Under Construction: RobotBob Internet Agent

by Bob Swart

In this article we'll discuss the concept of intelligent agents for the internet. The techniques we'll use include CGI, FTP, SMTP and POP3.

There are a number of web pages on my bookmark list that I plan to visit every week or so. However, sometimes I just don't have the time and may miss an important announcement. At other times maybe I do visit a bookmarked site, but nothing has changed since last time I visited.

Now imagine a little program that would automatically keep an eye on a number of websites, informing us whenever something has changed. A bit like the subscription service (or 'channels') of Internet Explorer 4, or maybe even a bit like a mailing list.

As another example, has anyone ever tried to download a big patch file by ftp from a site overseas (like the BDE 4.51 patch at 10,269,420 bytes from the Inprise site)? Even with ISDN, more often than not the connection is broken before the file is completely downloaded. And with a 14.4K modem I don't even *try* to download files bigger than a megabyte from the web.

Now imagine a little program automatically downloading the file for us, to a remote but *closer* location, sending it to us by email, or leaving it in a place where we can pick it up without too much trouble (at our own ISP, for example).

Agents And Robots

There isn't a unique definition of a software agent or robot. Basically, the general idea is that they are able to perform a certain task or set of tasks without human intervention or special monitoring. Of course, they should be able to communicate with a human being, if only to receive instructions (which websites to 'check' or which file to download) or send the answers and responses to.

By the way, the word 'intelligent' in combination with 'agent' or 'robot' usually indicates that the software is able to adapt to certain situations or changes in the environment. This can be done, for example, by keeping a profile of clients (or users) and adapting the behaviour based on the information in that profile. Alternatively, the robot may find out that a certain website always gets updated at the weekend, but seldom during working days. That may be a reason to report this finding to the client, or only check on the weekends from now on.

In this article, we'll develop two web robots, one to download a file by ftp and optionally send it on by email, and another to maintain a mailing list, for discussions or announcements of some sort. Making these robots truly intelligent was not part of my focus for this article, so I'll leave that to you!

RobotBob/FTP

The first robot is called RobotBob/FTP and can download a file specified by a remote location (URL) from the internet. We first need to implement the FTP protocol. And no, I don't want to use an existing ActiveX control, but rather re-invent the wheel myself (after all, this *is* called the *Under Construction* column, right?).

FTP stands for File Transfer Protocol, which is formally described in RFC 959. The FTP communication model can be implemented using sockets, just like the SMTP protocol we implemented last issue, or the POP3 protocol which we'll implement soon.

However, some time ago now, Microsoft released WININET, a special layer on top of the low-level internet APIs especially for Win32 programmers. WININET offers a higher level interface to otherwise low-level protocols such as HTTP and FTP. It's really easy to use and the best thing is the `WININET` unit with the API definitions in ObjectPascal is included with Delphi.

WININET uses something called an 'internet handle' (much like Windows uses Windows handles), and all APIs either need or return such a handle. For example, to open a new WININET session, we need to call `InternetOpen`, which returns a handle which we must use (and pass to other APIs) until the end of our session. To close any internet handle, we must always call `InternetCloseHandle` (so after we've received a handle we can use, we should immediate write a `try...finally` block, where we close the handle in the `finally` part).

To open a remote file (or URL) on the internet, we must call `InternetOpenURL`, which again returns an internet handle. Now, to download that remote file (URL) to our local machine, we only have to make a number of calls to `InternetReadFile`, which works a bit like `BlockRead` in that it copies data from the remote file to a data buffer. We can use `BlockWrite` to write the data from the buffer to a local file, and hence with only three WININET functions (four if we count the closing `InternetCloseHandle` function), we can write a basic but fast FTP download routine as can be seen in Listing 1.

Note that during the FTP process we report the status on the standard output in increasing steps (8, 16, 32, 64, 128, 256 and 512Kb, after which every 512Kb, until the file is completely downloaded, or an error occurs (like a broken connection). This gives an indication of progress, although the web server I used to deploy the `CopyURL` function has a T1 connection to the internet, so even files of 10Mb and more are downloaded in a reasonably short time anyway. Remember, you are downloading to a server, not to your (client) PC.

Instructions? CGI

Now we've written the 'engine' part, or the middleware, let's concentrate on the front end: the part of the application that receives the instructions about which URL to download into what file, and (optionally) who to mail it to. By far the easiest way to do this is just like any other CGI application, with a CGI HTML Form and three controls: one for the URL, one for the file and one for the email address. The `Action` will be the CGI application RobotFTP.exe that we're about to finish, using the `POST` method to send data. The CGI HTML Form is defined in Listing 2.

And if we try RobotBob/FTP in action as the 'live' URL on the internet at www.drbob42com/RobotBob we'll get the display shown in Figure 1.

Note that the email part is optional (on purpose). If we don't specify an email address, then the URL just gets downloaded to the local file in the www.drbob42.com/robotbob/ftp directory (which now indeed holds the delphiQA.zip file of just over 1Mb). While this may not sound too useful, I think it is, as my link to my own web server is much faster than my link to the Inprise website, and so is downloading. So if your mailer doesn't support messages with huge attachments, you can skip the email address and just make sure the file gets effectively uploaded to your own ISP where you can pick it up later (but much faster compared to a direct FTP from the original website).

