



maXbox Starter 6

Start with Network Programming

1.1 Build a Blog

Today we spend another time in programming with the internet (called social network) and FTP. Hope you did already work with the Starter 1 to 5 available at:

<http://www.softwareschule.ch/maxbox.htm>

This lesson will introduce you to FTP and HTTP. By the way do you know what's a Blog is? Blogs are usually maintained by an individual with regular entries of commentary, descriptions of events, or other material such as graphics or video. In the following lines we're going to build a blog script.

Let's begin with HTTP (Hypertext Transfer Protocol) and TCP. TCP/IP stands for Transmission Control Protocol and Internet Protocol. TCP/IP can mean many things, but in most cases, it refers to the network protocol itself.

Each computer on a TCP/IP network has a unique address associated with it, the IP-Address. Some computers may have more than one address associated with them. An IP address is a 32-bit number and is usually represented in a dot notation, e.g. 192.168.0.1. Each section represents one byte of the 32-bit address. Our connection with HTTP represents an object.

👉 When HTTP is used on the Internet, browsers like Firefox act as clients and the application that is hosting the website like [softwareschule.ch](http://www.softwareschule.ch) acts as the server.

What we want to do next is to download the last blog entries, write a new entry and upload it immediately with the help of FTP. File Transfer Protocol is a standard network protocol used to copy a file from one host to another over a TCP/IP-based network, such as the Internet. In our case we never write a file to the local disk we just upload a stream of a stream object.

1.2 Code it

As you already know the tool is split up into the toolbar across the top, the editor or code part in the centre and the output window at the bottom.

🔗 Before this starter code will work you will need to download maXbox from the website. It can be downloaded from <http://sourceforge.net/projects/maxbox> site. Once the download has finished, unzip the file, making sure that you preserve the folder structure as it is. If you double-click maxbox3.exe the box opens a default program. Test it with F9 or press **Compile** and you should hear a sound. So far so good now we'll open the example:

129_pas_blogger.txt

If you can't find the file use the link:

http://www.softwareschule.ch/examples/129_pas_blogger.txt

Or you use the Save Page as... function of your browser¹ and load it from examples (or wherever you stored it). Now let's take a look at the code of this project. Our first line is

```
11 program Blogger_MAX;
```

We have to name the program it's called Blogger_MAX;.

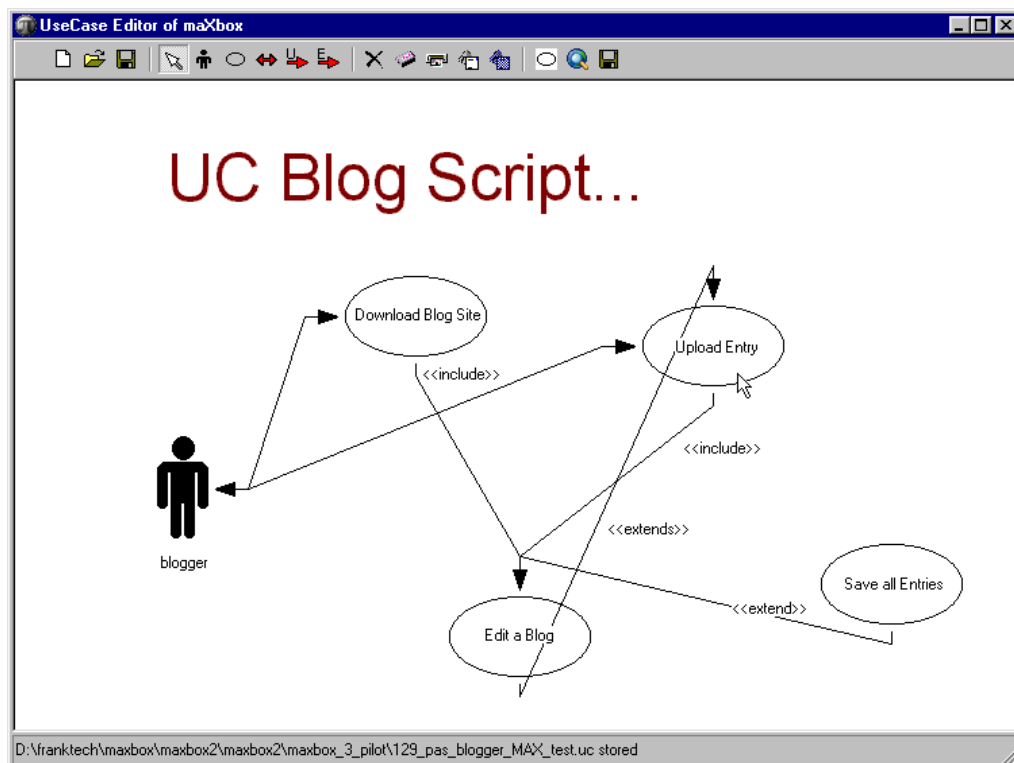
```
13 const
14   HTM_FILE = 'maxboxblog.htm';
15   CONTENT_URL = 'http://www.softwareschule.ch/'+HTM_FILE;
16   BLOGLINE =
17     //'Just got the IdHTTP.Request.CacheControl test with success ID:9';
18     'Last preparation with the tutorial called: how to build a blog in 100
```

