24.2.4.0 Klasse HTTPClient

Diese Klasse stellt einen HTTP-Client zur Verfügung, der Anfragen an einen HTTP-Server sendet und dessen Antwort empfängt.

So erzeugen Sie einen neuen HTTP-Client:

```
Dim hHTTPClient As HTTPClient
hHTTPClient = New HTTPClient() As "hHTTPClient"  ' → Ereignisname (optional)
```

24.2.4.0.1 Ausgewählte Eigenschaften

Die Klasse HTTPClient verfügt über diese Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
Auth	Integer	Gibt den HTTP-Authentifizierungsmodus zurück oder der Modus gesetzt. Es kann eine der folgenden Konstanten sein: Net.AuthNone für keine Authentifizierung (Standard), Net.AuthBasic für Basis-Authentifizierung oder Net.AuthNTLM für NTLM-Authentifizierung.
Async	Boolean	Gibt zurück oder legt fest, ob die FTP/HTTP-Anforderung asynchron abgewickelt wird. Anfragen sind standardmäßig asynchron.
Blocking	Boolean	Gibt 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 auf Null gesetzt, 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.
Code	Integer	Gibt den vom Server gesendeten HTTP-Status-Code zurück.
CookiesFile	String	Gibt die Datei zurück, die zum Lesen und Speichern von Cookies verwendet werden soll oder legt sie fest. Wenn keine Datei gesetzt ist, sind Cookies nicht persistent. Cookies werden nur dann gespeichert, wenn auch die Eigenschaft 'UpdateCookies' gesetzt ist.
Debug	Boolean	Gibt den Wert zurück oder legt den Debugging-Modusfest. Wenn diese Eigenschaft beispielsweise gesetzt ist, druckt ein Client-Objekt alle an den Server gesendeten Befehle in die Konsole der IDE.
Encoding	String	Gibt den HTTP-Header "Accept-Encoding" zurück oder setzt ihn.
Headers	String[ ]	Gibt die HTTP-Header in einem String-Array zurück.
Reason	String	Gibt den HTTP-Reason-String zurück. Dies ist derjenige String, der auf den *Fehler-Code* in einer HTTP-Antwort folgt.
Redirect	Boolean	Gibt den Wert zurück oder legt ihn fest, ob der HTTPClient HTTP-Umleitungen folgen muss. Standardmäßig werden Umleitungen *nicht* befolgt.
UpdateCookies	Boolean	Gibt den Wert zurück oder legt fest, ob Cookies aktualisiert werden müssen. Der Standardwert ist False. Die Eigenschaft CookiesFile muss (vorher) gesetzt sein, sonst werden Cookies nicht gespeichert.
UserAgent	String	Gibt den für die Anforderung verwendeten UserAgent zurück oder legt ihn fest. Der aktuell verwendete UserAgent ist 'Gambas/3.17 (gb.net.curl; Linux)'.
Downloaded	Long	Gibt die Byte-Anzahl zurück, die vom Server bereits heruntergeladen wurden.
TotalDownloaded	Long	Gibt die Gesamtzahl der Bytes zurück, von denen erwartet wird, dass sie heruntergeladen werden.
Uploaded	Long	Gibt die Anzahl der Bytes zurück, die bereits hochgeladen wurden.
TotalUploaded	Long	Gibt die Gesamtzahl der Bytes zurück, von denen erwartet wird, dass sie hochgeladen werden.

Eigenschaft	Datentyp	Beschreibung
ErrorText	String	Gibt die Fehlerzeichenfolge zurück, die mit dem von der Curl-Bibliothek zurückgegebenen Fehlercode verknüpft ist.
User	String	Gibt den für die Autorisierung verwendeten Benutzer zurück oder legt ihn fest.
Password	String	Gibt das für die Autorisierung verwendete Kennwort zurück oder legt es fest.
TargetFile	String	Gibt die Zieldatei, die für Download-Operationen verwendet wird, zurück oder legt sie fest. Dies ist ein Äquivalent des optionalen TargetFile-Arguments der Download-Methode der Klasse HTTPClient.
Timeout	Integer	Gibt das Client-Timeout in Sekunden zurück oder legt es fest. Die Anforderung wird abgebrochen, wenn sie mehr als die angegebene Anzahl von Sekunden dauert. Wenn der Wert Null ist, ist keine Zeitüberschreitung definiert. Vergessen Sie nicht, diese Eigenschaft zu setzen, wenn es sich um eine *synchrone* Anfrage handelt, da sonst kann der Client ewig auf den Server wartet.
Status	Integer	Gibt den Status des Clients zurück. Er kann einer der folgenden Werte sein: Net.inaktiv: Der Client ist inaktiv. Net.ReceivingData: Der Client empfängt Daten aus dem Netzwerk. Net.Connecting: Der Client verbindet sich mit dem Server. Der Status hat einen negativen Wert, wenn ein Fehler aufgetreten ist. Der Wert des Status ist eigentlich -1000 minus dem libcurl-Fehlercode. Die meisten dieser Fehlercodes haben eine entsprechende Konstante in der Klasse Net. Um zu wissen, was ein Fehlercode genau bedeutet, müssen Sie sich die libcurl-Fehlercodeliste ansehen.

