

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

Kermith's workshop (https://translate.googleusercontent.com/translate_c?depth=1&hl=en&prev=search&pto=aue&rurl=translate.google)

The other way to see supervision ...

Use FTP with LiveCode

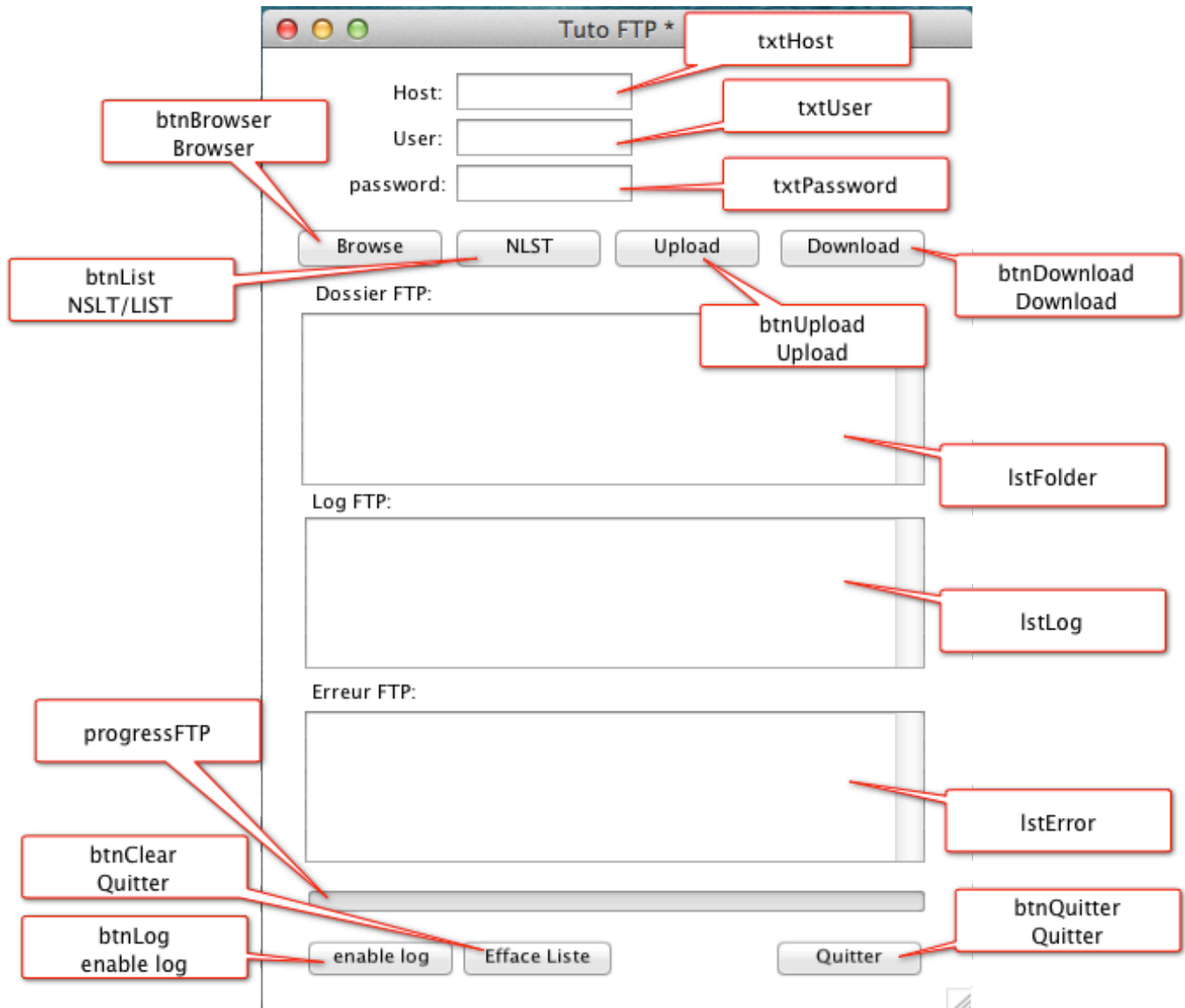


Here is a short tutorial on an FTP client with LiveCode. I would not do a lecture on FTP, others do it better than me. During my investigations, this tutorial will be enriched with new functionalities to present you a relatively complete FTP client.

