24.1.3.0 Klasse UDPSocket

Die Klasse UDPSocket (gb.net) ermöglicht es Gambas-Programmen über UDP-Sockets zu kommunizieren. Die Klasse erbt von der Stream-Klasse, so dass Sie Standard-Stream-Methoden zum Lesen, Schreiben und Schließen des Sockets verwenden können.

24.1.3.0.1 Hinweise zum UDP-Protokoll

Im Gegensatz zum TCP-Protokoll arbeitet das UDP-Protokoll (User Datagram Protocol) verbindungslos. Das bedeutet, dass zwischen zwei Programmen mit UDP-Sockets keine Netzverbindung mit kontrolliertem Datenfluss besteht. Daher werden Sie die beiden Methoden *Listen()* und *Accept()* bei UDP-Sockets nicht finden. Es gibt auch keine Bestätigung, ob die Daten – sogenannte Datagramme – beim Kommunikationspartner angekommen sind. Es werden nur Datagramme gesendet beziehungsweise empfangen. Nicht nur deshalb gilt das UDP-Protokoll als unsicheres Protokoll.

Gut zu wissen:

- Die Klasse kann als Server oder als Client verwendet werden, da jedes gesendete oder empfangene Datagramm über seine Host-IP und seinen Port identifiziert wird.
- Bei einem Unix-Socket müssen die Host-IP und der zu verwendende Port des Kommunikationspartners bekannt sein, ehe ein Datagramm gesendet werden kann.
- Wenn zum Beispiel für einen Server ein UDP-Socket erzeugt und initialisiert wurde, können Sie sofort mit dem Senden von Datagrammen beginnen. Die Initialisierung besteht darin, dass der Server sich an einen Port bindet, der auch den Clients bekannt sein muss!
- Mit Hilfe eines Rundrufs im Subnetz kann ein Client erfragen, welche Kommunikationspartner noch auf dem Port des Clients unterwegs sind. Sie benötigen dafür die Broadcast-IP-Adresse für das Subnetz und müssen unbedingt die Eigenschaft UDPSocket.Broadcast auf True setzen, weil sonst ein Fehler ausgelöst wird. Meldet sich ein Kommunikationspartner – zum Beispiel ein Server – dann können Sie dessen IP-Adresse aus seiner Antwort ablesen und die Kommunikation beginnen.
- Ein UDP-Socket ist – im Gegensatz zu einem TCP-Socket – nicht an einen einzelnen Kommunikationspartner gebunden!

Die zuletzt aufgeführten Punkte werden in den folgenden Quelltext-Fragmenten für einen UDP-Server und einen UDP-Client umgesetzt:

UDP-Server:

```
Public UDPServerSocket As UDPSocket

'-- Es wird ein UDP-Socket erzeugt ▶ SERVER
    UDPServerSocket = New UDPSocket As "UDPServerSocket"

    If UDPServerSocket.Status <= Net.Inactive Then
       UDPServerSocket.Port = Val(txbPort.Text)
       UDPServerSocket.Bind()
    Endif

Public Sub UDPServerSocket_Read()

    Dim sBuffer, sQuote As String

    Read #UDPServerSocket, sBuffer, Lof(UDPServerSocket)

'-- Run Service: Ausgabe eines zufälligen Zitates → Methode GetQuote(…)
    sQuote = GetQuote(sBuffer)

'-- Ziel-Adressierung: Host und Port
'-- Über die Eigenschaften UDPServerSocket.SourceHost und UDPServerSocket.SourcePort findet man heraus,
'-- woher das Datagramm kam – wer der Absender war. Damit sind die Ziel-Informationen zum Senden komplett.

    UDPServerSocket.TargetHost = UDPServerSocket.SourceHost
    UDPServerSocket.TargetPort = UDPServerSocket.SourcePort

'-- Das Zitat wird als Datagramm an das adressierte Ziel gesendet
    Write #UDPServerSocket, Trim(sQuote), Len(Trim(sQuote))

End
```

UDP-Client:

```
Public UDPClientSocket As UDPSocket

'-- Es wird ein UDP-Socket erzeugt ▶ CLIENT
    UDPClientSocket = New UDPSocket As "UDPClientSocket"

    If UDPClientSocket.Status <= Net.Inactive Then
       UDPClientSocket.Port = 0 ' Standard-Port für einen UDP-Client
       UDPClientSocket.Bind()
    Endif

Public Sub SendDatagram()

'-- Ziel-Adressierung: Host und Port
    UDPClientSocket.TargetHost = txbHost.Text
    UDPClientSocket.TargetPort = txbPort.Text

    Inc Application.Busy
    '-- Ein Datagramm wird an das adressierte Ziel gesendet
        Write #UDPClientSocket, txbRequest.Text, Len(txbRequest.Text)
    Dec Application.Busy

End
```

24.1.3.0.2 Eigenschaften

Die Klasse *UDPSocket* verfügt über diese Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
Host	String	Gibt die IP-Adresse zurück, an die der UDP-Socket gebunden ist oder setzt sie.
SourceHost	String	Liefert die IP-Adresse der Quelle zurück, wenn die Nachricht aus dem Internet kommt.
SourcePort	Integer	Liefert den Port der Quelle zurück, wenn die Nachricht aus dem Internet kommt.
SourcePath	String	Liefert den Pfad der Quelle zurück, wenn die Nachricht von einem lokalen Socket kommt.
TargetHost	String	Definiert die IP-Adresse (Ziel), wenn die Nachricht ins Internet geht.
TargetPort	Integer	Definiert den Port (Ziel), wenn die Nachricht ins Internet geht.
TargetPath	String	Definiert den Pfad (Ziel), wenn die Nachricht an einen lokalen Socket geht.
Path	String	Gibt den Pfad des *lokalen* Sockets zurück, zu dem verbunden werden soll oder legt den Pfad fest.
Port	Integer	Gibt die Port-Nummer zurück, die zum Binden des UDP-Sockets verwendet wird oder legt sie fest.
Status	Integer	Gibt den Status des UDPSockets als Konstante der Net-Klasse zurück → Kapitel 24.x.y.
Timeout	Integer	Gibt den Timeout in Millisekunden zurück oder setzt ihn.
Broadcast	Boolean	Diese Eigenschaft setzt oder löscht das Socket-Broadcast-Flag.

