
{
http("http://weibo.com/logout.php");
unset($this->status);
}
/*不需要了,只要不设置HTTP函数中不设置CURLOPT_COOKIESESSION参数就行了,要设可以设为false.
function ResetCookie()//重置相关cookie
{
global $cookie_file;
$str = file_get_contents($cookie_file);
$t = time()+3600;//设置cookie有效时间一个小时
$str = preg_replace("/\t0\t/", "\t".$t."\t", $str);
$f = fopen($cookie_file,"w");
fwrite($f,$str);
fclose($f);
}
*/
function login($su,$sp,$flags = 0)
{
$su = urlencode(base64_encode($su));
$data = http("http://login.sina.com.cn/sso/prelogin.php?entry=miniblog&client=ssologin.js&user=".$su);
if (empty($data))
return null;
//$data = substr($data,35,-1);
$data = json_decode($data);
if ($data->retcode != 0)
return null;
if ($flags == 0)
$sp = sha1(sha1($sp));
$sp .= strval($data->servertime).$data->nonce;
$sp = sha1($sp);
$data = "url=http%3A%2F%2Fweibo.com%2Fajaxlogin.php%3F&returntype=META&ssosimplelogin=1&su=".$su.'&service=miniblog&servertime='.$data->servertime."&nonce=".$data->nonce.'&pwe
ncode=wsse&sp='.$sp;
$data = http("http://login.sina.com.cn/sso/login.php?client=ssologin.js",$data);
//$this->ResetCookie();
if (preg_match("/location\.replace\('(.*)'\)/",$data,$url))
{
$data = http($url[1]);
//$this->ResetCookie();
$data = json_decode(substr($data,1,-2));
if ($data->result == true)
return $data->userinfo;
}
return null;
}
}
?>
```

以上内容给大家介绍了PHP Curl模拟登录微信公众平台、新浪微博实例代码，希望本文所述对大家有所帮助。

您可能感兴趣的文章:

PHP CURL模拟登录新浪微博抓取页面内容 基于EaglePHP框架开发

php使用curl模拟登录后采集页面的例子

PHP CURL获取cookies模拟登录的方法

PHP中使用CURL模拟登录并获取数据实例

PHP使用CURL实现对带有验证码的网站进行模拟登录的方法

PHP读取CURL模拟登录时生成Cookie文件的方法

PHP使用CURL模拟登录的方法

PHP curl模拟登录带验证码的网站