24.2.3.0 Klasse FTPClient (gb.net.curl)

Beachten Sie: FTP oder "File Transfer Protocol" (Dateiübertragungsprotokoll) ist ein unsicheres Protokoll zur Übertragung von Dateien zwischen zwei Systemen, das nur in begrenzten Fällen oder in Netzwerken verwendet werden sollte, denen Sie vertrauen.

Die Klasse stellt einen FTP-Client zur Verfügung, der das Herunterladen von Dateien von einem FTP-Server, das Hochladen von Dateien auf einen FTP-Server und das Senden von FTP-Befehlen zu einem FTP-Server ermöglicht. Die Klasse basiert auf der (Basis-)Klasse Curl in gb.net.curl.

So erzeugen Sie einen neuen FTP-Client:

```
Dim hFTPClient As FTPClient

hFTPClient = New FTPClient() As hFTPClient ' Event-Name
```

24.2.3.0.1 Eigenschaften

Die Klasse *FTPClient v*erfügt über diese Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
Async	Boolean	Gibt den Wert zurück oder legt fest, ob die FTP-Anforderung asynchron sein wird. Anfragen sind standardmäßig asynchron!
Blocking	Boolean	Gibt den Wert zurück oder legt fest, ob der Stream blockiert ist. Wenn diese Eigenschaft gesetzt ist, wird das Lesen aus dem Stream blockiert, wenn es nichts zu lesen gibt. Das Schreiben in den Stream wird blockiert, wenn z.B. der interne Systempuffer voll ist.
BufferSize	Integer	Gibt die bevorzugte Empfangspuffergröße zwischen 0 und 512 KB zurück oder stellt sie ein. Wenn der Puffer auf Null gesetzt wird, dann hat die Empfangspuffergröße ihren Standardwert, der gemäß der libcurl-Dokumentation 16 KB beträgt.
ByteOrder	Integer	Gibt die zum Lesen oder Schreiben von Binärdaten in den Datenstrom verwendete Byte-Reihenfolge zurück oder legt diese fest. Die Eigenschaft kann die folgenden Werte haben: gb.BigEndian oder gb.LittleEndian.
Debug	Boolean	Gibt den Debugging-Modus zurück oder stellt ihn ein. Wenn die Eigenschaft beispielsweise auf True gesetzt ist, so druckt ein FTPClient-Objekt alle an den FTP-Server gesendeten Befehle in die Konsole (der IDE).
Downloaded	Long	Gibt die Anzahl der Bytes zurück, die vom FTP-Server bereits heruntergeladen wurden. Die Eigenschaft kann nur gelesen werden.
TotalDownloaded	Long	Gibt die Gesamtzahl der Bytes zurück, von denen erwartet wird, dass sie heruntergeladen werden. Die Eigenschaft kann nur gelesen werden.
Uploaded	Long	Gibt die Anzahl der Bytes zurück, die bereits hochgeladen wurden. Die Eigenschaft kann nur gelesen werden.
TotalUploaded	Long	Gibt die Gesamtzahl der Bytes zurück, von denen erwartet wird, dass sie hochgeladen werden. Die Eigenschaft kann nur gelesen werden.
User	String	Gibt den für die Autorisierung verwendeten FTP-Benutzernamen zurück oder legt ihn fest.
Password	String	Gibt das für die Autorisierung verwendete FTP-Passwort zurück oder legt es fest.
EndOfFile	Boolean	Diese Eigenschaft signalisiert, ob mit der letzten Verwendung von LINE INPUT das Ende der Datei erreicht wurde, anstatt eine ganze Zeile mit einem Zeilenende-Zeichen zu lesen. Um zu überprüfen, ob das Dateiende in irgendeinem Kontext erreicht wurde, verwenden Sie die EOF-Funktion oder die EOF-Eigenschaft.
EndOfLine	Integer	Gibt das vom aktuellen Stream verwendete Zeilenumbruchstrennzeichen zurück oder setzt es. Die möglichen Werte sind: gb.Unix für Zeilen, die durch Chr$(10) getrennt sind, gb.Windows für Zeilen, die durch Chr$(13) & Chr$(10) getrennt sind oder gb.Mac für Zeilen, die durch Chr$(13) getrennt sind. Der Wert dieser Eigenschaft wird von LINE INPUT, PRINT und der Stream.Lines-