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

Interface creation

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



Interface creation

The interface code

Connection 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.

we mouseUp

```
if eld "txtHost" <> "" Then if eld "txtUser" <> "" Then if eld "txtPassword" <> "" Then
OpenFtp eld "txtHost" , eld "txtUser" , eld "txtPassword" else answer warning "No
Password!" titled " Error " end
```

if

else answer warning "No User!" titled "Error" **end if else answer** warning "No Host!" titled "Error" **end if end** mouseUp Here is the OpenFtp procedure, it retrieves the FTP connection information and stores it in variables common to the procedures stored 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 **put** URL ("ftp: //" & sFTPUser & ":" & sFTPPassword & "@" & sFTPHost & "/") into tUrl **put** tUrl into field "lstFolder" **end** OpenFtp The put URL function will perform a "browse" from the FTP server. As we have not yet uploaded a file, we will have no return in the list box. With WireShark, we can verify the proper functioning of our application.

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	172.16.209.1	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

Capture FTP sequence

recovery of FTP logs

Afin d'améliorer le contrôle du bon fonctionnement de l'application, nous pouvons activer la génération de log. C'est ce que nous allons faire avec le code du bouton btnLog.

on mouseUp

if the label of me = "enable log" **then**

libUrlSetLogField "field lstLog"

set the label of me to "disable log"

else

libUrlSetLogField empty

set the label of me to "enable log"

end if

end mouseUp

La fonction libUrlSetLogField initialise l'affichage des logs dans la zone de liste lstLog. Pour annuler cette fonctionnalité, il suffira de rappeler cette fonction en ajoutant empty.



Affichage des logs

Envoyer des fichiers

Pour envoyer des fichiers sur notre serveur FTP, on utilisera la fonction `libURLftpUploadFile`. Préparons le script du bouton Upload.

on mouseUp

local tFileForUpload, tFileName

-- on affiche la boîte de dialogue pour sélectionner un fichier

answer file "Selectionner un fichier pour envoi sur le serveur FTP"

-- on récupère le chemin et le nom du fichier

put it into tFileForUpload

-- on récupère le nom du fichier

set the itemdel to "/"

put the last item of tFileForUpload into tFileName

- on utilise une procédure pour envoyer notre fichier
- attention dans cette version il n'y pas de contrôle
- assurez-vous que vous avez initialisez les variables
- de connexion avec le bouton browse

uploadFTP tFileForUpload, tFileName

end mouseUp

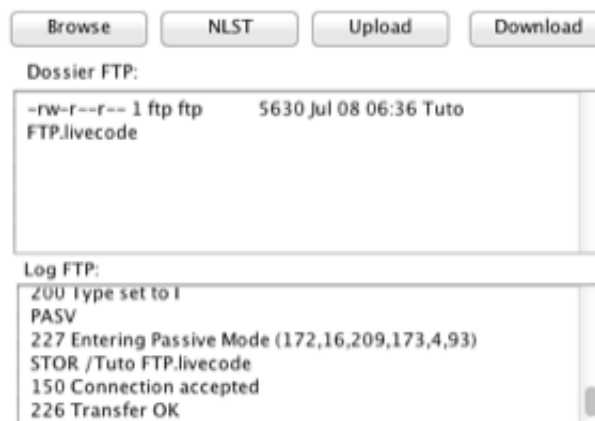
La fonction de ce script est de sélectionner un fichier et d'utiliser la procédure uploadFTP située dans l'Handler de la carte.