Response By Email

Given a URL and local filename, it would be even more useful at times to be able to send the file back to our mailbox, uuencoded of course. That way, we never have to go through the trouble of trying to download a big file, but instead can use a simple robot that will FTP

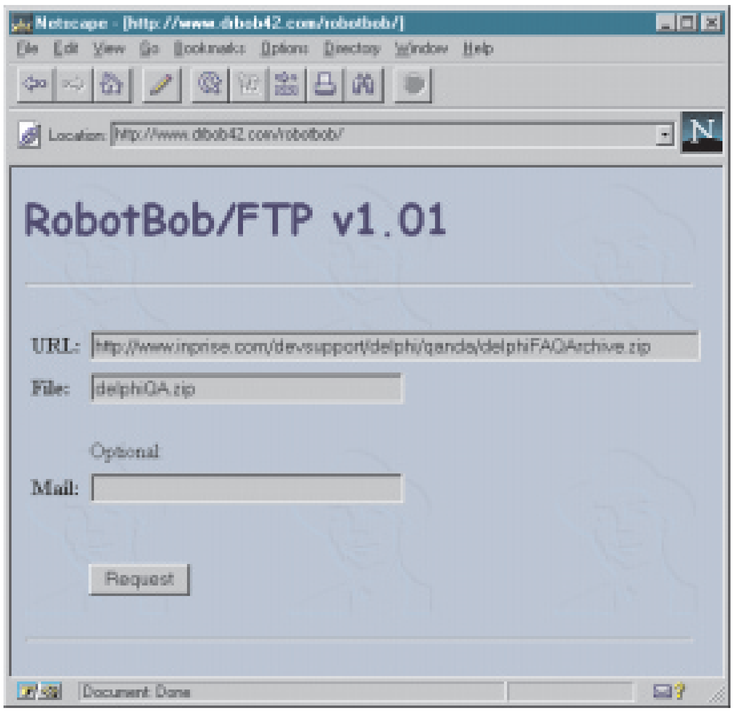
```
procedure CopyURL(const URL, OutputFile: String);
const
  BufferSize = 8 * 1024;
var
  hSession, hURL: HInternet;
  Buffer: Array[0..Pred(BufferSize)] of Byte;
  BufferLength: DWORD;
  i: Integer;
  f: File;
begin
  hSession := InternetOpen('DrBob',INTERNET_OPEN_TYPE_PRECONFIG,nil,nil,0);
  try
    hURL := InternetOpenURL(hSession, PChar(URL), nil,0,0,0);
    try
      Assign(f, OutputFile);
      Rewrite(f,1);
      writeln('Downloading...<P>');
      i := 0;
      repeat
        InternetReadFile(hURL, @Buffer, BufferSize, BufferLength);
        Inc(i);
        if (i in [1,2,4,8,16,32,64]) or (i mod 64 = 0) then
          writeln(i*8,'KBytes...<BR>');
        BlockWrite(f, Buffer, BufferLength)
      until BufferLength < BufferSize;
      Close(f);
      writeln('<BR>',(i-1)*BufferSize + BufferLength,' Bytes downloaded.');
    finally
      InternetCloseHandle(hURL)
    end
  finally;
    InternetCloseHandle(hSession)
  end
end;
```

➤ *Listing 1: CopyURL.*

```
<HTML>
<HEAD>
<TITLE>RobotBob/FTP</TITLE>
</HEAD>
<BODY BACKGROUND="../gif/back.gif">
<H1>RobotBob/FTP v1.0</H1>
<HR>
<FORM ACTION="../cgi-bin/RobotFTP.exe" METHOD=POST>
<TABLE>
<TR><TD><B>URL:</TD><TD><INPUT TYPE=edit NAME=URL SIZE=64></TD></TR>
<TR><TD><B>File:</TD><TD><INPUT TYPE=edit NAME=File SIZE=32></TD></TR>
<TR><TD></TD><TD>Optional:</TD></TR>
<TR><TD><B>Mail:</TD><TD><INPUT TYPE=edit NAME=Mail SIZE=32></TD></TR>
<TR><TD></TD><TD><BR><INPUT TYPE=SUBMIT VALUE=Request></TD></TR>
</TABLE>
</FORM>
<HR>
</BODY>
</HTML>
```

➤ *Listing 2: RobotBob/FTP HTML.*

➤ *Figure 1*


Netscape - [http://www.drbob42.com/robotbob/]
File Edit View Go Bookmarks Options Directory Window Help
Location: http://www.drbob42.com/robotbob/
Document: Done

RobotBob/FTP v1.01

URL: http://www.inprise.com/devsupport/delphi/qanda/delphiFAQArchive.zip

File: delphiQA.zip

Optional

Mail:

Request

and email the results to us. Provided that our mail client can handle messages with big attachments, this should help when downloading the next patch of Delphi, Windows 98, Internet Explorer or whatever.

