

Forum: [Hack Forums](#)

Thread: [Asynchronous WinInet client wrapper](#)

Delirium™ - June 14, 2015, 7:28 pm

Hello.

I had a WinInet client wrapper which I often used in my projects. I modified it today to support both POST and GET methods along with sending your own data and all that done asynchronously.

So here's the code:

AsyncInet.cpp:

Code #include "AsyncInet.h"

AsyncInet::AsyncInet()

```
{
    this->hConnectEvent = CreateEvent(NULL, FALSE, FALSE, NULL);
    this->hRequestOpenEvent = CreateEvent(NULL, FALSE, FALSE, NULL);
    this->hRequestCompleteEvent = CreateEvent(NULL, FALSE, FALSE, NULL);

    this->hInetInstance = NULL;
    this->hInetConnect = NULL;
    this->hInetRequest = NULL;
}
```

AsyncInet::~~AsyncInet()

```
{
    if (this->hConnectEvent)
        CloseHandle(this->hConnectEvent);

    if (this->hRequestOpenEvent)
        CloseHandle(this->hRequestOpenEvent);

    if (this->hRequestCompleteEvent)
        CloseHandle(this->hRequestCompleteEvent);

    CloseConnection();
}
```

BOOL AsyncInet::Connect(

```
    string szAddr,
    unsigned short port,
    string szAgent,
    DWORD dwTimeout)
{
    CloseConnection();
```

```
    ResetEvent(this->hConnectEvent);
    ResetEvent(this->hRequestOpenEvent);
```

```

ResetEvent(this->hRequestCompleteEvent);

if (!(this->hInetInstance =
    InternetOpenA(szAgent.c_str(),
        INTERNET_OPEN_TYPE_PRECONFIG, NULL, NULL, INTERNET_FLAG_ASYNC)))
{
    return FALSE;
}

if (InternetSetStatusCallbackA(this->hInetInstance,
    (INTERNET_STATUS_CALLBACK)&CallbackFunction))
{
    return FALSE;
}

this->context.dwContext = AsyncInet::CONTEXT_CONNECT;
this->context.obj = this;

this->hInetConnect = InternetConnectA(this->hInetInstance, szAddr.c_str(),
    port, NULL, NULL,
    INTERNET_SERVICE_HTTP, INTERNET_FLAG_KEEP_CONNECTION |
    INTERNET_FLAG_NO_CACHE_WRITE, reinterpret_cast<DWORD>(&this->context));

if (this->hInetConnect == NULL)
{
    if (GetLastError() == ERROR_IO_PENDING)
    {
        if (WaitForSingleObject(this->hConnectEvent, dwTimeout))
            return FALSE;
    }
    return FALSE;
}

if (this->hInetConnect == NULL)
{
    return FALSE;
}

return TRUE;
}

void AsyncInet::CloseConnection()
{
    if (this->hInetInstance)
        InternetCloseHandle(this->hInetInstance);

    if (this->hInetConnect)
        InternetCloseHandle(this->hInetConnect);

    if (this->hInetRequest)
        InternetCloseHandle(this->hInetRequest);

    this->hInetInstance = NULL;
}

```

```

this->hInetConnect = NULL;
this->hInetRequest = NULL;

}

void WINAPI AsyncInet::CallbackFunction(
    HINTERNET hInternet,
    DWORD dwContext,
    DWORD dwInetStatus,
    LPVOID lpStatusInfo,
    DWORD dwStatusInfoLength)
{
    InetContext *ctx = reinterpret_cast<InetContext*>(dwContext);

    switch (ctx->dwContext)
    {
    case AsyncInet::CONTEXT_CONNECT:
        if (dwInetStatus == INTERNET_STATUS_HANDLE_CREATED)
        {
            INTERNET_ASYNC_RESULT *inetResult =
reinterpret_cast<INTERNET_ASYNC_RESULT*>(lpStatusInfo);
            ctx->obj->hInetConnect = reinterpret_cast<HINTERNET>(inetResult->dwResult);
            SetEvent(ctx->obj->hConnectEvent);
        }
        break;
    case AsyncInet::CONTEXT_REQUESTHANDLE:
        switch (dwInetStatus)
        {
        case INTERNET_STATUS_HANDLE_CREATED:
            {
                INTERNET_ASYNC_RESULT *inetResult1 =
reinterpret_cast<INTERNET_ASYNC_RESULT*>(lpStatusInfo);
                ctx->obj->hInetRequest = reinterpret_cast<HINTERNET>(inetResult1->dwResult);
                SetEvent(ctx->obj->hRequestOpenEvent);
            }
            break;
        case INTERNET_STATUS_REQUEST_COMPLETE:
            SetEvent(ctx->obj->hRequestCompleteEvent);
            break;
        }
        break;
    }
}
}

```

```

BOOL AsyncInet::SendRequest(string szURL, RequestType requestType, string szRequestData,
string szReferrer, DWORD dwTimeout)
{

```

```

    this->context.dwContext = AsyncInet::CONTEXT_REQUESTHANDLE;
    this->context.obj = this;

    string szVerb;

