

maXbox



maXbox Starter 5

Start with Internet Programming

1.1 First Step of Indy

Today we spend some time in programming with the internet and their protocols too. Hope you did already work with the Starter 1 and 2 or 3 and 4 at:

http://www.softwareschule.ch/download/maxbox_starter.pdf

This lesson will introduce you to Indy Sockets and the library in a separate way. So what are the Indy Sockets?

Indy.Sockets (VCL) is an open source socket library that supports clients, servers, TCP, UDP, raw sockets, as well as over 100 higher level protocols such as SMTP, POP3, FTP, HTTP, and many more. Indy.Sockets is written in Delphi but is available for C#, C++, Delphi any .net language, and Kylix (CLX) too.

In our case we show two examples of HTTP (and a third one at your own service).

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 so called 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. In maXbox a 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 acts as the server.

1.2 Get the Code

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.

☞ In maXbox you treat download / upload to get code in a file also with objects.

📄 Before this starter code will work you will need to download maXbox from the website. It can be downloaded from <http://www.softwareschule.ch/maxbox.htm> (you'll find the download maxbox3.zip on the top left of the page). 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 demo program. Test it with F9 or press **Compile** and you should hear a sound. So far so good now we'll open the first of our two examples.

101_pas_http_tester.txt
102_pas_http_download.txt

If you can't find the two files try also the zip-file loaded from:
http://www.softwareschule.ch/download/maxbox_internet.zip

In future days you can download your very file with the box itself ;), because you connect like the following way in ex. 102. 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

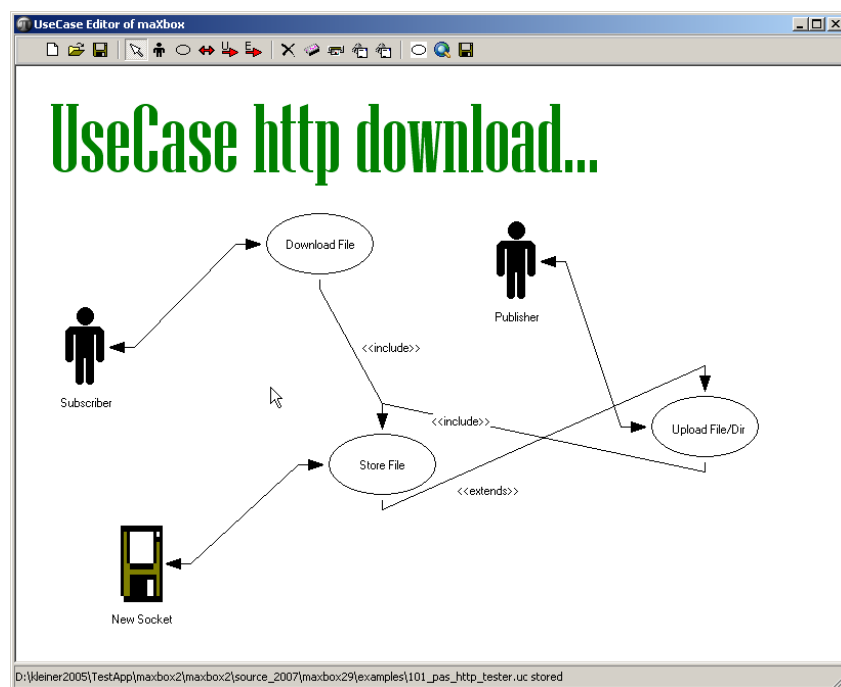
```
01 program motion_HTTPTester;
```

We have to name the game, means the program's name is `motion_HTTPTester`;

☞ This example requires two objects from the classes: `TIdHTTP` and `TIdHTTPRequest` but the second one is just for testing of the type. After creating the object in line 18 we use a first method calling `GET`. The program makes a connection with the `Get2` method by passing a website address.

```
19 IdHTTP.Get2('http://www.softwareschule.ch');
```

So the object `IdHTTP` has a method named `Get2()` you can find in the `IdHTTP.pas` unit or library. A library is a collection of code or classes, which you can include in your program. By storing your commonly used code in a library, you can reuse code for many times in different projects and also hide difficult sections of code from the developer; another advantage of modular programming. Once a unit is tested it's stable to use.