Tabelle 24.2.4.0.1 : Eigenschaften der Klasse HTTPClient

24.2.4.0.2 Methoden

Die Klasse HTTPClient verfügt über die statische Methode DownLoad(...):

```
Static Function Download ( URL As String [ , Headers As String[] ] ) As String
- URL ist die Adresse
- Headers ist ein optionales String-Array der verwendeten Header
```

Die Klasse HTTPClient besitzt diese ausgewählten Methoden:

Methode	Beschreibung
Head ([Headers As String[]])	Diese Methode führt einen Aufruf an den HTTP-Server mit der Standardmethode 'HEAD' durch. Vor der Verwendung dieser Methode müssen Sie die URL-Eigenschaft mit dem gewünschten Hostnamen und dem abzurufenden Dokument füllen. 'Headers' ist ein optionales String-Array von HTTP-Headern, die mit der Anfrage an den Server gesendet werden.
Post (ContentType As String, Data As String [, Headers As String[], TargetFile As String])	Diese Methode führt einen Aufruf an den http-Server mit der Standardmethode 'POST' durch. Bevor Sie sie verwenden können, müssen Sie die URL-Eigenschaft mit dem gewünschten Hostnamen und dem abzurufenden Dokument füllen.
Get ([Headers As String[], TargetFile As String])	Diese Methode führt eine Anfrage an den HTTP-Server mit der Standardmethode 'GET' durch. Vor der Verwendung dieser Methode müssen Sie die URL-Eigenschaft mit dem gewünschten Hostnamen und dem abzurufenden Dokument füllen.
CopyFrom (HttpClient As Source)	Füllt das HttpClient-Objekt mit den Eigenschaften eines anderen HttpClient-Objekts. Der Parameter 'HttpClient' gibt das Quell-HttpClient-Objekt zurück.
Put (ContentType As String, Data As String [, Headers As String[], TargetFile As String])	Sendet Daten mit der "PUT"-Methode an eine bestimmte URL.
PostFile (ContentType As String, Path As String [, Headers As String[], TargetFile As String])	Sie können eine Datei über eine POST-Anforderung senden.

Tabelle 24.2.4.0.2 : Methoden der Klasse HTTPClient

Hinweise

- Post (ContentType As String, Data As String [, Headers As String[], TargetFile As String]):
  ContentType ist der MIME-Typ der Daten. *Data* sind die zu setzenden Daten. *Headers* ist ein optionales String-Array von HTTP-Headern. TargetFile ist eine optionale Datei, in die die Antwort geschrieben wird. Wenn TargetFile nicht angegeben wird, werden die vom Server empfangenen Daten im Speicher abgelegt. Sie können auf die Daten mit den Standard-Stream-Methoden oder der Peek-Methode auf das Dokument zugreifen. Wenn TargetFile angegeben wird, werden die vom Server empfangenen Daten in der angegebenen Datei gespeichert und sind im internen Speicherpuffer nicht verfügbar. Wenn Sie Formulardaten übermitteln möchten, verwenden Sie 'application/x-www-form-urlencoded' als ContentType-Header der Anforderung. Anschließend formatieren Sie den Parameter 'Data' mit einer Zeichenfolge nach der folgenden Syntax: Key1=Wert1&Schlüssel2=Wert2&... .
- Get ([Headers As String[], TargetFile As String]):
  Header ist ein optionales String-Array von HTTP-Headern, die mit der Anfrage an den Server gesendet werden. *TargetFile* ist eine optionale Datei, in die die Antwort geschrieben wird. Wenn TargetFile nicht angegeben ist, werden die vom Server empfangenen Daten im Speicher abgelegt. Sie können auf die Daten mit Standard-Stream-Methoden oder der Peek-Methode zugreifen. Wenn TargetFile angegeben wird, werden die vom Server empfangenen Daten in der angegebenen Datei gespeichert und sind danach im internen Speicherpuffer nicht mehr verfügbar.
- Put (ContentType As String, Data As String [, Headers As String[ ], TargetFile As String]):
  ContentType ist der MIME-Typ der Daten. *Data* sind die zu setzenden Daten. *Headers* ist ein optionales String-Array von HTTP-Headern. *TargetFile* ist der Pfad einer optionalen Datei, in die die Antwort geschrieben wird.
- PostFile (ContentType As String, Path As String [, Headers As String[ ], TargetFile As String]):
  ContentType ist der MIME-Typ der Datei. *Path* ist der Pfad der zu sendenden Datei. *Headers* ist ein optionales String-Array von HTTP-Headern. *TargetFile* ist eine optionale Datei, in die die Antwort geschrieben wird. Wenn TargetFile nicht angegeben ist, werden die vom Server empfangenen Daten im Speicher abgelegt. Sie können mit Standard-Stream-Methoden oder der Peek-Methode auf das Dokument zugreifen. Wenn TargetFile jedoch angegeben wird, werden die vom Server empfangenen Daten in der angegebenen Datei gespeichert und sind im internen Speicherpuffer danach nicht mehr verfügbar.