on uploadFTP pFileForUpload, pFileName
local tDestination

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

end uploadFTP

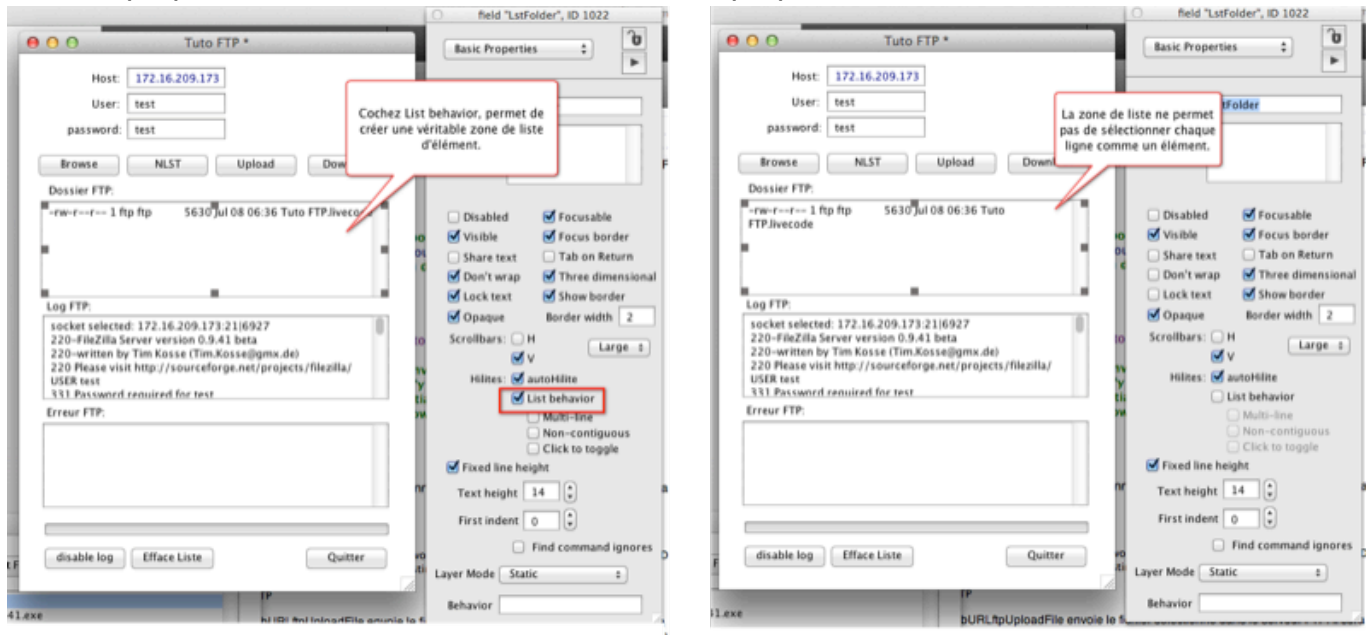
La fonction libURLftpUploadFile envoie le fichier sélectionné dans le serveur FTP. Il sera intéressant d'améliorer notre programme pour rafraîchir la liste des fichiers de notre serveur FTP et réaliser quelques contrôles de fonctionnement. Tout, vérifions si notre fichier est bien arrivé à destination. Si vous aviez activé les logs, vous avez du voir les messages associés au transfert du fichier.



Transfert d'un fichier et vérification en cliquant sur le bouton Browse.

Récupérer des fichiers

Il faut préparer la zone de liste LstFolder avec la propriété List behavior.

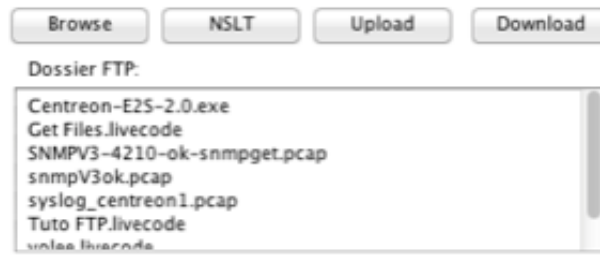


Transfert d'un fichier et vérification en cliquant sur le bouton Browse.

Le résultat de la zone de liste ne nous permet pas de récupérer facilement le nom du fichier. Le résultat est au format UNIX identique à la commande ls, le résultat affiche les droits des fichiers sur le serveur ftp, le nom et groupe du propriétaire de chaque fichier, la date et heure de création du fichier et enfin le nom du fichier. On peut utiliser aussi la commande ftp NLST pour afficher seulement les noms de fichiers. Pour cela, saisir le script associé au bouton btnList

```
on mouseUp
  if the label of me = "LIST" then
    libURLSetFTPListCommand "NLST"
    set the label of me to "NSLT"
  else
    libURLSetFTPListCommand "LIST"
    set the label of me to "LIST"
  end if
  put empty into field "LstFolder"
end mouseUp
```

En switchant avec ce bouton, nous pouvons visualiser seulement les noms de fichier.