In the last issue, we developed the TBSMTP component and this time we also need the TBUUCode component to uuencode the (usually binary) attachment. The TBUUCode component is based on code from *Under Construction* in Issue 4, November 1995. Full 32-bit source code is on the accompanying disk with this issue. Apart from these two components, we also need the DrBobCGI unit (first described in Issues 29, 30 and 33) to obtain the CGI Form variables the user specified.

The RobotBob/FTP program is indeed a standard CGI application: using the DrBobCGI unit, it only needs to parse the three input fields URL, File and Mail. It then uses the CopyURL routine to download the URL to the local filename, and optionally sends the uuencoded local file to the email address specified in Mail *(please, don't use this to send mailbombs to your friends, and I do keep a logfile of all use of RobotBob/FTP on my site...).*

Note that the two calls to ExtractFileName on Value('File') are actually needed to prevent anyone from downloading a CGI executable to my cgi-bin directory (by downloading hack.exe to ..\..\cgi-bin\hack.exe or \cgi-bin\hack.exe for example). This could violate system integrity, of course, and now I'm sure that every downloaded file will end up in the /RobotBob/FTP directory.

As an example of using RobotBob/FTP, I always like to download the latest collection of Delphi FAQ items from Inprise's website. I once found out that Inprise keeps an archive of these at

www.inprise.com/devsupport/
delphi/qanda/
delphiFAQArchive.zip

Of course, ideally I should check if this file has changed before trying to download it, but maybe that's something I can teach my robot.

RobotBob/FTP reports the number of the request (Figure 2). In fact, it keeps a logfile of downloaded files, including the recipient of the downloaded file, so this can be used as a start of the user profile. I can enhance the RobotFTP with some location detection code, that may detect the fact that a certain user mainly downloads files from the Inprise developer support for Delphi download area. That might be a reason to send this particular user a list of new files that appear in that particular location on a weekly

➤ *Listing 3: RobotFTP.*

```
program RobotFTP;
{$I+,O+}
{$APPTYPE CONSOLE}
uses
  Windows, WinINet, SysUtils, DrBobCGI, DrBobUUE, DrBobEml;
  { see previous listing for implementation of CopyURL }
var
  f: Text;
  Str: String;
  counter: Integer;
begin
  ShortDateFormat := 'YYYY/MM/DD';
  writeln('content-type: text/html');
  writeln;
  writeln('<HTML>');
  writeln('<HEAD>');
  writeln('<TITLE>',Value('URL'),
    ' -> ',Value('File'),'</TITLE>');
  writeln('</HEAD>');
  writeln('<BODY BACKGROUND="/gif/back.gif">');
  writeln('<H1>RobotBob/FTP v1.0</H1>');
  writeln('<HR>');
  writeln('<BR>URL=',Value('URL'));
  writeln('<BR>File=',Value('File'));
  writeln('<BR>Mail=',Value('Mail'));
  writeln('<P>');
  ChDir('RobotBob'); { place to leave FTP-files }
  {$I-}
  Assign(f,'robotbob.log');
  Reset(f);
  counter := 0;
  if IOResult = 0 then
    while not eof(f) do begin
    readln(f);
    Inc(counter)
  end else
    Rewrite(f);
  if IOResult <> 0 then
    { skip };
  Append(f);
  Inc(counter);
  writeln(f,counter:4,': ', Value('URL'),' -> ',
    Value('File'),' to ',Value('Mail'));
  Close(f);
  if IOResult <> 0 then
    { skip };
  writeln('Request #', counter, ' at ',
    DateTimeToStr(Now),'<P>');
  {$I+}
  ChDir('FTP');
  if (Value('URL')<>'') and (Value('File')<>'') then begin
    CopyURL(Value('URL'), ExtractFileName(Value('File')));
    if Value('Mail') <> '' then begin
      with TBUUCode.Create(nil) do
      try
        InputFile := ExtractFileName(Value('File'));
        Str := InputFile;
        if Pos('.',Str) > 0 then
          Delete(Str,Pos('.',Str),255);
        if Str = '' then
          Str := 'output';
        OutputFile := Str + '.uue';
        Algorithm := uuencode;
        try
          UUCode;
          with TBSMTP.Create(nil) do
          try
            MailServer := 'smtp.server.com';
            MessageFrom := 'RobotBob';
            MessageTo := Value('Mail');
            MessageSubject := 'Automatic E-mail using SMTP';
            MessageText.Add('Hi '+Value('Mail')+',');
            MessageText.Add('');
            MessageText.Add('URL='+Value('URL'));
            MessageText.Add('File='+Value('File'));
            MessageText.Add('UUCode='+OutputFile);
            MessageText.Add('Mail='+Value('Mail'));
            MessageText.Add('');
            System.Assign(f,OutputFile);
            Reset(f);
            while not eof(f) do begin
              readln(f,Str);
              MessageText.Add(Str)
            end;
            System.Close(f);
            Erase(f);
            MessageText.Add('');
            MessageText.Add('Groetjes,');
            MessageText.Add(
              '           RobotBob - RobotBob@drbob42.com');
            writeln(
               '<P>Sending mail to ',Value('Mail'),'...');
            SendMail;
            writeln('<P>Sent.');
          finally
            Free
          end
        except
          on E: Exception do
            writeln('<P><B>Error: </B>',E.Message)
        end
      finally
        writeln('<P>Done.');
        Free
      end
    end
  end;
  writeln('</BODY>');
  writeln('</HTML>')
end.
```

basis, so its easier for the user to decide which files (if any) to download and email this time.