First in line 14 we define the blog file, initially a local file you want to upload for the first time on a web server. If you don't have a provider with your own web server I propose to use a local web server to test (see last page link). In line 15 you find the IP address of the web server and line 16 is indeed the blogline with your entries you want to publish.

☞ This example requires four objects of the classes: TIdHTTP, TIdFTP, TMemoryStream and TStringList and the string-list passes your entries with the help of a stream to the blog site server. The app makes first a connection with the Get2 method by passing a website address.

```
32   vcontList.text:= idHTTP.Get2(vcontentURL);
```

So the object idHTTP has a method named Get2() you find in the IdHTTP.pas unit or library. A library is a collection of code or classes, which is included in your program.



1: The UC of the App

In line 32 you recognize the host name (IP address) as a parameter of the GET2 method.


¹ Or copy & paste

Also the no-cache property is set in line 30. When the HTML code is parsed, it is read from top to bottom. When the `<HTTP-EQUIV="PRAGMA" CONTENT="NO-CACHE">` meta tag is read, a browser looks for the existence of the page in cache at that exact moment. If it is there, it is removed.

```
30     idHTTP.Request.Pragma:= 'no-cache';
31     IdHTTP.Request.CacheControl:= 'no-cache';
```

But one way that can happen is if the HTTP request is passing through a proxy server. `TIdHTTP` does not perform any caching of its own. That's the reason I made two lines with proxy code in case you'll need that.

Indy is also very useful for including file or memory streams in your call to upload the stream you write your nice blog line in line 63, more of this later on. First I want to explain what a stream is.

 **Streams** are classes that let you read and write data. They provide a common interface for reading and writing to different media such as memory, strings, sockets, and BLOB fields.

There are several stream classes, which all descend from `TStream`. Each stream class is specific to one media type. For our example, `TMemoryStream` reads from or writes to a memory image without store a file on the disk.

```
62     myftp.ChangeDir('httpdocs')
63     myftp.Put1(ftpUpStream, myFile, false);
64     Writeln('Upload Size :'+inttoStr(myftp.size(myfile)));
```

The `Put1` method of the `myftp` object sends the stream up to the server. The method has the following interface and his parameters:

```
Procedure Put1(const ASource: TStream; const ADestFile:
               string; const AAppend: boolean);
```

`TMemoryStream` descends from `TStream` (in unit `Classes`), inheriting most of its members. `ADestfile` is the file name of the blog-site, you remember (`HTM_FILE = 'maxboxblog.htm'`).