```

```

if (requestType == RequestType::GET)
    szVerb = "GET";
else
    szVerb = "POST";

this->hInetRequest = HttpOpenRequestA(this->hInetConnect, szVerb.c_str(), szURL.c_str(),
    NULL, szReferrer.c_str(), NULL,
    INTERNET_FLAG_RELOAD | INTERNET_FLAG_KEEP_CONNECTION
    | INTERNET_FLAG_NO_CACHE_WRITE | INTERNET_FLAG_FORMS_SUBMIT
    | INTERNET_FLAG_IGNORE_REDIRECT_TO_HTTPS
    | INTERNET_FLAG_IGNORE_REDIRECT_TO_HTTP,
reinterpret_cast<DWORD>(&this->context));

if (this->hInetRequest == NULL)
{
    cerr << GetLastError() << endl;
    if (GetLastError() == ERROR_IO_PENDING)
    {
        if (WaitForSingleObject(this->hRequestOpenEvent, dwTimeout))
            return FALSE;
    }
}

if (this->hInetRequest == NULL)
    return FALSE;

BOOL requestFlag;

if (requestType == RequestType::GET)
{
    requestFlag = HttpSendRequestA(this->hInetRequest, NULL, 0,
const_cast<char*>(szRequestData.c_str()), szRequestData.size());
}
else {
    string szHeaders = "Content-Type: application/x-www-form-urlencoded";
    requestFlag = HttpSendRequestA(this->hInetRequest, szHeaders.c_str(), szHeaders.size(),
const_cast<char*>(szRequestData.c_str()), szRequestData.size());
}

if (!requestFlag)
{
    if (GetLastError() == ERROR_IO_PENDING)
    {
        if (WaitForSingleObject(this->hRequestCompleteEvent, dwTimeout) == WAIT_TIMEOUT)
        {
            CloseConnection();
            return FALSE;
        }
    }
}
return TRUE;
}

```

```

unsigned long AsyncInet::ReadData(PBYTE lpBuffer, DWORD dwSize, DWORD dwTimeout)
{
    INTERNET_BUFFERS inetBuffers;
    memset(&inetBuffers, 0, sizeof(inetBuffers));

    inetBuffers.dwStructSize = sizeof(inetBuffers);
    inetBuffers.lpvBuffer = lpBuffer;
    inetBuffers.dwBufferLength = dwSize - 1;

    this->context.dwContext = AsyncInet::CONTEXT_REQUESTHANDLE;
    this->context.obj = this;

    if (!InternetReadFileEx(this->hInetRequest, &inetBuffers, 0,
        reinterpret_cast<DWORD>(&this->context)))
    {
        if (GetLastError() == ERROR_IO_PENDING)
        {
            if (WaitForSingleObject(this->hRequestCompleteEvent, dwTimeout) == WAIT_TIMEOUT)
            {
                return FALSE;
            }
        }
    }

    return inetBuffers.dwBufferLength;
}

```

AsyncInet.h:

Code #pragma once

```

#include <stdio.h>
#include <tchar.h>
#include <iostream>
#include <Windows.h>
#include <WinInet.h>

```

```

#include "AsyncInet.h"

```

```

#pragma comment (lib, "Wininet.lib")

```

```

using namespace std;

```

```

class AsyncInet;

```

```

typedef struct _InetContext
{
    AsyncInet *obj;
    DWORD dwContext;
} InetContext;

```

```

typedef enum _RequestType {
    GET,
    POST
} RequestType;

```

```

class AsyncInet
{
public:
    AsyncInet();
    ~AsyncInet();

    enum {
        CONTEXT_CONNECT,
        CONTEXT_REQUESTHANDLE
    };

    BOOL Connect(
        string szAddr,
        unsigned short port = INTERNET_DEFAULT_HTTP_PORT,
        string szAgent = "AsyncInet",
        DWORD dwTimeout = 20 * 1000);

    BOOL SendRequest(
        string szURL,
        RequestType requestType,
        string szRequestData,
        string szReferrer,
        DWORD dwTimeout = 20 * 1000);

    unsigned long ReadData(
        PBYTE lpBuffer,
        DWORD dwSize,
        DWORD dwTimeout = 20 * 1000);

    void CloseConnection();

    static void WINAPI CallbackFunction(
        HINTERNET hInternet,
        DWORD dwContext,
        DWORD dwInetStatus,
        LPVOID lpStatusInfo,
        DWORD dwStatusInfoLength);

protected:
    InetContext context;
    HANDLE hConnectEvent, hRequestOpenEvent, hRequestCompleteEvent;
    HINTERNET hInetInstance, hInetConnect, hInetRequest;

};

Usage example for a GET request:
Code AsyncInet inet;
if (inet.Connect("example.com"))
{
    if (inet.SendRequest("test.php?id=5", RequestType::GET, "", ""))
    {
        string szResponse;
        char szBuf[1024];
    }
}

```

```

    int iLength;

    while ((iLength = inet.ReadData(reinterpret_cast<PBYTE>(szBuf), 1024)) > 0)
    {

        szBuf[iLength] = 0;
        szResponse += szBuf;
    }

    cout << szResponse.c_str() << endl;
}
}Usage example for a POST request:
Code AsyncInet inet;
if (inet.Connect("example.com"))
{
    if (inet.SendRequest("test.php", RequestType::POST,
"somedata=somevalue&anotherone=anothervalue", ""))
    {
        string szResponse;
        char szBuf[1024];
        int iLength;

        while ((iLength = inet.ReadData(reinterpret_cast<PBYTE>(szBuf), 1024)) > 0)
        {

            szBuf[iLength] = 0;
            szResponse += szBuf;
        }

        cout << szResponse.c_str() << endl;
    }
}
}

```

I do not remember where I found the original class before my own modifications therefore I cannot credit anyone at the time being. It's been a long time.