Tabelle 24.1.3.0.1 : Eigenschaften der Klasse UDPSocket

Hinweise:

- Port: Der Wert muss eine natürliche Zahl zwischen 0 und 65535 sein. Port 0 ist der übliche Port für UDP-Clients. 1 bis 65535 bindet einen Server-Socket an den angegebenen Port. Eigene Ports sollten im Bereich von 49152 bis 65535 vergeben werden.
- Auf https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers finden Sie eine Übersicht vergebener UDP-Port-Nummern.
- Broadcast: Wenn die Eigenschaft UDPSocket.Broadcast den Wert True hat, können UDP-Sockets Datagramme an eine Broadcast-Adresse gesendet werden. Beispiel: Bei einer Subnetz-IP-Adresse von 192.168.2 und der Standard-Subnetz-Maske 255.255.255.0 ist die Broadcast-IP-Adresse 192.168.2.255. Allgemeiner wäre die Broadcast-IP-Adresse 255.255.255.255.

24.1.3.0.3 Methoden

Die Klasse *UDPSocket* hat die folgenden Methoden:

Methode	Rückgabetyp	Beschreibung
Begin ( )	-	Beginn der Pufferung der in den Stream geschriebenen Daten im Socket-Puffer, so dass beim Aufruf der Send-Methode alles gesendet wird.
Close ()	-	Schließt den UDP-Socket.
Send ( )	-	Sendet alle Daten auf einen Schlag, die seit dem letzten Aufruf der Begin()-Methode im Socket-Puffer gepuffert wurden.
Bind ( )	-	Bindet einen Socket an einen Host oder an einen Pfad, damit das Objekt mit dem Senden und Empfangen von Daten beginnen kann.
Drop ( )	-	Löscht alle Daten, die seit dem letzten Aufruf der Begin()-Methode im Socket-Puffer gepuffert wurden.
Peek ( )	String	Mit dieser Funktion können Sie Daten vom Socket empfangen. Sie darf nur verwendet werden, wenn die Verbindung aktiv ist (Status = Net.Connected), sonst wird ein Fehler ausgelöst.
ReadLine ([Escape As String])	String	Liest eine Textzeile aus dem Datenstrom, wie die bei der Anweisung LINE INPUT.
Watch (Mode As Integer, Watch As Boolean)	-	Starten oder stoppen Sie die Überwachung des Stream-Datei-Deskriptors zum Lesen oder Schreiben, nachdem er geöffnet wurde.

Tabelle 24.1.3.0.2 : Methoden der Klasse UDPSocket

Hinweise:

- Bind(): Wenn Path definiert ist, dann ist der Socket ein lokaler Socket, der an ihn gebunden ist. Andernfalls ist der Socket ein Internet-Socket, der an den durch die Eigenschaft Port angegebenen Port gebunden ist. Diese Funktion kann nur verwendet werden, wenn die Eigenschaft Status gleich Null (inaktiv) oder kleiner als Null ist. Andernfalls wird ein Fehler ausgelöst.
- Peek(): Die Zeichenfolge wird nicht aus dem Socket-Puffer gelöscht. Wenn Sie also das nächste Mal Daten als Datagramm empfangen, erhalten Sie die gleiche Zeichenfolge, ergänzt durch neue Daten, die im UDP-Socket angekommen sind.
- ReadLine(…): Wenn ein 'Escape'-Zeichen angegeben ist, dann werden Zeilenumbrüche zwischen zwei Escape-Zeichen ignoriert. Diese Methode ist beim Lesen von CSV-Dateien sehr nützlich.
- Watch(): 'Mode' ist der Überwachungstyp: Setzen Sie gb.Read zum Beobachten beim Lesen und gb.Write zum Beobachten beim Schreiben. Mit Watch = True wird die Überwachung aktiviert und mit Watch = False deaktiviert.

24.1.3.0.4 Ereignisse

Die Klasse *UDPSocket* verfügt nur über diese beiden Ereignisse:

Ereignis	Beschreibung
Error ( )	Dieses Ereignis tritt auf, wenn Anweisungen fehlgeschlagen sind. Zum Beispiel dann, wenn ein Bindungsversuch fehlschlug. Die Status-Eigenschaft hat dann einen negativen Wert, der den Fehlertyp widerspiegelt. Der UDP-Socket wird nach einem Fehler *automatisch* geschlossen!
Read ( )	Dieses Ereignis wird ausgelöst, wenn Daten vom Kommunikationspartner der anderen Seite der IP-Verbindung im Socket ankommen.

Tabelle 24.1.3.0.3 : Ereignisse der Klasse UDPSocket

Hinweis:

- Error(): Die Kenntnis ausgewählter Fehler-Konstanten kann Ihnen helfen, den Grund für auftretende Fehler zu erkennen:

```
Net.CannotCreateSocket     (-2)   Das System erlaubte es nicht, einen UDP-Socket zu erzeugen.
Net.CannotRead             (-4)   Fehler beim Versuch, Daten zu empfangen.
Net.CannotWrite            (-5)   Fehler beim Versuch, Daten zu senden.
Net.CannotBindSocket       (-10)  Der gewünschte Port konnte nicht verwendet (gebunden) werden.
```