Instructions By Mail

Now it's time to consider those of us who are unable to connect to the internet during the day (because they have to operate on a company intranet within a firewall, for example). If this applies to you, then I know how you feel, because I also only have internet access at home, unless I want to go to the library. And yet I want to be able to download some files right now, and get them to my mailbox!

For this to work, all we need is the ability to send instructions by email rather than by CGI. Email usually gets through every firewall, although it may encounter some delays along the way.

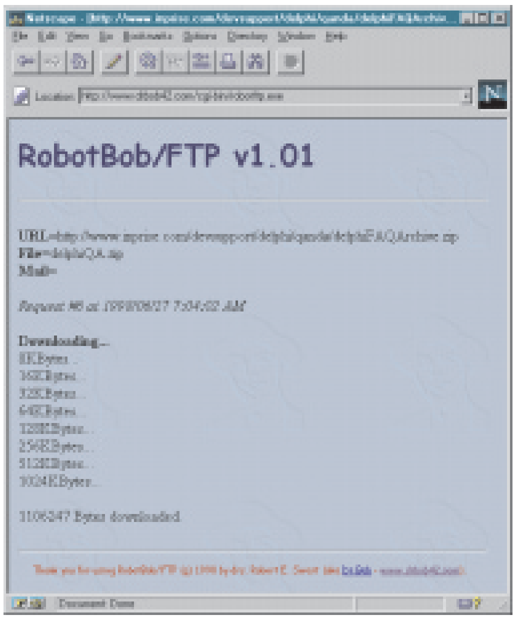
Anyway, to be able to send email instructions to RobotBob/FTP, we have to make sure RobotBob in turn can both receive and read those email messages. And where we used SMTP to send messages, we'll now use POP3 to read messages, as seen in Listing 4 which presents my `TBPOP3` component.

POP3 is a protocol that can be implemented using sockets and thus Delphi's `TClientSocket` component, connected to Port 110 (as described in RFC 1081 and 1725) is what we need. In fact, for the POP3 component I use a technique similar to the one I used for SMTP, with the differences only defined by the difference in 'communication' between POP3 and SMTP.

With the POP3 protocol, we start by sending the username (`USER`) and password (`PASS`) to make ourselves known, and get permission to access the mailbox. We can then use `STAT` to find out how many messages are waiting (if any), `RETR` to retrieve a single message, and `DELE` to delete a specific message. `RETR` and `DELE` both need a message number as their argument, of course. Finally, we can `QUIT` (making all changes to the mailbox permanent) or `RSET` (reset the mailbox, undeleting deleted messages).

The only other thing worth noticing is that a `RETE` command may not return the entire message at once. The POP3 mail server I used for my testing only returned blocks of about 8Kb at a time, which means we have to loop and add blocks until we reach the end (which is denoted by a single '.' on an empty line, which is the last line of the message).

Note that `TBPOP3` only compiles with Delphi 4 (a good reason to upgrade?), as I already used the new, very handy, feature of dynamic arrays to store the (unknown number of) messages. If you need to use this component with Delphi 3, you can simply replace the `Array of String` with a `StringList`, and use that structure to store the content of each email message.



URL=http://www.inprise.com/devsupport/delphi/qanda/delphiFAQArchive.zip
File=delphiQA.zip
Mail=

Request #8 at 1998/08/27 7:34:02 AM

Downloading...
8KBytes...
16KBytes...
32KBytes...
64KBytes...
128KBytes...
256KBytes...
512KBytes...
1024KBytes...

1106247 Bytes downloaded.

Thank you for using RobotBob/FTP (c) 1998 by drs. Robert E. Swart (aka Dr.Bob - www.drbob42.com)

Document: Done

➤ *Figure 2*

I leave it as an exercise for the reader to implement the last step, and design a 'command language' and accompanying parser for RobotBob/FTP in order to receive its instructions. We're continuing now with yet another application (pun intended) for the `TBPOP3` component, a dedicated mailing list.

RobotBob/Mail: Distribution List

Having a `TBPOP3` component available opens a number of new possibilities for RobotBob. We can now send and receive email messages and respond to them in a semi-intelligent manner.

As an example, I've set up a special email account (robotbob @drbob42.com) that is only used to receive messages and will forward these messages to a group of people who are subscribed to the RobotBob mailing list. People inside the list only need to send messages to RobotBob@drbob42 .com, and automatically everyone on the list will get the message in their mailbox.

This is very useful for a news or announcement mailing list, or for a group discussion list (where multiple people want to discuss certain things together, without having to 'CC' everyone at all times). It's almost like a newsgroup, but again, this feature should also be available to those of us who don't

Delphi Dynamic Arrays

Dynamic arrays specify type information (the number of dimensions and the type of the elements) but not the number of elements. Thus, our `fMessage: Array of String;` declares a dynamic array of strings.