1: The UC of the Code


Indy is designed to provide a very high level of abstraction. Much more stuff or intricacies and details of the TCP/IP stack are hidden from the Indy programmer. A typical Indy client session looks like this:

```
with IndyClient do begin
  Host:= 'zip.pbe.com'; // Host to call
  Port:= 6000; // Port to call the server on
  Connect; // get something to with it
  Disconnect;
end;
```

¹ Or copy & paste

Indy is different than other so called winsock components you may be familiar with. If you've worked with other components, the best approach for you may be to forget how they work. Nearly all other components use non-blocking (asynchronous) calls and act asynchronously. They require you to respond to events, set up state machines, and often perform wait loops. Indy is also very useful for including file streams in your call to store your file you want to download, more of this in ex. 102.

Let's get back to ex. 101. In line 20 you see the host name as a parameter of the GET method.

 **Host names** are "human-readable" names for IP addresses. An example host name is max.kleiner.com, the www is just a convention and not necessary. Every host name has an equivalent IP address, e.g. www.hower.org = 207.65.96.71.

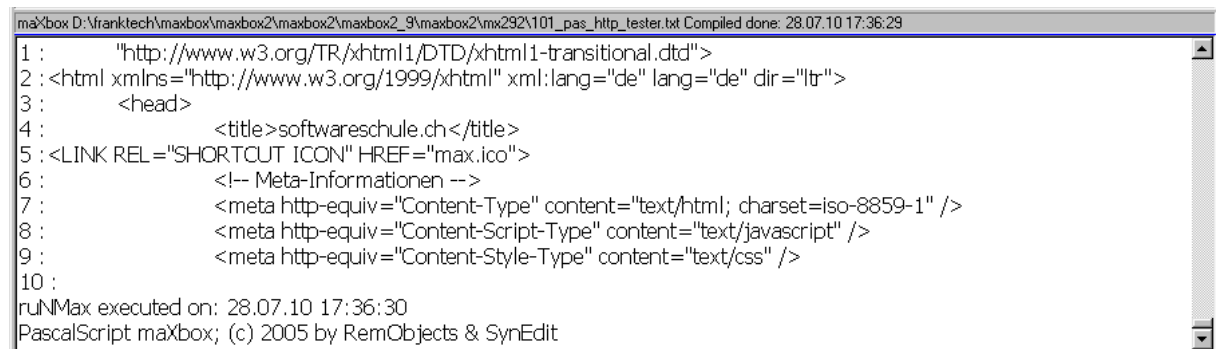
Host names are used both to make it easier on us humans, and to allow a computer to change its IP address without causing all of its potential clients (callers) to lose track of it.

Often called "URL or links" because the host address is normally inserted toward the top of the browser as one of the first items to look at for searching something.



So far we have learned something about HTTP and host names. Now it's time to run your program at first with F9 (if you haven't done yet) and learn something about HTML. The program generates an HTML output of the host name '<http://www.softwareschule.ch>' after downloading with GET.

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



```

maXbox D:\franktech\maXbox\maXbox2\maXbox2_9\maXbox2\mx292\101_pas_http_tester.txt Compiled done: 28.07.10 17:36:29
1 :      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2 : <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de" dir="ltr">
3 :      <head>
4 :          <title>softwareschule.ch</title>
5 : <LINK REL="SHORTCUT ICON" HREF="max.ico">
6 :      <!-- Meta-Informationen -->
7 :          <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
8 :          <meta http-equiv="Content-Script-Type" content="text/javascript" />
9 :          <meta http-equiv="Content-Style-Type" content="text/css" />
10 :
runMax executed on: 28.07.10 17:36:30
PascalScript maXbox; (c) 2005 by RemObjects & SynEdit

```


2: The Output Window

The **Compile** button is also used to check that your code is correct, by verifying the syntax before the program starts. Another way to check the syntax before run is F2 or the **Syntax Check** in the menu Program. When you run this code you will see the content (first 10 lines) of the site in HTML format with the help of the method `memo2.lines.add`:

```

begin
    idHTTP:= TIdHTTP.Create(NIL)
    try
        memo2.lines.text:= idHTTP.Get2('http://www.softwareschule.ch')
        for i:= 1 to 10 do
            memo2.lines.add(IntToStr(i)+' :'+memo2.lines[i])
            //idhttp.get2('http://www.softwareschule.ch/maxbox.htm')
        finally
            idHTTP.Free
        end
end

```

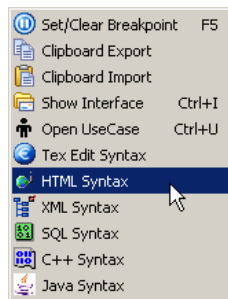
 The Object `TIdHTTP` is a dynamically allocated block of memory whose structure is determined by its class type. Each object has a unique copy of every field defined in the class, but all instances of a class share the same methods.