The `Stream` object is created in line 44, the `FTP` object next line and the important thing has to be done in line 52 ff:

```
52     with ftpUpStream do begin
53         Seek(0, soFromBeginning);
54         Write(mysource, length(mysource))
55         SetLength(mysource, ftpupstream.Size);
56         //saveToFile(ExePath+'docs/'+myFile);
57     end;
```

The `Write` method writes `Count` bytes from a buffer to the stream, starting at the current `Position`. The prototype (interface) for `Write` is:

```
function Write(const Buffer; Count: Longint): Longint;
```

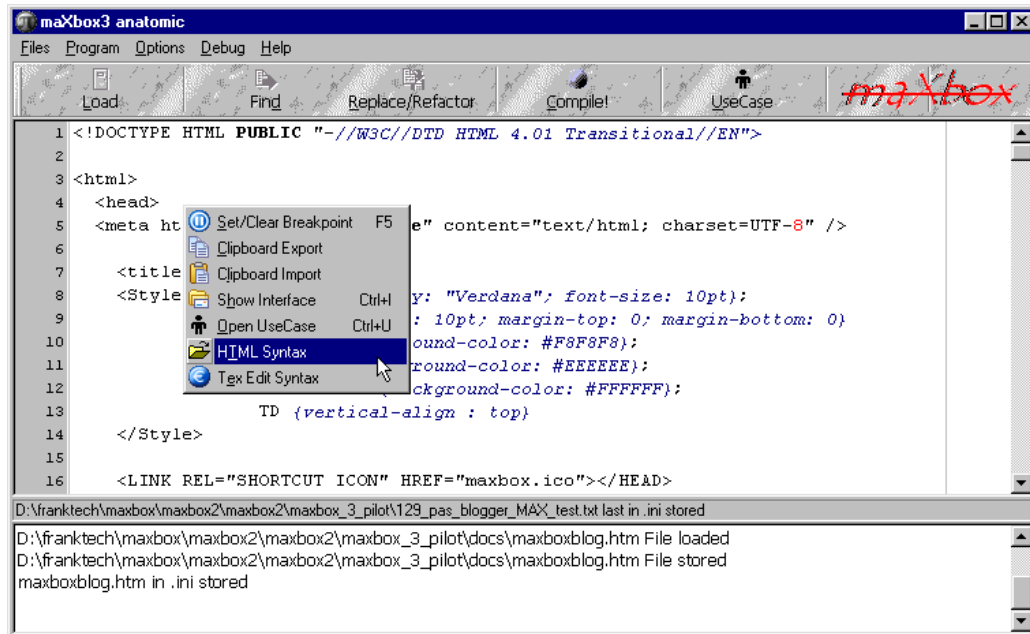
After writing to the file, `Write` advances the current position by the number bytes written, and returns the number of bytes actually written, which may be less than `Count` if the end of the buffer is encountered or the stream can't accept any more bytes. With `Seek()` in line 53 we move to a specified position in the streamed resource, in our case from the beginning.



So far we have learned something about HTTP, FTP and Streams. Now it's time to set our FTP settings before we run the app at first with F9 (if you haven't done yet) and learn something about HTML.

The acronym HTML stands for Hyper Text Markup Language - the primary programming language used to write content on the web. One of a practical way to learn more about actually writing HTML is

to get in the maXbox editor and load or open a web-file with the extension htm or html. Or you copy the output and paste it in a new maXbox instance. Then you click on the right mouse pad (context menu) and change to HTML Syntax!



2: Editor in HTML Mode

Let's back to the **FTP Account**. In order for the users to be able to connect to the respective host server through an FTP client, they need to have rights to access that server. These authorization access rights are assigned to users by their hosting providers in the form of FTP accounts. Each FTP account consists of unique username and password granting users access to the files of a given website on a certain FTP server and consists following lines:

```
46  try
47      with myFTP do begin
48          Host:= 'www.softwareschule.ch' //www.yourftpserveraddress.ch
49          Username:= 'YourFtpUsername';
50          Password:= 'YourFtpPassword';
```

