
Subject: Re: error / memory leak in HttpServer example
Posted by [BetoValle](#) on Tue, 13 Jun 2023 14:11:51 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi,
I made some changes and it worked by terminating from Ctrl+c (or via uri)
(windows 10!)
Tks!

```
#include <Core/Core.h>
#include <signal.h>
using namespace Upp;

TcpSocket server;
StaticMutex ServerMutex;

void signal_callback_handler(int signum)
{
    Cout() << "Caught signal " << signum << EOL;
    // Terminate program
    exit(signum);
}

void Server()
{
    bool stop = false; // Of course it is not recommended to manipulate the server through
                       // the client! This condition is for experimental testing purposes!
    for(;;) {

        TcpSocket socket;
        LOG("Waiting...");
        ServerMutex.Enter();

        bool b = socket.Accept(server);
        ServerMutex.Leave();

        if((b) && stop == false) {
            LOG("Connection accepted");
            HttpHeader http;
            http.Read(socket);
            String html;
            html << "<html>"
                << "<b>Method:</b> " << http.GetMethod() << "<br>"
                << "<b>URI:</b> " << http.GetURI() << "<br>";
            stop = (http.GetURI().Find("stop=ok") > -1);
            if(stop)
                html << "<p>atenção! vai encerrar o servidor!</p>"
```

```

    << "<br>";
    for(int i = 0; i < http.fields.GetCount(); i++)
        html << "<b>" << http.fields.GetKey(i) << ":</b> " << http.fields[i] << "<br>";
    int len = (int)http.GetContentLength();
    if(len > 0)
        socket.GetAll(len);
    html << "<b><i>Current time:</i></b> " << GetSysTime() << "</html>";
    HttpResponse(socket, http.scgi, 200, "OK", "text/html", html);
    if(stop) {
        socket.Close();
        server.Close();
        exit(0);
    }
    signal(SIGINT, signal_callback_handler);
}
}
}

```

CONSOLE_APP_MAIN

```

{

    StdLogSetup(LOG_COUT | LOG_FILE);

    if(!server.Listen(4000, 10)) {
        LOG("Cannot open server port for listening\r\n");
        return;
    }
#ifdef _MULTITHREADED
    const int NTHREADS = 10;
    for(int i = 0; i < NTHREADS; i++)
        Thread::Start(callback(Server));
#endif
    Server();
}

```