Dynamic arrays do not have a fixed size or length. Instead, memory for a dynamic array is allocated when we pass it to `SetLength`, such as the call to `SetLength(fMessage, 42)`.

Dynamic arrays are always integer-indexed, always starting from 0, and we can use the `High` function to return the highest possible valid index of the dynamic array. Note that I actually used the dynamic `Array of Strings` as implementation for the indexed property `Message`, using the `GetMessage` function to return the appropriate string at the specified index of the dynamic array (or an empty string if the index was out-of-bounds).

All in all, I find dynamic arrays to be a very convenient addition to Delphi's Object Pascal language.

have access to the ‘outside world’, and only have email.

Without thinking too much about the way the application is actually running, we can first focus on the main ‘loop’ within the new Robot, which is triggered by a timer event. The timer event can be set at a regular, but not too short, interval of say 10 minutes, which means that every 10 minutes the Robot will get activated and use POP3 to login to the mail server,

➤ *Listing 4: DrBobPOP.*

```
unit DrBobPOP;
interface
uses Classes, ScktComp;
type
  TBPOP3 = class(TComponent)
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
  public
    procedure ReadMail;
    procedure DeleteMessage(Nr: Integer);
  protected
    _Socket: TClientSocket;
    Step,Mess: Integer;
    procedure SocketRead(Sender: TObject; Socket:
      TCustomWinSocket);
    procedure SocketWrite(Sender: TObject; Socket:
      TCustomWinSocket);
  private
    fMailServer: String;
    fUser: String;
    fPassword: String;
    fMessages: Integer;
  published
    property MailServer: String
     read fMailServer write fMailServer;
    property User: String read fUser write fUser;
    property Password: String write fPassword;
  protected
    function GetMessage(Index: Integer): String;
  public
    property Messages: Integer read fMessages;
    property Message[Index: Integer]: String read
      GetMessage;
  private
    LastSocket: TCustomWinSocket;
    fMessage: Array of String;  // Delphi 4 feature
    Status: String;
  end;
implementation
uses
  SysUtils, Forms; { for Application.ProcessMessages loop }
const
  St_USER = 1;
  St_PASS = 2;
  St_STAT = 3;
  St_RETR = 4;
  St_QUIT = 5;
const
  CRLF = #13#10;
constructor TBPOP3.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  _Socket := TClientSocket.Create(Self);
  _Socket.Port := 110;
  _Socket.OnRead := SocketRead;
  _Socket.OnWrite := SocketWrite;
  LastSocket := nil;
  Step := 0
end {Create};
destructor TBPOP3.Destroy;
begin
  _Socket.OnRead := nil;
  Step := St_QUIT;
  if Assigned(LastSocket) then
    LastSocket.SendText('QUIT'+CRLF);
  _Socket.Free;
  _Socket := nil;
  fMessage := nil;
  inherited Destroy
end {Destroy};
function TBPOP3.GetMessage(Index: Integer): String;
begin
  if Index in [0..High(fMessage)] then
    Result := fMessage[Index] else Result := ''
end {GetMessage};
procedure TBPOP3.SocketRead(Sender: TObject;
  Socket: TCustomWinSocket);
var
  EOM: Boolean;
  i: Integer;
begin
  LastSocket := Socket; { talk back ? }
  Status := Socket.ReceiveText;
  EOM := Pos(#13#10'.'#13#10,Status) = Length(Status)-4;
  while (Length(Status) > 0) and
    (Status[Length(Status)] in [#10,#13]) do
    Delete(Status,Length(Status),1);
  case Step of
    0: Step := St_USER;
    St_USER :
      if Pos('-ERR',Status) > 0 then
        Step := St_QUIT
      else
        Step := St_PASS;
    St_PASS :
      if Pos('-ERR',Status) > 0 then
        Step := St_QUIT
      else
        Step := St_STAT;
    St_STAT :
      if Pos('+OK',Status) = 1 then
      try
        try { get number of messages }
          Delete(Status,1,3);
          while Status[1] = #32 do Delete(Status,1,1);
          Delete(Status,Pos(#32,Status),255);
          fMessages := StrToInt(Status);
          Mess := fMessages;
          if fMessages > 0 then begin
            SetLength(fMessage,Mess);  // Delphi 4 feature
            for i:=Pred(fMessages) downto 0 do
              fMessage[i] := ''
          end
        except
          fMessages := 0
        end
      finally
        if fMessages <= 0 then
          Step := St_QUIT { Bye, Bye }
        else begin
          Status := '+OK'; { retrieve first message }
          Mess := 1;
          Step := St_RETR
        end
      end;
    St_RETR :
      begin
        fMessage[Pred(Mess)] := fMessage[Pred(Mess)] +
          Status;
        if EOM then begin
          Delete(fMessage[Pred(Mess)],1,Pos(#13#10,
            fMessage[Pred(Mess)])+1);
          Delete(fMessage[Pred(Mess)],
            Length(fMessage[Pred(Mess)]),1);
          Inc(Mess);
          Status := '+OK' { retrieve next message }
        end;
        if Mess > fMessages then
          Step := St_QUIT
      end;
    St_QUIT : Inc(Step)
  end;
  SocketWrite(Sender, Socket)
end {SocketRead};
procedure TBPOP3.SocketWrite(Sender: TObject;
 Socket: TCustomWinSocket);
var Send: String;
begin
  Send := 'NOOP';
  case Step of
    St_USER : Send := 'USER ' + fUser;
    St_PASS : Send := 'PASS ' + fPassword;
    St_STAT : Send := 'STAT';
    St_RETR :
      if Status = '+OK' then
        Send := 'RETR ' + IntToStr(Mess);
    St_QUIT : Send := 'NOOP'; // 'QUIT';
  end;
  if (Step in [St_USER..St_QUIT]) and (Send <> 'NOOP') then
    Socket.SendText(Send + CRLF)
end {SocketWrite};
procedure TBPOP3.ReadMail;
begin
  Step := 0;
  _Socket.Active := False;
  _Socket.Host := fMailServer;
  _Socket.Open;
  repeat
    Application.ProcessMessages
  until Step >= St_QUIT
end {ReadMail};

procedure TBPOP3.DeleteMessage(Nr: Integer);
begin
  Step := St_QUIT;
  if Assigned(LastSocket) then
    LastSocket.SendText('DELE '+ IntToStr(Nr) + CRLF);
  repeat
    Application.ProcessMessages
  until Step > St_QUIT
end {DeleteMessage};
end.
```