Eigenschaft	Datentyp	Beschreibung
		Eigenschaft verwendet. Tatsächlich erlaubt es gb.Unix, sowohl Unix- als auch Windows-Zeilenformate zu lesen. Aber es schreibt nur das Unix-Format!
Eof	Boolean	Gibt True zurück, wenn ein Stream sein Ende erreicht hat. Das ist ein Äquivalent der EOF-Funktion.
Lines	Stream.Lines	Gibt ein (virtuelles) Objekt zurück, mit dem Sie einen Datenstrom Zeile für Zeile aufzählen und auswerten können.
Timeout	Integer	Gibt das Client-Timeout in Sekunden zurück oder legt es fest. Die Verbindungsaufnahme vom Client zum FTP-Server wird abgebrochen, wenn sie mehr als die angegebene Anzahl von Sekunden dauert. Wenn die Zeitüberschreitung Null ist, wird keine Zeitüberschreitung definiert. Vergessen Sie auf keinen Fall, diese Eigenschaft zu setzen, wenn es sich um eine *synchrone* Anfrage handelt, sonst kann der FTP-Client ewig auf die Verbindungsaufnahme zum FTP-Server warten!
URL	String	Gibt den URL der Ressource zurück, die heruntergeladen werden soll oder legt ihn fest.
Proxy	Curl.Proxy	Gibt ein (virtuelles) Objekt zurück, das zur Definition von Proxy-Parametern verwendet wird (Auth, Host, Password, Type, User). Wenn kein Proxy definiert ist, so wird der Standard-Proxy verwendet.
NoEPSV	Boolean	Gibt den Wert zurück, ob der FTP-Client bei passiven FTP-Downloads EPSV verwenden kann oder nicht oder Sie legen den Wert fest. Wenn diese Eigenschaft gesetzt ist, wird kein EPSV verwendet, sondern nur PASV. Standardmäßig ist diese Eigenschaft nicht gesetzt, und EPSV ist immer erlaubt. Ist der FTP-Server ein IPv6-Host, so hat das Festlegen dieser Eigenschaft keine Auswirkung.
TargetFile	String	Gibt die Zieldatei zurück, die nur für Download-Operationen verwendet wird oder legt den Pfad für die Zieldatei fest.
Tag	Variant	Gibt die mit dem Stream verknüpfte Tag-Eigenschaft zurück oder legt sie fest. Diese Eigenschaft ist für den Programmierer gedacht und wird von der Komponente nie verwendet. Sie kann jeden beliebigen Variant-Wert annehmen.
Status	Integer	Gibt den Status des FTP-Clients zurück. Er kann drei Werte annehmen.
SSL	Curl.SSL	Diese (virtuelle) Eigenschaft erlaubt es, die SSL-Eigenschaften der zu Grunde liegenden CURL-Verbindung mit den beiden Eigenschaften VerifyHost (Standard) und VerifyPeer zu definieren.
Handle	Integer	Gibt den Systemdatei-Deskriptor zurück, der mit diesem Stream verknüpft ist.
IsTerm	Boolean	Gibt den Wert True zurück, wenn ein Stream mit einem Terminal assoziiert ist.
Term	Stream.Term	Rückgabe eines (virtuellen) Objekts, das die Verwaltung des mit dem Stream verbundenen Terminals ermöglicht.
ErrorText	String	Gibt die Fehlerzeichenfolge zurück, die mit dem von der Curl-Bibliothek zurückgegebenen Fehlercode verknüpft ist. Wenn der Status einen positiven Wert besitzt, dann wird eine leere Zeichenkette zurückgegeben.