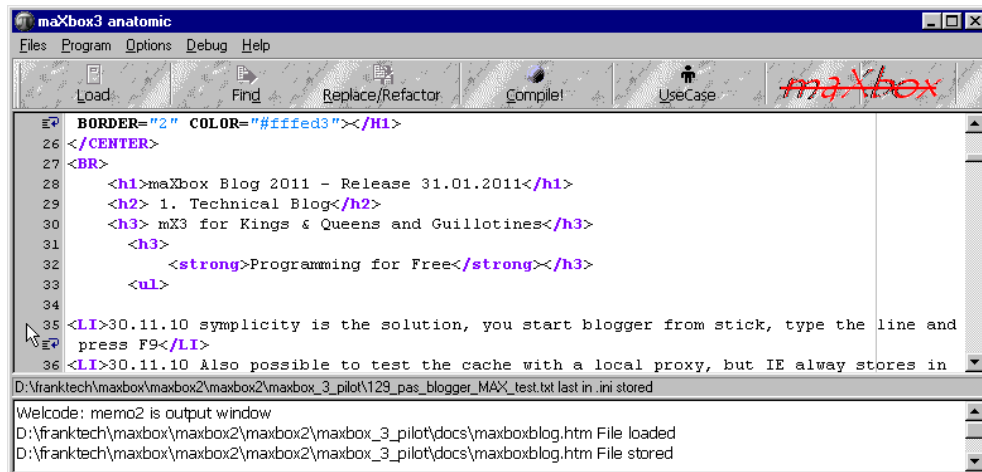
👉 The ability to connect to a host server through an FTP client is given by the authorization access rights in the form of FTP accounts.

Object myFTP is a dynamically allocated block of memory whose structure is determined by its class type. Each object has a unique copy of every field defined in the class, but all instances of a class share the same methods. Next we finally step to the main lines.

```
74  begin
75      //initialization
76      contentLst:= TStringList.create;
77      try
78          LetHTTPConnect(CONTENT_URL, contentLst);
79          contentLst.insert(35, '<LI>'+dateToStr(Date)+' '+BLOGLINE+'</LI>')
80          //contentlst.delete(35);
81          Writeln(contentlst.text)
82          FTP_UploadStream(contentlst.text, HTM_FILE)
```

TStringList is created to manage the content we write and insert. (Yes, this is not a Content Management System ;). In line 78 we connect to the web site with blog content. Now we insert in next line our content we have edit before in line 17 in the script, so we pass this to the method insert. The parameter 35 needs an explanation because this is a defined line in our htm file to be inserted always on the same place.

Take a look at the next picture and you'll get the dependence between the line 35 in html code² and the insert command in the script code! Call Insert to add the string (our blog line) to the list at the position specified by Index 35. If Index is 0, the string is inserted at the beginning of the list. If Index is 1, the string is put in the second position of the list, and so on. And the good thing is that each line moves one line further down, so the most new is always at the top of it.



After in line 81 we show the new content with `Writeln` in the output window below, we pass in line 82 to the `FTP_UploadStream` method our content with the information of the file name.



You can always delete a line in the blog with a delete method of the string-list. Main conclusion:

- 1 Construct a string-list object.
- 2 Connect to a web site and fill the string-list with the content of the web site
- 3 Change the content with inserting a blog line and show the new content to check
- 4 Send (upload) the content with the help of FTP to the web site
- 5 In the finally part, free the string-list object.



Try to upload an entry with link information in the blogline:

```
BLOGLINE =
  'test with a link <a href="http://www.softwareschule.ch/download/maxblog.txt">a
    simple text logger</a> also Linux (January 2011).';
```

Some notes at last about firewalls or proxy-servers. It depends on your network infrastructure to get a file or not, maybe you can't download content cause of security reasons and it stops with Socket-Error # 10060 and a time out error. Hope you did learn in this tutorial more of the internet.

max@kleiner.com

Links of maXbox and a simple Web Server:

<http://www.softwareschule.ch/maxbox.htm>
<http://www.softwareschule.ch/download/httpsserver2.zip>
<http://sourceforge.net/projects/maxbox>
<http://sourceforge.net/apps/mediawiki/maxbox/>

² Marked as the last blog entry