#contentWrapper #fs, #sidebarContent #fs, #contentWrapper div [id * = 'myExtraContent'], #sidebarContent div [id * = 'myExtraContent'] {display: block;}

The Kermith workshop (https://translate.googleusercontent depth=1&hl=en&prev=search&rurl=translate.google.com&sl

The other way to see supervision ...

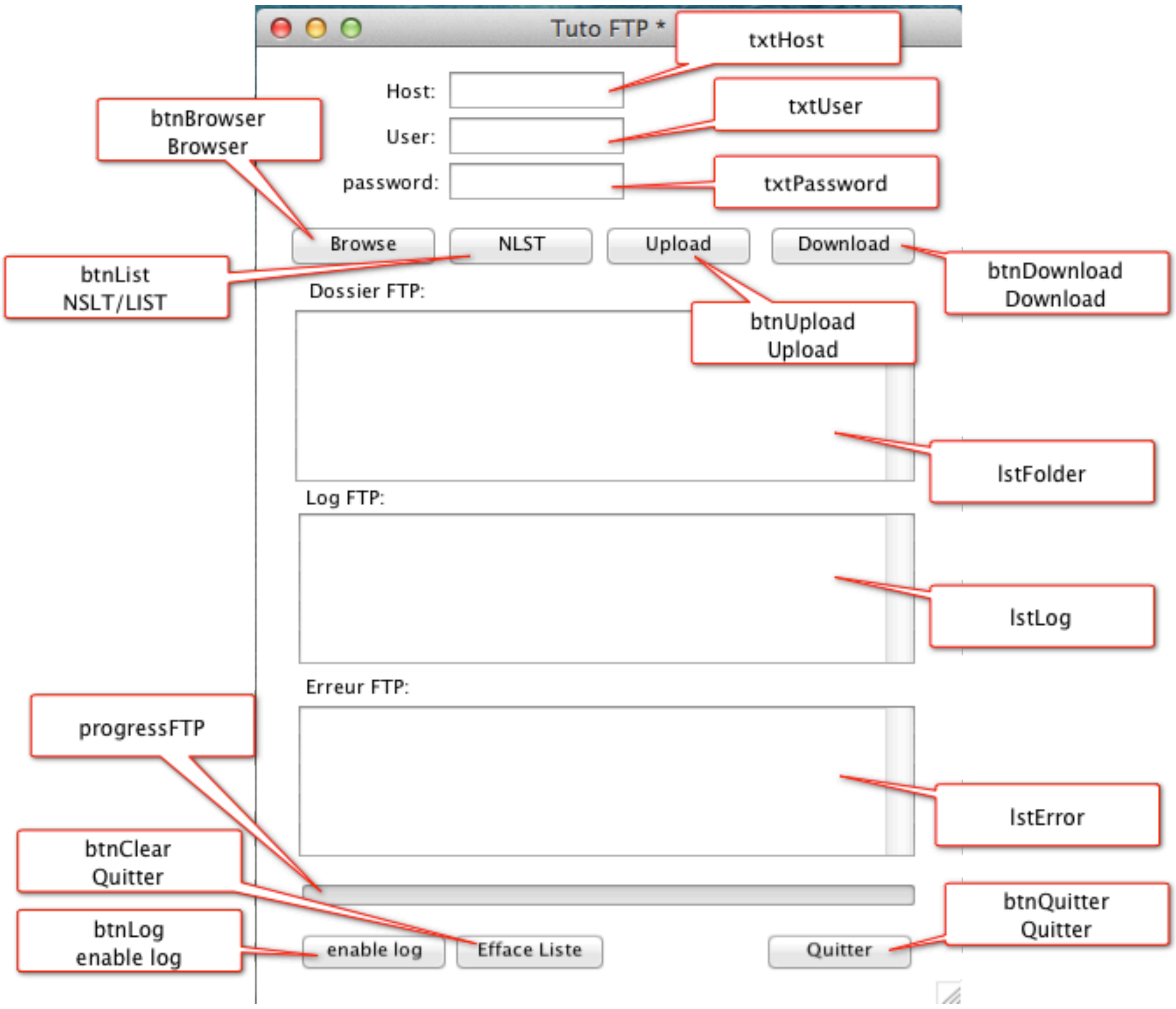
Use FTP with LiveCode

Here's a little tutorial on an FTP client with LiveCode. I would not do a lecture on the FTP protocol, others do it better than me. During my investigations, this tutorial will be enriched as new features to present you a relatively complete FTP client.