Tabelle 24.2.3.0.1 : Eigenschaften der Klasse FTPClient

Beispiel für die Eigenschaft Lines:

```
Dim hFile As File
Dim sLine As String

hFile = Open "/var/log/syslog"
For Each sLine In hFile.Lines
  If InStr(sLine, "[drm]") Then Print sLine
Next
```

Werte der Status-Eigenschaft, die nur gelesen werden können:

```
Net.Inactive        (0)    Der FTP-Client ist inaktiv.
Net.ReceivingData   (4)    Der FTP-Client empfängt Daten aus dem Netzwerk.
Net.Connecting      (6)    Der FTP-Client verbindet sich mit dem FTP-Server.
```

Trat ein Fehler auf, dann ist der Status-Wert negativ. Der Wert des Status ist eigentlich -1000 vermindert um den libcurl-Fehlercode. Die meisten dieser Fehlercodes haben eine entsprechende Konstante in der Net-Klasse. Um zu wissen, was ein Fehlercode genau bedeutet, müssen Sie sich die libcurl-Fehlercodeliste unter https://curl.se/libcurl/c/libcurl-errors.html oder im Gambas-Club unter https://www.gambas-club.de/viewtopic.php?f=3&t=5580&p=13190&hilit=Curl+error#p13190 ansehen.

FTPClient hat in Abhängigkeit von der Async-Eigenschaft zwei Möglichkeiten zu arbeiten. Wenn Async den Wert TRUE hat, dann hält das Programm an, bis der angeforderte Vorgang abgeschlossen ist. Dies garantiert, dass das gewünschte Ergebnis fertiggestellt ist, bevor das Programm fortgesetzt wird. Sie erhalten aber keine Informationen während des laufenden Prozesses. Hat die Eigenschaft den Wert FALSE, so fordert das Programm die FTP-Operation vom Server an und setzt sie fort. In diesem Fall können Sie auf den Status des Prozesses zugreifen.

Für die Programmentwicklung ist es von Vorteil, wenn Sie die Debug-Eigenschaft auf True setzen. Dann sehen Sie stets das komplette Verbindungsprotokoll in der Konsole der IDE – so wie im folgenden Beispiel:

```
*   Trying 109.xyz.140.40:21...
* TCP_NODELAY set
* Connected to gambas-buch.de (109.xyz.140.40) port 21 (#0)
< 220 FTP Server ready.
> USER ftp_username
< 331 Password required for ftp_username
> PASS ftp_password
< 230 Zugriff gewaehrt fuer ftp_username
> PWD
< 257 "/" is the current directory
* Entry path is '/'
> CWD base
* ftp_perform ends with SECONDARY: 0
< 250 CWD command successful
> CWD dwiki
< 250 CWD command successful
> CWD data
< 250 CWD command successful
> CWD k3
< 250 CWD command successful
> CWD k3.1
< 250 CWD command successful
> EPSV
* Connect data stream passively
< 229 Entering Extended Passive Mode (|||65089|)
*   Trying 109.xyz.140.40:65089...
* TCP_NODELAY set
* Connecting to 109.xyz.140.40 (109.xyz.140.40) port 65089
* Connected to gambas-buch.de (109.xyz.140.40) port 21 (#0)
> TYPE I
< 200 Type set to I
> STOR start.txt
< 150 Opening BINARY mode data connection for start.txt
* We are completely uploaded and fine
* Remembering we are in dir "base/dwiki/data/k3/k3.1/"
< 226 Transfer complete
* Connection #0 to host gambas-buch.de left intact
```

24.2.3.0.2 Methoden

Die Klasse *FTPClient* verfügt über folgende Methoden:

Methode	Rückgabetyp	Beschreibung
Begin( )	-	Beginn der Pufferung der in den Stream geschriebenen Daten, so dass alles beim Aufruf der Send-Methode auf einmal gesendet wird.
Send( )	-	Alle Daten werden seit dem letzten Aufruf der Methode 'Begin' auf einen Schlag gesendet.
Stop( )	-	Bricht den aktuellen Befehl ab.
Peek( )	String	Gibt den Inhalt des internen Stream-Puffers zurück. Die Daten werden nicht aus dem Puffer entfernt, so dass sie für das nächste Lesen zur Verfügung stehen.