Next we step to `ex. 102_pas_http_download.txt`. We work with the method `Get1()` and one object of `TFileStream`. You can use them in your apps as a way to save or load files.

```
11 begin
12  myURL:= 'http://www.softwareschule.ch/download/maxbox_examples.zip';
13  zipStream:= TFileStream.Create('myexamples2.zip', fmCreate)
14  idHTTP:= TIdHTTP.create(NIL)
15  try
16    idHTTP.Get1(myURL, zipStream)
```

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 in databases. There are several stream classes, which all descend from `TStream`.

`TFileStream` reads from or writes to a file.



 So we pass to `Get1` the host name and a file stream. After writing to the file you can open the zip. Objects are created and destroyed by special methods called constructors and destructors.

The constructor:

```
zipStream:= TFileStream.Create();
```

The destructor (just the free method that calls the destructor):

```
zipStream.Free;
```

In Line 28 we find a last function of the RTL (Runtime Library) of Indy:

```
28 Writeln(DateTimeToInternetStr(Now, true))
```

We get the real time zone based time back! This information of RTL functions is contained in various unit files that are a standard part of Delphi or Indy. This collection of units is referred to as the RTL (run time library). The RTL contains a very large number of functions and procedures for you to use.


By the way: In my research and in my debugging, I found that the function `GetTimeZoneInformation` was returning a value oriented towards converting a Time from GMT to Local Time. We wanted to do the reverse for getting the difference.

The issue with `TIdMessage.UseNowForTime = False` bug was that the `TIdMessage` was calling Borland's date function instead of using the `Date` property.



Try to change the host address of the `myURL:=` with the host name of line 17, so you can download and save a html file, but change also the name in line 13 of `'myexamples2.zip'`



 Try to download mp3 file with the 109 example `109_pas_mp3_download.txt`

maXbox

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.

Furthermore, it also depends on the firewall in use at both ends. If it's automatic and recognises data that needs a response automatically it will work. It needs an administrator to open ports etc. you're stuffed or configured.

Hope you did learn in this tutorial the topic of the internet.

Feedback @

max@kleiner.com

Literature:

Kleiner et al., Patterns konkret, 2003, Software & Support

Links of maXbox and Indy:

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

<http://www.indyproject.org/Socket/index.EN.aspx>

<http://sourceforge.net/projects/maxbox>

<http://sourceforge.net/apps/mediawiki/maxbox/>

1.3 Appendix

```
Function DateTimeToInternetStr(const Value: TDateTime): String;
var
  strOldFormat, strOldTFormat,
  strDate: String;
  wDay,
  wMonth,
  wYear:WORD;
begin
  {needed to prevent wild results}
  Result := '';
  strOldFormat := ShortDateFormat ;

  ShortDateFormat := 'DD.MM.YYYY';

  // Date
  case DayOfWeek(Value) of
    1: strDate := 'Sun, ';
    2: strDate := 'Mon, ';
    3: strDate := 'Tue, ';
    4: strDate := 'Wed, ';
    5: strDate := 'Thu, ';
    6: strDate := 'Fri, ';
    7: strDate := 'Sat, ';
  end;
  DecodeDate(Value, wYear, wMonth, wDay);
  strDate := strDate + IntToStr(wDay) + #32;
```

```

case wMonth of
  1: strDate := strDate + 'Jan ';
  2: strDate := strDate + 'Feb ';
  3: strDate := strDate + 'Mar ';
  4: strDate := strDate + 'Apr ';
  5: strDate := strDate + 'May ';
  6: strDate := strDate + 'Jun ';
  7: strDate := strDate + 'Jul ';
  8: strDate := strDate + 'Aug ';
  9: strDate := strDate + 'Sep ';
 10: strDate := strDate + 'Oct ';
 11: strDate := strDate + 'Nov ';
 12: strDate := strDate + 'Dec ';
end;
//Correction
strOldTFormat := LongTimeFormat;
LongTimeFormat := 'HH:NN:SS';
strDate := strDate + IntToStr(wYear) + #32 + TimeToStr(Value);
Result := strDate + #32 + DateTimeToGmtOffsetStr(OffsetFromUTC,False);

LongTimeFormat := strOldTFormat;
{
  strOldTFormat := LongDateFormat;
  LongDateFormat := 'HH:NN:SS';
  strDate := strDate + IntToStr(wYear) + #32 + TimeToStr(Value);
  LongDateFormat := strOldTFormat;
  Result := strDate + #32 + DateTimeToGmtOffsetStr(OffsetFromUTC,False);
  ShortDateFormat := strOldFormat ;
}
end;

```