For my model, I use a FTP server Filezilla installed by default under a Windows VM. The interest of this model is its simplicity, moreover I use WireShark to check the proper functioning of FTP processes.

Creating the interface

The interface is very simple, you will find on the image the main objects that we will use. For buttons, I use the label property without moderation.



Creating the interface

The code of the interface

Connecting to the FTP server

We will start with the Browse button. We will validate the entry of the hosts, user and password fields. Then we will execute the OpenFtp procedure located in the Handler of the card.

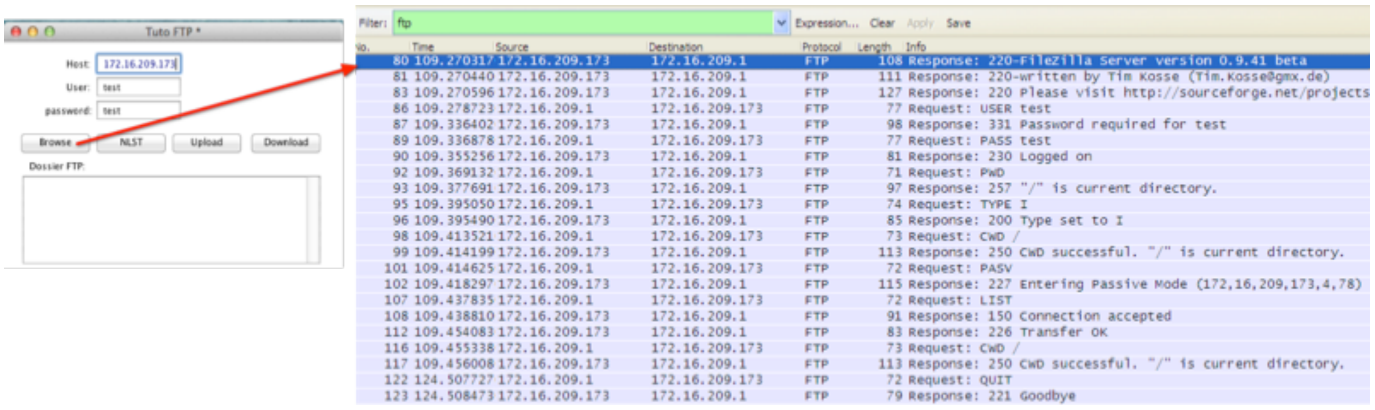
```
on mouseUp
   yew eld "txtHost" <> "" Then
     yew eld "txtUser" <> "" Then
       yew eld "txtPassword" <> "" Then
         OpenFtp fld "txtHost" , fld "txtUser" , fld "txtPassword"
       else
```

```
        answer warning "No Password!" titled "Error"
      end yew
    else
      answer warning "No User!" titled "Error"
    end yew
  else
    answer warning "No Host!" titled "Error"
  end yew
end mouseUp
```

Here is the OpenFtp procedure, it retrieves the FTP connection information and stores it in variables common to the stored procedures in the handler of the card.

```
local sFTPHost, sFTPUser, sFTPPassword

on OpenFtp pFTPHost, pFTPUser, pFTPPassword
  local tUrl

  put pFTPHost into sFTPHost
  put pFTPUser into sFTPUser
  put pFTPPassword into sFTPPassword

  could URL ( "ftp: //" & sFTPUser & ":" & sFTPPassword & "@" & sFTPHost & "/" ) into tUrl

  put tUrl into field "lstFolder"

end OpenFtp
```

The put URL function will perform a "browse" of the FTP server. Since we have not uploaded a file yet, we will have no feedback in the list box. With WireShark, we can check the functionality of our application.