Methode	Rückgabetyp	Beschreibung
Watch (Mode As Integer, Watch As Boolean)	-	Starten oder stoppen Sie die Überwachung des Streamdatei-Deskriptors zum Lesen oder Schreiben, nachdem er geöffnet wurde. Modus ist der Überwachungstyp: (1) gb.Read zum Beobachten beim Lesen und (2) gb.Write zum Beobachten beim Schreiben. Setzen Sie Watch auf TRUE, um die Überwachung zu aktivieren oder auf False, um sie zu deaktivieren.
Close( )	-	Schließt den Stream.
ReadLine ([Escape As String])	String	Lesen einer Textzeile aus dem Datenstrom, wie bei der Anweisung LINE INPUT. Wenn der optionale Parameter 'Escape' angegeben wird, dann werden Zeilenumbrüche zwischen zwei Escape-Zeichen ignoriert. Diese Methode ist beim Lesen von CSV-Dateien sehr nützlich.
Get ([TargetFile As String]	-	Lädt eine Datei vom FTP-Server. Wenn die Eigenschaft *TargetFile* angegeben ist, wird die Datei automatisch im Pfad von TargetFile gespeichert. Bevor Sie diese Methode verwenden, müssen Sie die URL-Eigenschaft mit dem gewünschten Hostnamen und dem Pfad zum abzurufenden Dokument füllen.
Put (LocalFile As String)	-	Senden Sie eine Datei auf den FTP-Server. Der Parameter *LocalFile* enthält den Pfad der zu sendenden Datei des FTP-Client-Systems.
Exec (Commands As String[])	-	Führt bestimmte FTP-Befehle direkt aus. Der Parameter 'Commands' ist ein String-Array, in dem jedes Element einen einzelnen FTP-Befehl repräsentiert.

Tabelle 24.2.3.0.2 : Methoden der Klasse FTPClient

24.2.3.0.3 Ereignisse

Die Klasse *FTPClient* verfügt über 6 Ereignisse:

Methode	Beschreibung
Connect( )	Das Ereignis wird ausgelöst, wenn eine Verbindung vom FTP-Client zum FTP-Server hergestellt wurde.
Read( )	Das Ereignis wird ausgelöst, wenn (Nutz-)Daten empfangen werden.
Progress( )	Dieses Ereignis wird periodisch ausgelöst, so lange etwas heruntergeladen oder hochgeladen wird (Nutz-Daten, FTP-Befehle, Quittungen, Fehlermeldungen).
Finished( )	Das Ereignis wird ausgelöst, wenn ein Download oder ein Upload ohne Fehler beendet wird.
Error( )	Das Ereignis wird ausgelöst, wenn ein Fehler (von der CURL-Bibliothek) zurückgegeben wurde.
Cancel( )	Dieses Ereignis wird ausgelöst, wenn eine Anfrage storniert wurde.

Tabelle 24.2.3.0.3 : Ereignisse der Klasse FTPClient

24.2.3.0.4 Beispiele

Beispiel 1: Put()-Methode (Upload)