If your project includes a precompiled header (Visual Studio users mainly), add the following line before `#include "AsyncInet.h"` in `AsyncInet.cpp`. Also make sure that you've got no double includes or any conflict might arise.

Code `#include "stdafx.h"`
Enjoy ?

Only3 - June 14, 2015, 9:09 pm

will help for ppl that need async request

maybe <http://veridium.net/programming-tutorials/http-client-async-wininet>

D3M0T3P - June 15, 2015, 1:35 am

don't use "using namespace std;" in your header

thanks ;-) for sharing

Delirium™ - June 15, 2015, 3:12 am

(Jun 14, 2015 09:09 PM)Only3 Wrote: [\[CITE\]](#) will help for ppl that need async request

maybe <http://veridium.net/programming-tutorials/http-client-async-wininet>

This is what my code provides. Async HTTP requests.

(Jun 15, 2015 01:35 AM)D3M0T3P Wrote: [\[CITE\]](#) don't use "using namespace std;" in your header thanks ;-)

for sharing
Why? It saves me time from writing string blabla instead of std::string blabla.

Benoit balls - June 16, 2015, 1:01 pm

(Jun 15, 2015 03:12 AM)Delirium™ Wrote: [\[CITE\]](#) This is what my code provides. Async HTTP requests.

Why? It saves me time from writing string blabla instead of std::string blabla.

It is considered a bad coding practice , beats the whole purpose of the namespace.

FSFHDFY - June 16, 2015, 1:09 pm

Don't even lecture this kid on bad practice, this code is an abomination of cluster fucks.

Delirium™ - June 16, 2015, 4:06 pm

(Jun 16, 2015 01:01 PM)KeDDy Wrote: [\[CITE\]](#) It is considered a bad coding practice , beats the whole purpose of the namespace.

I'm using it mainly because it's a habit I had since I was learning C++. I know it's bad practice because it can cause conflicts and prevent a reader from understanding where an identifier originates. The only reason I asked you why is because, any advice is merely of no use if you don't explain it, right? ?

(Jun 16, 2015 01:09 PM)FSFHDFY Wrote: [\[CITE\]](#) Don't even lecture this kid on bad practice, this code is an abomination of cluster fucks.

Are you probably mad? Because people getting mad of the internet are the kids.

0xDEAD10CC - June 16, 2015, 8:55 pm

Header files should NEVER have 'using namespace std;' ever... Do you know how header files are included? You're just opening up a can of worms for any other programmer wanting to use your code when other conflicts may possibly arise. Having that in a header file is the worst thing you can do.

Benoit balls - June 17, 2015, 2:28 am

(Jun 16, 2015 04:06 PM)Delirium™ Wrote: [\[CITE\]](#)

Yeah of course, thanks I might use it in the future.

Delirium™ - June 17, 2015, 8:56 am

(Jun 16, 2015 08:55 PM)0xDEAD10CC Wrote: [\[CITE\]](#) Header files should NEVER have 'using namespace std;' ever... Do you know how header files are included? You're just opening up a can of worms for any other programmer wanting to use your code when other conflicts may possibly arise. Having that in a header file is the worst thing you can do.

As I mentioned before, I know why it's bad practice but it's a bad habit I have left since the days I've been learning C++. I sometimes forget it :P

I will edit the source later to fix this.

0xDEAD10CC - June 17, 2015, 11:01 am

(Jun 17, 2015 08:56 AM)Delirium™ Wrote: [\[CITE\]](#) As I mentioned before, I know why it's bad practice but it's a bad habit I have left since the days I've been learning C++. I sometimes forget it :P

I will edit the source later to fix this.

I'm not talking about 'using namespace std;' in general. Whether you like doing that or not, it should never ever be used in a header file. Source files ONLY, if you must, but never header files. I don't think you understand what I was getting at. People above only mentioned that 'using namespace std;' is bad practice... I was talking about it being worse that it's in a header file.

Delirium™ - June 17, 2015, 11:49 am

(Jun 17, 2015 11:01 AM)0xDEAD10CC Wrote: [\[CITE\]](#) I'm not talking about 'using namespace std;' in general. Whether you like doing that or not, it should never ever be used in a header file. Source files ONLY, if you must, but never header files. I don't think you understand what I was getting at. People above only mentioned that 'using namespace std;' is bad practice... I was talking about it being worse that it's in a header file.

Indeed, I missed that point. Thank you for notifying me, something new I learned today I guess :)