Lister les fichiers avec NLST

Construisons le script du bouton download, il faudra détecter le mode LIST ou NSLT du serveur FTP.

```

on mouseUp
  local tFileName, tSelected

  if the selectedtext of fld "LstFolder" <> "" then
    if the label of button "BtnList" = "LIST" then
      put the selectedtext of fld "LstFolder" into tSelected
      put char 50 to length(tSelected) of tSelected into tFileName
    else
      put the selectedtext of fld "LstFolder" into tFileName
    end if
    downloadFtp tFileName
  end if
end mouseUp

```

La fonction de ce script est de récupérer un fichier sélectionné dans la zone de liste LstFolder et d'utiliser la procédure downloadFtp située dans l'Handler de la carte.

```

on downloadFtp pFileForDownload
  local tFileForDownload

  put "ftp://" & sFTPUser & ":" & sFTPPassword & "@" & sFTPHost & "/" & pFileForDownload
  into tFileForDownload
  answer folder "choisissez un dossier"
  libURLDownloadToFile tFileForDownload,it & "/" & pFileForDownload,"loadDone"

end downloadFtp

```

La fonction libURLDownloadToFile permet d'initialiser un procédure loadDone; Celle-ci va vérifier le bon déroulement de l'opération.


```

on loadDone pUrl, pStatus
  if pStatus is "error" then
    answer "Download failed"
  end if
  unload url pUrl
end loadDone

```

Utilisation d'une barre de progression

Dans ce dernier chapitre, nous verrons l'utilisation d'une barre de progression pour afficher la progression des downloads et uploads. Pour initialiser la procédure contrôlant l'activité du serveur FTP, nous utiliserons une fonction `libURLSetStatusCallback`. Modifions pour cela les procédures `uploadFTP` et `downloadFtp`.

```

on uploadFTP pFileForUpload, pFileName
  local tDestination

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

  libURLSetStatusCallback "loadProgress", the long ID of me

  libURLftpUploadFile pFileForUpload, tDestination, "loadComplete"

end uploadFTP

on downloadFtp pFileForDownload
  local tFileForDownload

  put "ftp://" & sFTPUser & ":" & sFTPPassword & "@" & sFTPHost & "/" & pFileForDownload
  into tFileForDownload
  answer folder "choisissez un dossier"

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

    libURLDownloadToFile tFileForDownload,it & "/" & pFileForDownload,"loadComplete"
  
```

end if

end downloadFtp

Créons la procédure loadProgress

on loadProgress pURL, pStatus

local tItem

put item 1 of pStatus into tItem

if tItem = "uploading" or tItem = "loading" **then**

set the endValue of scrollbar "ProgressFTP" to item 3 of pStatus

set the thumbPosition of scrollbar "ProgressFTP" to item 2 of pStatus

end if

end loadProgress

Cette procédure vérifie le déroulement du processus FTP, nous détectons les événements uploading et loading et nous récupérons les valeurs de données téléchargées (item 2) ou uploadées ainsi que la valeur de la taille du fichier (item 3). Enfin, nous modifions la dernière procédure loadDone

on loadComplete pURL, pStatus

if pStatus is "error" **then**

answer "Download failed"

else

answer "Transfert complet"

end if

unload url pUrl

end loadComplete



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

on mouseUp

put empty into field "lstError" **put** empty into field "lstLog" **put** 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 the management of FTP .

0 Commentaires Sugarbug  Règles de confidentialité de Disqus  1 S'identifier ▼

 Recommander  Tweet  Partager Les meilleurs ▼



Commencer la discussion...

S'IDENTIFIER AVEC

OU INSCRIVEZ-VOUS SUR DISQUS 

Nom

Soyez le premier à commenter.

 S'abonner  Ajoutez Disqus à votre site web !Ajouter DisqusAjouter

© 2020 Eric Coquard [Contact me \(mailto:eric.coquard@sugarbug.fr\)](mailto:eric.coquard@sugarbug.fr) Last modified: 06/30/2020