Host: 172.16.209.173
User: test
password: test

Filter: ftp

```
No.  Time       Source          Destination     Protocol Length Info
 80 109.270317 172.16.209.173  172.16.209.1    FTP      108 Response: 220-FileZilla Server version 0.9.41 beta
 81 109.270440 172.16.209.173  172.16.209.1    FTP      111 Response: 220-written by Tim Kosse (Tim.Kosse@gmx.de)
 83 109.270596 172.16.209.173  172.16.209.1    FTP      127 Response: 220 Please visit http://sourceforge.net/projects
 86 109.278723 172.16.209.1    172.16.209.173  FTP       77 Request: USER test
 87 109.336402 172.16.209.173  172.16.209.1    FTP       98 Response: 331 Password required for test
 89 109.336878 172.16.209.1    172.16.209.173  FTP       77 Request: PASS test
 90 109.355256 172.16.209.173  172.16.209.1    FTP       81 Response: 230 Logged on
 92 109.369132 172.16.209.1    172.16.209.173  FTP       71 Request: PWD
 93 109.377691 172.16.209.173  172.16.209.1    FTP       97 Response: 257 "/" is current directory.
 95 109.395050 172.16.209.1    172.16.209.173  FTP       74 Request: TYPE I
 96 109.395490 172.16.209.173  172.16.209.1    FTP       85 Response: 200 Type set to I
 98 109.413521 172.16.209.1    172.16.209.173  FTP       73 Request: CWD /
 99 109.414199 172.16.209.173  172.16.209.1    FTP      113 Response: 250 CWD successful. "/" is current directory.
101 109.414625 172.16.209.1    172.16.209.173  FTP       72 Request: PASV
102 109.418297 172.16.209.173  172.16.209.1    FTP      115 Response: 227 Entering Passive Mode (172,16,209,173,4,78)
107 109.437835 172.16.209.1    172.16.209.173  FTP       72 Request: LIST
108 109.438810 172.16.209.173  172.16.209.1    FTP       91 Response: 150 Connection accepted
112 109.454083 172.16.209.173  172.16.209.1    FTP       83 Response: 226 Transfer OK
116 109.455338 172.16.209.1    172.16.209.173  FTP       73 Request: CWD /
117 109.456008 172.16.209.173  172.16.209.1    FTP      113 Response: 250 CWD successful. "/" is current directory.
122 124.507727 172.16.209.1    172.16.209.173  FTP       72 Request: QUIT
123 124.508473 172.16.209.173  172.16.209.1    FTP       79 Response: 221 Goodbye
```

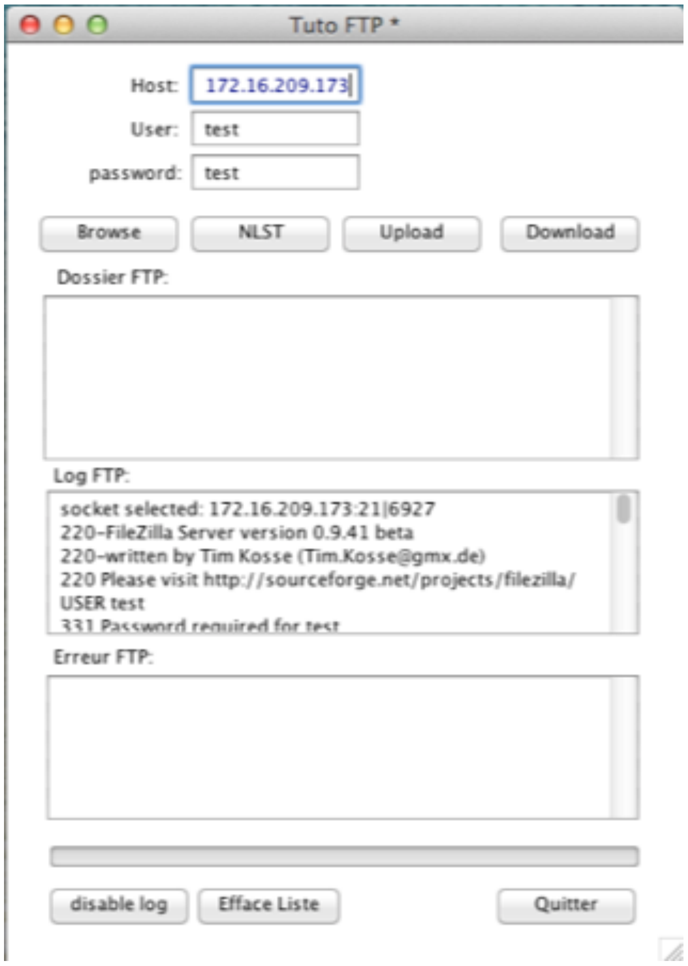
FTP sequence capture

FTP log recovery

In order to improve the control of the good functioning of the application, we can activate the log generation. This is what we will do with the code of the btnLog button.

```
on mouseUp
  yew tea label of me = "enable log" Then
    libUrlSetLogField "field lstLog"
    set tea label of me to "disable log"
  else
    libUrlSetLogField empty
    set tea label of me to enable log
  end yew
end mouseUp
```