check the number of messages, download the messages (but leave them on the server for now), check to see if any of these messages is intended for RobotBob@drbob42 .com, and then forward this message (using the SMTP component) to the relevant group of people, finally deleting this message from the mail server.

The `TimerTimer` event reports the status on standard output. In fact, it just 'appends' to an existing file called mailbob.html which I can check every 10 minutes to see if there were any new messages, and if RobotBob/Mail decided to do something with any of these messages.

If you check the source code in Listing 5 for the `TimerTimer` event, you may note that I did not include the list of subscribed people here. Rather, I only send the message from RobotBob to myself. You can either repeat this piece of code for every subscriber, or modify the `TBSMTP` component to allow multiple recipients or a big CC or BCC list.

In my enhanced version of the `TBSMTP` component, I solved this problem by adding a list of (blind) receivers to a message, thereby making sure every subscriber on the mailing list would get a message, but without actually publishing the email address of each member.

Artificial Life

Now that we know how RobotBob/Mail works, based on a timer event, let's think about how it would actually get activated. Since we need a timer event from inside a running program to activate the Robot itself, it would seem that the application would need to be running at all times. Can we do this with a CGI application? I would think not, as a CGI application basically gets executed, generates some dynamic output, and then terminates. Generally, CGI applications are not meant to 'stay alive' (as opposed to ISAPI DLLs). we could try to use an ISAPI DLL, but these have the disadvantage of being hard to unload and update.

So, MailBob remains a 'regular' application that runs on the web server. Using a normal form with a `Timer` component (and `TimerTimer` event) and `FormCreate` and `FormDestroy` events to set up and clean up the RobotBob/Mail things, like changing to the right directory, initiating the logfile mailbob.html, etc.

The `TMailBox.TimerTimer` event contains the code for MailBob that we saw in the previous listing. Note that we set the `Application.ShowMainForm` property to `False`, indicating that the main form (`TMailBox`) should remain invisible at all times, so the webmaster at the console won't see our application running.

Most of the time, the application does nothing, it simply waits for the `TimerTimer` event to fire every 10 minutes (every 600000 milliseconds). This ensures that performance on the web server does not suffer from this application that should be running at all times.

RobotBob/Start

Now that we've decided to make RobotBob/Mail a regular application, it's time to figure out how to actually start this application remotely (I don't have physical access to my web server machine, which is located somewhere in the USA). I decided to use the easiest way I could think of: a call to WinExec made from a regular CGI application.

Note that we do need to ensure that we can find the executable, so we may need to change directories. It's also important to make sure that only one instance of