24.2.4.0.3 Ereignisse

Die Klasse HTTPClient verfügt über diese Ereignisse:

Methode	Beschreibung
Cancel ( )	Dieses Ereignis wird ausgelöst, wenn ein Request abgebrochen wurde.
Connect ( )	Das Ereignis wird ausgelöst, wenn die Verbindung hergestellt ist.
Error ( )	Das Ereignis wird ausgelöst, wenn ein Fehler (von der CURL-Bibliothek) zurückgegeben wurde.
Finished ( )	Das Ereignis wird ausgelöst, wenn ein Befehl korrekt beendet wurde.
Progress ( )	Dieses Ereignis wird periodisch ausgelöst, wenn etwas heruntergeladen oder hochgeladen wird.
Read()	Das Ereignis wird ausgelöst, wenn Daten empfangen wurden.

Tabelle 24.2.4.0.3 : Ereignisse der Klasse HTTPClient

24.2.4.0.4 Beispiel – HTTP-Client

Im Beispiel wird die Get()-Methode ohne Parameter genutzt. Deshalb werden die vom HTTP-Server empfangenen Daten im Speicher abgelegt. Auf diese Daten wird mit Standard-Stream-Methoden zugegriffen. Die Daten der Website sind überschaubar klein und werden in einer Textbox angezeigt:

Abbildung 24.2.4.0.1: Ausgabe der externen IP-Adresse

Das ist der komplette Quelltext:

```
[1]  ' Gambas class file
[2]
[3]  Public Sub Form_Open()
[4]
[5]      MAdditional.CheckNetwork()
[6]      FMain.Caption = ("Current external IP address")
[7]
[8]  '-- Note the text on this web page: https://ipecho.net/developers.html
[9]      lblIPAdress.Text = HTTPGetIP("https://ipecho.net/plain")
[10]
[11] End
[12]
[13] Public Function HTTPGetIP(argURL As String) As String
[14]
[15]     Dim hHTTPClient As HttpClient
[16]     Dim sResult As String
[17]
[18]     hHTTPClient = New HttpClient
[19]     hHTTPClient.URL = argURL
[20]     hHTTPClient.Async = False
[21]     hHTTPClient.TimeOut = 10
[22]
[23]     hHTTPClient.Get()
[24]
[25]     If Lof(hHTTPClient) Then sResult = Read #hHTTPClient, Lof(hHTTPClient)
[26]
[27]     Return sResult
[28]
[29] End
```

Kommentar:

- In der Zeile 5 wird mit der Prozedur CheckNetwork getestet, ob ein Netz erreichbar ist.
- Die externe IP-Adresse wird in der Zeile 9 als Wert der Funktion HTTPGetIP(…) angezeigt.
- Zuerst wird in der Zeile 18 in der Funktion HTTPGetIP(…) ein neuer HTTPClient erzeugt.
- Dem Client werden dann wesentliche Eigenschaften zugewiesen. Die URL der aufzurufenden Seite wird als Argument der o.a. Funktion in der Zeile 19 übergeben.
- Danach wird die Eigenschaft Async auf False gesetzt, was das Setzen eines Timeout nach sich zieht → Eigenschaft TimeOut in der Tabelle 24.2.4.0.1.
- Anschließend werden die Daten in der Zeile 23 mit der Get()-Methode abgerufen und im Speicher abgelegt.
- In der Zeile 25 wird abschließend der Speicher komplett ausgelesen und im Funktionswert von der Funktion HTTPGetIP(…) gespeichert, der in der Zeile 27 ausgegeben wird.

Das Projekt nutzt das Modul MAdditional, das u.a. diese Prozedur enthält:

```
Public Sub CheckNetwork()

    Dim sResponse, sCommand As String
    Dim sIPAddress As String

'-- With Desktop.NetworkAvailable (gb.desktop) you can determine whether a network connection exists.
    If Not Desktop.NetworkAvailable Then
       Message.Error(("No network available!<hr>The application will be terminated."))
       Quit
    Endif

'-- Check connection to local router by trying to obtain the IP address of the gateway
    sCommand = "route -n | grep ^0.0.0.0 | awk '{print $2}'"
    Shell sCommand To sIPAddress
    If Not sIPAddress Then
       Message.Error(("No connection to lokal router!<hr>The application will be terminated."))
       Quit
    Endif

    sCommand = "ping -c 1 8.8.8.8"
    Shell sCommand To sResponse
    If Not InStr(sResponse, "8.8.8.8 ping") Then
       Message.Error(("No connection to the Internet!<hr>The application will be terminated."))
       Quit
    Endif

End
```