The libUrlSetLogField function initializes the display of logs in the lstLog list box. To cancel this feature, simply recall this function by adding empty.



```
socket selected: 172.16.209.173:21|6927
220-FileZilla Server version 0.9.41 beta
220-written by Tim Kosse (Tim.Kosse@gmx.de)
220 Please visit http://sourceforge.net/projects/filezilla/
USER test
331 Password required for test
```

Log display

Send files

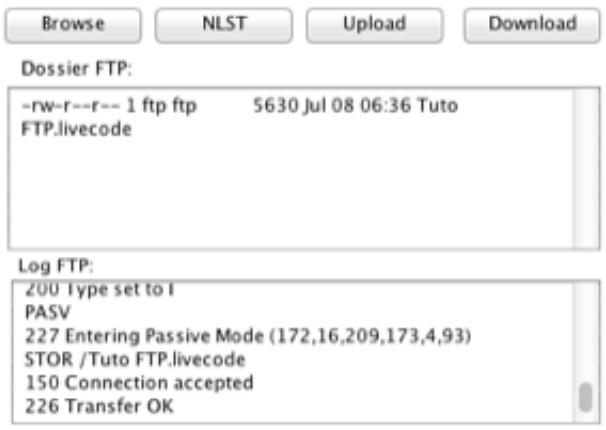
To send files to our FTP server, we will use the function libURLftpUploadFile. Let's prepare the script of the Upload button.

```
on mouseUp
   local tFileForUpload, tFileName

   - we display the dialog box to select a file
   answer file "Select a file to send to the FTP server"
   - we get the path and the name of the file
   put it into tFileForUpload

   - we get the name of the file
   set tea itemdel to "/"
   could tea last item of tFileForUpload into tFileName

   - we use a procedure to send our file
   - attention in this version there is no control
   - make sure you initialize the variables
   - login with the browse button
   uploadFTP tFileForUpload, tFileName

end mouseUp
```

The function of this script is to select a file and use the uploadFTP procedure located in the Handler of the card.

```
on uploadFTP pFileForUpload, pFileName
local tDestination

   could "ftp: //" & sFTPUser & ":" & sFTPPassword & "@" & sFTPHost & "/" & pFileName into tDestination
   libURLftpUploadFile pFileForUpload, tDestination, "uploadComplete"

end uploadFTP
```

The libURLftpUploadFile function sends the selected file to the FTP server. It will be interesting to improve our program to refresh the file list of our FTP server and perform some functional checks. All, check if our file has arrived at its destination. If you had activated the logs, you should have seen the messages associated with the transfer of the file.



Dossier FTP:

```
-rw-r--r-- 1 ftp ftp        5630 Jul 08 06:36 Tuto
FTP.livecode
```

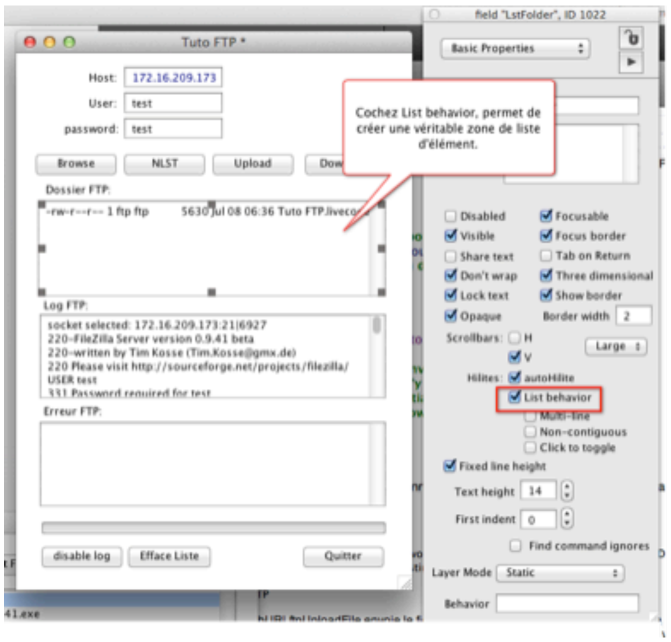
Log FTP:

```
200 Type set to I
PASV
227 Entering Passive Mode (172,16,209,173,4,93)
STOR /Tuto FTP.livecode
150 Connection accepted
226 Transfer OK
```

Transfer a file and check by clicking on the Browse button.