➤ *Listing 5: TimerTimer Event.*

```
procedure TMailBox.TimerTimer;
var i,j,k: Integer;
begin
  System.Assign(output,'mailbob.html');
  Append(output);
  writeln;
  writeln('MailBob waking up at: ',DateTimeToStr(Now));
  try
    with TBPOP3.Create(nil) do
    try
      MailServer := 'pop3.server.com';
      User := 'drbob';
      Password := '********';
      ReadMail;
      writeln(Messages:2,' messages waiting...');
      Lines := TStringList.Create;
      for i:=0 to Pred(Messages) do begin
        System.Assign(f,'mailbox');
        Rewrite(f);
        writeln(f,Message[i]);
        System.Close(f);
        Lines.LoadFromFile('mailbox');
        k := 0;
        while (k < Lines.Count) and (Lines[k] <> '') do
          Inc(k);
        for j:=Pred(k) downto 0 do begin
          if (Pos('From: ',Lines[j]) <> 1) and
            (Pos('To: ',Lines[j]) <> 1) and
            (Pos('Subject: ',Lines[j]) <> 1) and
            (Pos('Date: ',Lines[j]) <> 1) then
            Lines.Delete(j)
          else begin
            if Pos('From: ',Lines[j]) = 1 then
              From := Copy(Lines[j],7,255)
            else if Pos('To: ',Lines[j]) = 1 then
              MailTo := Copy(Lines[j],4,255)
            else if Pos('Subject: ',Lines[j]) = 1 then
              Subject := Copy(Lines[j],10,255)
            else
              Date := Copy(Lines[j],6,255)
          end
        end {headers};
        if Pos('robotbob@drbob42.com',
          LowerCase(MailTo)) > 0 then begin
          // repeat for each subscriber,
          // or use CC-list in SMTP component
          with TBSMTP.Create(nil) do
          try
            MailServer := 'smtp.server.com';
            MessageFrom := 'RobotBob';
            MessageTo := 'drbob@drbob.demon.nl';
            MessageSubject := Subject;
            MessageText.Add(
              'Forwarded messages from '+From);
            MessageText.Add('');
            writeln;
            k := 0;
            while (k < Lines.Count) and (Lines[k] <> '') do
              Inc(k);
            for j:=0 to Pred(k) do writeln(Lines[j]);
            for j:=k to Pred(Lines.Count) do
              MessageText.Add(Lines[j]);
            SendMail;
            write('forwarded to ',MessageTo);
          finally
            Free
          end;
          DeleteMessage(i+1);
          writeln(' and deleted from mailhost')
        end
      end
    finally
      Free
    end;
  except
    on E: Exception do
      writeln('<P><B>Error:</B> ',E.Message)
  end;
  System.Close(output)
end;
```

```
program MailBob;
{$R MAILBOB.DFM}
{$I-,O+}
uses
  Forms, Extctrls, SysUtils, Classes, DrBobPop, DrBobEml;
type
  TMailBox = class(TForm)
    Timer: TTimer;
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure TimerTimer(Sender: TObject);
  private
    From,MailTo,Subject,Date: ShortString;
    Lines: TStringList;
    f: Text;
  end;
  procedure TMailBox.FormCreate;
  begin
    ChDir('..');
    ChDir('RobotBob');
    System.Assign(output,'mailbob.html');
    Rewrite(output);
    writeln('<HTML>');
    writeln('<BODY BACKGROUND="/gif/back.gif">');
    writeln('<HEAD>');
    writeln('<TITLE>RobotBob/Mail</TITLE>');
    writeln('</HEAD>');
    writeln('<FONT FACE="Comic Sans MS"COLOR=000077>');
    writeln('<H1>RobotBob/Mail v1.0</H1>');
    writeln('</FONT>');
    writeln('<PRE>');
    writeln('MailBob initiated at: ',DateTimeToStr(Now));
    System.Close(output);
    Timer.Enabled := True
  end {FormCreate};
  procedure TMailBox.FormDestroy;
  begin
    System.Assign(output,'mailbob.html');
    Append(output);
    writeln;
    writeln('MailBob closing down: ',DateTimeToStr(Now));
    writeln('</BODY>');
    writeln('</HTML>');
    System.Close(output)
  end {FormDestroy};
{ see previous listing for TMailBox.TimerTimer event code }
var MailBox: TMailBox;
begin
  Application.Initialize;
  Application.ShowMainForm := False;
  Application.CreateForm(TMailBox, MailBox);
  Application.Run
end.
```

➤ *Listing 6: MailBob.*

RobotBob/Mail gets started, and we can ensure that by checking a call to `FindWindow`, where `TApplication` is the type of every Delphi application main Window, and MailBob is the name of that main Window in our case.

Listing 7 shows the code used to start RobotBob/Mail remotely, and the effect can be seen Figure 3. Note that a WinExec value higher than 32 means success, so RobotBob/Mail is now up and running.

RobotBob/Mail

Once RobotBob/Mail is running, we can check the logfile to see if new messages are waiting and whether or not RobotBob/Mail decided to process them.

In real life, RobotBob/Mail now handles two mailing lists as well as an auto-response list (which actually replies to a message to send to it). The latter can be tried at WebDoc@drbob42.com, which is installed as an email search engine interface to a Delphi Internet Programming FAQ on my website at www.drbob42.com/faq. Send a message to WebDoc@drbob42.com to find out more.

RobotBob/Kill

Being able to load RobotBob/Mail remotely is nice, but we should also be able to unload it, otherwise we're no better off than using an ISAPI DLL. Fortunately, anything that we can load with WinExec can be unloaded by telling it to quit. This can be done by sending a `WM_QUIT` message, for example, or a `WM_CLOSE` message, or a `WM_DESTROY` message (or all three of them just to be sure).