Der folgende Quelltext-Ausschnitt stammt aus einem Programm zum Einfügen von einzelnen Seiten im DokuWiki-Format in das Gambas-Buch auf gambas-buch.de:

```
Public hFTPClient As FtpClient

Public Sub Form_Open()

    MAdditional.CheckNetwork()

    hFTPClient = New FtpClient As "hFTPClient"

    hFTPClient.User = "ftp123"
    hFTPClient.Password = "pass456"
```

```
...

End

Public Sub btnFileUpLoad_Click()

    Dim bFTPClientStatus As Boolean = False

    FileUpload()

    Repeat
      Wait 0.001
      lblFTPStatus.Text = "FTP-Status = " & hFTPClient.Status
    Until bFTPClientStatus = bConnected

End

Public Sub hFTPClient_Connect()
    lblFTPStatus.Text = "FTP status: Connection to FTP server established!"
    bConnected = True
End

Public Sub hFTPClient_Finished()

    ProgressBar1.Value = 1
    btnFileUpLoad.Picture = Picture["leds/led_green_16x16.png"]
    lblFTPStatus.Text = "FTP status: Transfer complete"

    If bConnected = True Then hFTPClient.Close()

End

Public Sub hFTPClient_Progress()

'-- (Relative) Display of the uploaded bytes in relation to the size of the transferred file.
    If hFTPClient.TotalUploaded > 0 Then
       ProgressBar1.Value = hFTPClient.Uploaded / hFTPclient.TotalUploaded
    Endif

End

Public Sub hFTPClient_Error()
  Message.Error("<b><font color='DarkRed'>FTP:  ERROR</b></font><hr>" & hFTPClient.ErrorText)
End

Private Sub FileUpload()

    hFTPClient.URL = txbTargetURL.Text
'-- The Argument in the Put(sArg) method is the path of the file to be uploaded *on the local PC*
'-- Upload the selected DokuWiki source text file:
    hFTPClient.Put(txbSourcePath.Text)

End
```

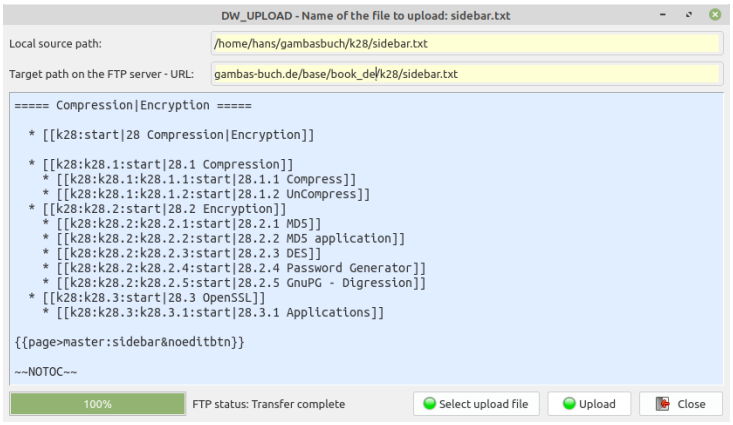

Abbildung 24.2.3.0.1: Der Upload einer Datei war erfolgreich

Beachten Sie:
Die Namen der LocalSource-Datei und der Target-Datei müssen nicht notwendigerweise gleich sein!
Das gilt für den Upload genauso wie für den Download!

Fehler beim Upload von einzelnen Seiten werden vom Programm abgefangen und angezeigt:

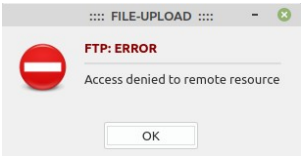

Abbildung 24.2.3.0.2: Fehler beim Upload

Beispiel 2: Get()-Methode (Download)

Das Beispiel wurde gewählt, weil es eine Besonderheit präsentiert. Normalerweise geben Sie als Argument für die Get()-Methode einen kompletten Pfad für die Datei auf dem FTP-Server an, die vom FTP-Server in ein ausgewähltes Verzeichnis auf dem FTP-Client-System herunter geladen werden soll. Geben Sie jedoch einen kompletten Pfad für ein Verzeichnis auf dem FTP-Server an – der mit / enden muss – dann wird der Inhalt dieses Verzeichnisses zurückgegeben:

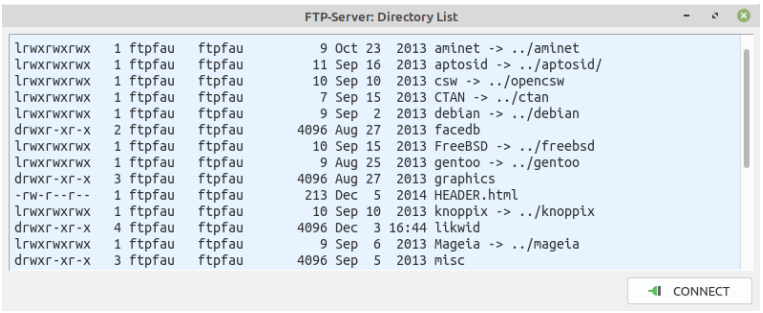

Abbildung 24.2.3.0.3: Inhalt eines ausgewählten Verzeichnisses auf dem FTP-Server

Der Quelltext für das Auslesen des Inhaltes eines ausgewählten Verzeichnisses auf dem FTP-Server ist erfreulich kurz und wird deshalb komplett angegeben. Sie können das Projekt selbst erproben, da die FTP-Konto-Daten für den öffentlichen FTP-Server mit angegeben werden:

```
' Gambas class file

Public hFTPClient As FtpClient

Public Sub Form_Open()

    FMain.Caption = "FTP-Server: Directory List"

    hFTPClient = New FtpClient As "hFTPClient"

    hFTPClient.User = "anonymous"      '-- Required: `anonymous`
    hFTPClient.Password = "wer@ist.da" '-- Required: An arbitrary, but syntactically correct email address

    hFTPClient.Debug = True

End

Public Sub hFTPClient_Error()
    txaOutput.Insert("FTP-Error-Text => " & hFTPClient.ErrorText & gb.NewLine)
    txaOutput.Insert("FTP Status: " & hFTPClient.Status & gb.NewLine)
End

Public Sub btnGetList_Click()

    Dim aList As String[]
    Dim iCount As Integer

    aList = GetList("ftp.uni-erlangen.de/pub/")
'-- aList = GetList("ftp.uni-erlangen.de/pub/ubuntu/")

    txaOutput.Clear()

    For iCount = 0 To aList.Max
      If iCount < aList.Max Then
         txaOutput.Insert(aList[iCount] & gb.NewLine)
      Else
         txaOutput.Insert(aList[iCount])
      Endif
```

```
    Next

End

Private Function GetList(argURL As String) As String[]

    Dim sTempFileName As String = Temp()
    Dim aList As String[]

    hFTPClient.URL = argURL
    hFTPClient.Async = False
    hFTPClient.Timeout = 3

    hFTPClient.Get(sTempFileName)

    aList = Split(File.Load(sTempFileName), gb.NewLine)

    Return aList

End
```

Die Hauptlast trägt die Funktion GetList(argURL). Als (Download-)Datei wird eine temporäre Datei benutzt, da deren Inhalt interessiert, der im Rückgabewert der Funktion gespeichert wird.

Beispiel 3: Exec()-Methode

In der praktischen Arbeit hat sich die Exec()-Methode u.a. zum Anlegen und zum Löschen eines Verzeichnisses auf dem FTP-Server bewährt:

```
  Inc Application.Busy
    hFTPClient.Async = False
    hFTPClient.Exec(["MKD " & "BaseDirectoryPath/DirectoryName_Create"])
  Dec Application.Busy
```

```
  Inc Application.Busy
    hFTPClient.Async = False
    hFTPClient.Exec(["RMD " & "BaseDirectoryPath/DirectoryName_Delete"])
  Dec Application.Busy
```

Eine Übersicht aller auf dem FTP-Server implementierten FTP-Befehle erhalten Sie mit folgendem Quelltext-Abschnitt. Berücksichtigen Sie, dass die Ausgabe nur dann in der Konsole der IDE angezeigt wird, wenn die Debug-Eigenschaft den Wert True hat:

```
    hFTPClient = New FtpClient As "hFTPClient"

    hFTPClient.User = "ftp123"
    hFTPClient.Password = "pass345"
    hFTPClient.URL = "gambas-buch.de"
    hFTPClient.Async = False
    hFTPClient.Timeout = 3

    hFTPClient.Debug = True
    hFTPClient.Exec(["HELP"])
```