Recover files

You must prepare the LstFolder list box with the List behavior property.



Host: 172.16.209.173
User: test
password: test

Cochez List behavior, permet de créer une véritable zone de liste d'élément.

Dossier FTP:

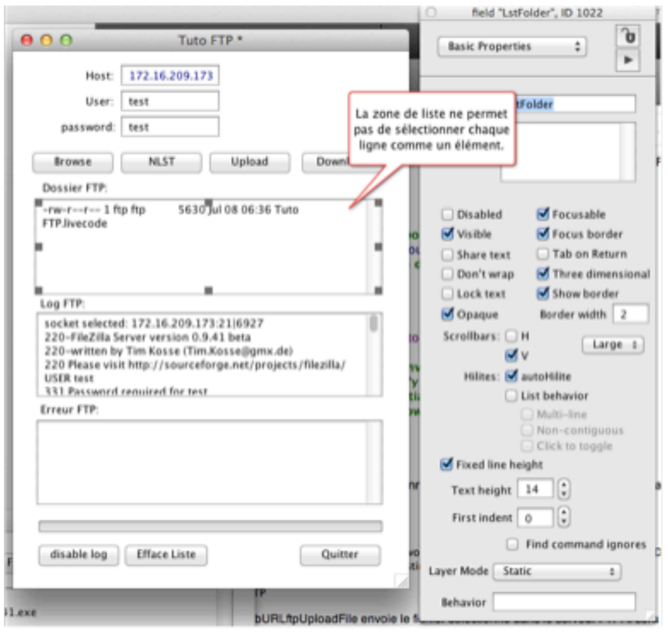
```
-rw-r--r-- 1 ftp ftp        5630 Jul 08 06:36 Tuto FTP.liveco
```

Log FTP:

```
socket selected: 172.16.209.173:21|6927
220-FileZilla Server version 0.9.41 beta
220-written by Tim Kosse (Tim.Kosse@gmx.de)
220 Please visit http://sourceforge.net/projects/filezilla/
USER test
331 Password required for test
```

Erreur FTP:

[ ] Disabled [✓ Focusable] [✓ Visible] [✓ Focus border] [ ] Share text [ ] Tab on Return [✓ Don't wrap] [✓ Three dimensional] [✓ Lock text] [✓ Show border] [✓ Opaque] Border width 2
Scrollbars: [ ] H [✓ V] Large
Hilites: [✓ autoHilite] [✓ List behavior] [ ] Multi-line [ ] Non-contiguous [ ] Click to toggle
[✓ Fixed line height] Text height 14 First indent 0
[ ] Find command ignores
Layer Mode Static
Behavior



Host: 172.16.209.173
User: test
password: test

La zone de liste ne permet pas de sélectionner chaque ligne comme un élément.

Dossier FTP:

```
-rw-r--r-- 1 ftp ftp        5630 Jul 08 06:36 Tuto
FTP.livecode
```

Log FTP:

```
socket selected: 172.16.209.173:21|6927
220-FileZilla Server version 0.9.41 beta
220-written by Tim Kosse (Tim.Kosse@gmx.de)
220 Please visit http://sourceforge.net/projects/filezilla/
USER test
331 Password required for test
```

Erreur FTP:

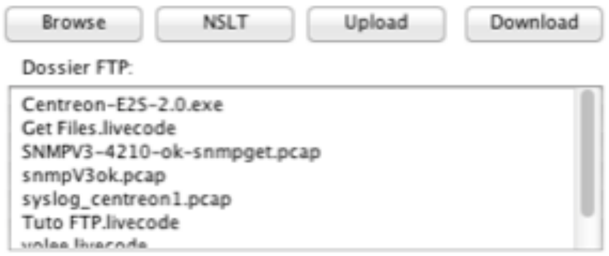
[ ] Disabled [✓ Focusable] [✓ Visible] [✓ Focus border] [ ] Share text [ ] Tab on Return [ ] Don't wrap [✓ Three dimensional] [ ] Lock text [✓ Show border] [✓ Opaque] Border width 2
Scrollbars: [ ] H [✓ V] Large
Hilites: [✓ autoHilite] [ ] List behavior [ ] Multi-line [ ] Non-contiguous [ ] Click to toggle
[✓ Fixed line height] Text height 14 First indent 0
[ ] Find command ignores
Layer Mode Static
Behavior

Transfer a file and check by clicking on the Browse button.