➤ *Listing 7: StartBob.*

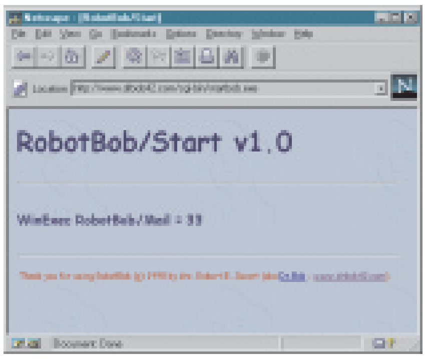
```
program StartBob;
{$APPTYPE CONSOLE}
{$I-}
uses
  Windows;
begin
  writeln('content-type: text/html');
  writeln;
  writeln('<HTML>');
  writeln('<HEAD>');
  writeln('<TITLE>RobotBob/Start</TITLE>');
  writeln('</HEAD>');
  writeln('<BODY BACKGROUND="/gif/back.gif">');
  writeln('<FONT FACE="Comic Sans MS"COLOR=000077>');
  writeln('<H1>RobotBob/Start v1.0</H1>');
  writeln('<HR>');
  writeln('<P>');
  ChDir('cgi-bin');
  if FindWindow('TApplication','MailBob') = 0 then begin
    writeln('WinExec RobotBob/Mail = ');
    writeln(WinExec('MailBob.exe',SW_NORMAL))
  end else
    writeln('RobotBob/Mail already running...');
  writeln('</BODY>');
  writeln('</HTML>')
end.
```

In practice, the CGI application RobotBob/Kill works just fine to remotely bring RobotBob/Mail down. And we can use these techniques to remotely start or stop any application on a web server. As long as we have initial permission to upload CGI applications ourselves, that is, and as long as the NT security settings allow us to do so. Something to keep in mind.

Since the web server is shut down and rebooted every day at 4.00am (local US time), this means I seldom need to actually run RobotBob/Kill, only if I want to upload a new version of RobotBob/Mail. And I need to run RobotBob/Start every morning to get RobotBob up and running again.

This could be automated as well, using schedulers or registry settings (turning RobotBob/Mail into a Windows NT Service comes to mind). But I will leave it as an exercise for you to ponder on that one. For now, I think we've seen enough artificial life (with or without intelligence).

➤ *Figure 3*



RobotBob/Start v1.0

WinExec RobotBob/Mail = 33

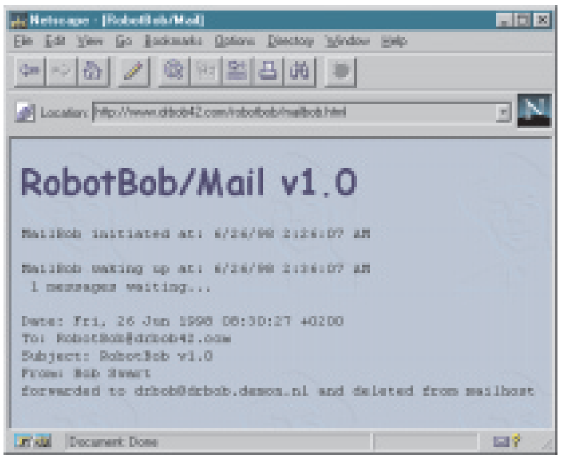
Next Time...

Next time, I'll look inside the Delphi 4 Client/Server box and report on the new internet components, both the Web Modules and others, including a few words on MiddleWare support with CORBA and MIDAS.

Bob Swart (aka Dr.Bob, visit www.drbob42.com) is a professional knowledge engineer technical consultant using Delphi, C++Builder and JBuilder for Bolesian (www.bolesian.com) and freelance technical author. In his spare time, Bob likes to watch video tapes of Star Trek Voyager and Deep Space Nine with his 4-year old son Erik Mark Pascal and his 1.5-year old daughter Natasha Louise Delphine.

```
program KillBob;
{$APPTYPE CONSOLE}
{$I-}
uses
  Windows, Messages;
begin
  writeln('content-type: text/html');
  writeln;
  writeln('<HTML>');
  writeln('<HEAD>');
  writeln('<TITLE>RobotBob/Kill</TITLE>');
  writeln('</HEAD>');
  writeln('<BODY BACKGROUND="/gif/back.gif">');
  writeln('<FONT FACE="Comic Sans MS"COLOR=000077>');
  writeln('<H1>RobotBob/Kill v1.0</H1>');
  writeln('<HR>');
  writeln('<P>');
  ChDir('cgi-bin');
  writeln('Killing RobotBob/Mail...<BR>');
  if FindWindow('TApplication','MailBob') <> 0 then begin
    writeln('Killing RobotBob/Mail...<BR>');
    PostMessage(FindWindow('TApplication','MailBob'),
      WM_DESTROY,0,0);
    PostMessage(FindWindow('TApplication','MailBob'),
      WM_CLOSE,0,0);
    PostMessage(FindWindow('TApplication','MailBob'),
      WM_QUIT,0,0)
  end;
  writeln('<P>Killed.');
  writeln('</BODY>');
  writeln('</HTML>')
end.
```

➤ *Listing 8: KillBob.*


Netscape - [RobotBob/Mail]
File Edit View Go Bookmarks Options Directory Window Help
Location: http://www.drbob42.com/robotbob/mailbob.html
Document: Done

RobotBob/Mail v1.0

```
MailBob initiated at: 6/26/98 2:26:07 AM

MailBob waking up at: 6/26/98 2:36:07 AM
 1 messages waiting...

Date: Fri, 26 Jun 1998 08:30:27 +0200
To: RobotBob@drbob42.com
Subject: RobotBob v1.0
From: Bob Swart
forwarded to drbob@drbob.demon.nl and deleted from mailbost
```

➤ *Figure 4*