Das ist die Ausgabe für den verwendeten FTP-Server (vsFTPd) des Autors:

```
> HELP
* ftp_perform ends with SECONDARY: 0
< 214-The following commands are recognized (* =>'s unimplemented):
<  CWD     XCWD    CDUP    XCUP    SMNT*   QUIT    PORT    PASV
<  EPRT    EPSV    ALLO*   RNFR    RNTO    DELE    MDTM    RMD
<  XRMD    MKD     XMKD    PWD     XPWD    SIZE    SYST    HELP
<  NOOP    FEAT    OPTS    HOST    CLNT    AUTH    CCC*    CONF*
<  ENC*    MIC*    PBSZ    PROT    TYPE    STRU    MODE    RETR
<  STOR    STOU    APPE    REST    ABOR    USER    PASS    ACCT*
<  REIN*   LIST    NLST    STAT    SITE    MLSD    MLST
```

Eine einzelne Datei löschen Sie auf dem FTP-Server so:

```
  Inc Application.Busy
    hFTPClient.Async = False
    hFTPClient.Exec(["DELE " & "BaseDirectoryPath/FileName"])
  Dec Application.Busy
```

```
If hFTPClient.ErrorText Then
   Message.Error("ERROR:<br>" & hFTPClient.ErrorText)
Endif
```

24.2.3.0.5 FTP-Konten

Es ist eine gute Idee, die erforderlichen FTP-Konto-Daten für ein FTP-Programm entweder durch einen einzelnen Dialog oder durch einen FTP-Konto-Manager mit mehreren Dialogen bereitzustellen. Anregungen für geeignete Dialoge oder für Konto-Manager finden Sie bereits im Gambas-Buch:

```
https://gambas-buch.de/doku.php?id=k12:k12.4:k12.4.4:k12.4.4.1:start
https://gambas-buch.de/doku.php?id=k24:k24.3:k24.3.4:start
```

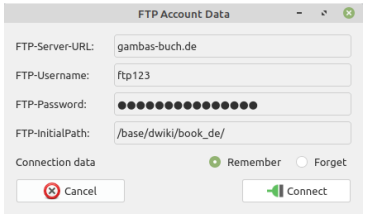

Abbildung 24.2.3.0.4: Konto-Dialog für einen FTP-Server

Wenn Sie mehrere FTP-Server nutzen, dann wird ein FTP-Konto-Manager die richtige Wahl sein:

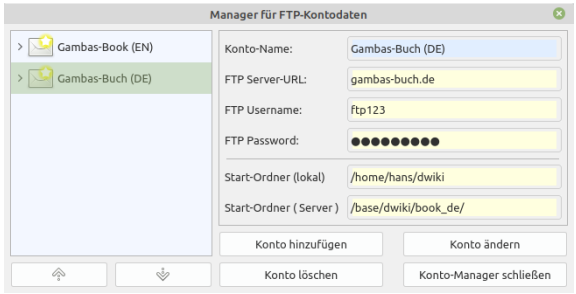

Abbildung 24.2.3.0.5: FTP-Konto-Manager

24.2.3.0.6 Exkurs: Verwendung des Befehlszeilentools Curl mit FTP, SFTP und FTPS

Eine kurze, auf das Wesentliche reduzierte Einführung in die Verwendung von Curl gibt es auf dieser Webseite (Stand: 24.01.2022):

```
http://www.mukeshkumar.net/articles/curl/how-to-use-curl-command-line-tool-with-ftp-and-sftp
```

24.2.3.0.7 Projekt FTP-Client

Im nächsten Kapitel wird Ihnen ein Projekt für einen FTP-Client auf der Basis der Klasse FTPCLient vorgestellt.