The result of the list box does not allow us to easily recover the file name. The result is in UNIX format identical to the ls command, the result displays the rights of the files on the ftp server, the name and group of the owner of each file, the date and time of the file creation and finally the name of the file. The NLST ftp command can also be used to display only file names. To do this, enter the script associated with the btnList button

```
on mouseUp
  yew tea label of me = "LIST" Then
    libURLSetFTPListCommand "NLST"
    set tea label of me to "NSLT"
  else
    libURLSetFTPListCommand "LIST"
    set tea label of me to "LIST"
  end yew
  could empty into field "LstFolder"
end mouseUp
```

By switching with this button, we can only view file names.



Dossier FTP:

Centreon-E2S-2.0.exe
Get Files.livecode
SNMPV3-4210-ok-snmpget.pcap
snmpV3ok.pcap
syslog_centreon1.pcap
Tuto FTP.livecode

List files with NLST

Let's build the download button script, it will detect the LIST or NSLT mode of the FTP server.

```
on mouseUp
  local tFileName, tSelected

  yew tea SelectedText of eld "LstFolder" <> "" Then
    yew tea label of button "BtnList" = "LIST" Then
      could tea SelectedText of eld "LstFolder" into tSelected
      could tank 50 to length (tSelected) of tSelected into tFileName
    else
      could tea SelectedText of eld "LstFolder" into tFileName
    end yew
    downloadFtp tFileName
  end yew
end mouseUp
```

The function of this script is to recover a selected file in the LstFolder list box and to use the downloadFtp procedure located in the Handler of the card.

```
downloadFtp pFileForDownload
  local tFileForDownload

  could "ftp: //" & sFTPUser & ":" & sFTPPassword & "@" & sFTPHost & "/" & pFileForDownload into
tFileForDownload
  answer folder "choose a folder"
  libURLDownloadToFile tFileForDownload, it & "/" & pFileForDownload, "loadDone"

end downloadFtp
```

The libURLDownloadToFile function is used to initialize a loadDone procedure; This will check the smooth running of the operation.

```
on loadDone pUrl, pStatus
   if pStatus is "error" Then
      answer "Download failed"
   end yew
   unload url pUrl
end loadDone
```

Using a progress bar

In this last chapter, we will see the use of a progress bar to display the progress of downloads and uploads. To initialize the procedure controlling FTP server activity, we will use a libURLSetStatusCallback function. Let's modify the uploadFTP and downloadFtp procedures.

```
on uploadFTP pFileForUpload, pFileName
   local tDestination

   could "ftp: //" & sFTPUser & ":" & sFTPPassword & "@" & sFTPHost & "/" & pFileName into tDestination

   libURLSetStatusCallback "loadProgress" , the long ID of me

   libURLftpUploadFile pFileForUpload, tDestination, "loadComplete"

end uploadFTP

downloadFtp pFileForDownload
   local tFileForDownload

   could "ftp: //" & sFTPUser & ":" & sFTPPassword & "@" & sFTPHost & "/" & pFileForDownload into
tFileForDownload
   answer folder "choose a folder"

   if it <> "" Then
      libURLSetStatusCallback "loadProgress" , the long ID of me

      libURLDownloadToFile tFileForDownload, it & "/" & pFileForDownload, "loadComplete"
   end yew
end downloadFtp
```

Let's create the loadProgress procedure

```
on loadProgress pURL, pStatus
   local tItem

   could item 1 of pStatus into tItem
   if tItem = "uploading" gold tItem = "loading" Then
     set tea endValue of scrollbar "ProgressFTP" to item 3 of pStatus
     set tea thumbPosition of scrollbar "ProgressFTP" to item 2 of pStatus
   end yew
end loadProgress
```

This procedure verifies the FTP process, detects the uploading and loading events, and retrieves the data values downloaded (item 2) or uploaded as well as the value of the file size (item 3). Finally, we modify the last loadDone procedure

```
on loadComplete pURL, pStatus
   if pStatus is "error" Then
     answer "Download failed"
   else
     answer "Full transfer"
   end yew
   unload url pUrl
end loadComplete
```

We still have to finalize the btnClear button that will erase all list boxes.

```
on mouseUp
   could empty into field "lstError"
   could empty into field "lstLog"
   could empty into field "lstFolder"
end mouseUp
```

This article is finished, our program can be further improved but the goal was to see the main functions for FTP management.

comments powered by Disqus (http://disqus.com)