24.1.2.0 Klasse ServerSocket

Mit der Klasse ServerSocket (gb.net) wird das server-seitige Warten auf eingehende Verbindungen und deren Aufbau ermöglicht. Sie ist nicht für die eigentliche Kommunikation vorgesehen. Die Klasse kann sowohl für das TCP- als auch für das Unix-Protokoll verwendet werden. Beachten Sie: Den Unix-Sockets ist ein eigenes Kapitel in '24.1.4 UnixSocket' gewidmet.

Die Klasse führt ihre Arbeit asynchron aus, so dass Ihr Programm nicht durch die interne Arbeit des Server-Objekts gestoppt wird.

So erzeugen Sie einen neuen Internet-Server-Socket:

```
Dim hServerSocket As ServerSocket
hServerSocket = New ServerSocket ( [ Port As Integer, MaxConn As Integer ] )  As "EventName"
```

Alternative:

```
Dim hServerSocket As ServerSocket

hServerSocket = New ServerSocket As "EventName"
hServerSocket.Type = Net.Internet
hServerSocket.Port = iPort
hServerSocket.Listen(10) ' Der Server baut maximal 10 TCP/IP-Verbindungen auf
```

So erzeugen Sie einen neuen Unix-Server-Socket:

```
Dim hServerSocket As ServerSocket
hServerSocket = New ServerSocket ( [ Path As String, MaxConn As Integer ] )  As "EventName"
```

Alternative:

```
Dim hServerSocket As ServerSocket

hServerSocket = New ServerSocket As "EventName"
hServerSocket.Type = Net.Unix
hServerSocket.Path = sPath
hServerSocket.Listen(10) ' Der Server baut maximal 10 Verbindungen auf
```

24.1.2.0.1 Accept()-Methode

Diese Klasse ist so konzipiert, dass sie einen Server präsentiert, der angeforderte Verbindungen annimmt oder ablehnt. Die gesamte restliche Arbeit wie zum Beispiel das Senden von Daten oder das Empfangen von Daten wird von einem Socket ausgeführt, der vom Server erzeugt und gestartet wird, einer pro Verbindung Server ↔ Client.

Mit der folgenden Kontroll-Struktur inspizieren Sie alle Sockets, die mit der Accept()-Methode erzeugt wurden:

```
Dim hServerSocket As ServerSocket
Dim hConnetionSocket As Socket

For Each hConnetionSocket In hServerSocket
  ...
Next
```

24.1.2.0.2 Eigenschaften

Die Klasse *ServerSocket* verfügt über diese Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
Count	Integer	Gibt die Anzahl der Verbindungssockets zurück, die mit der Accept()-Methode erzeugt wurden.
Type	Integer	Gibt den zu verwendenden Socket-Typ zurück oder legt ihn fest. Der Socket-Typ kann einer der folgenden Werte sein: Net.Internet (1) oder Net.Local (0).
Port	Integer	Gibt den Server-Socket-Port für Sockets vom Server-Typ *Net.Internet* zurück oder

Eigenschaft	Datentyp	Beschreibung
		legt ihn fest.
Path	String	Gibt den Server-Socket-Pfad für Unix-Sockets vom Server-Typ *Net.Local* zurück oder legt ihn fest.
Status	Integer	Gibt den Status des Server-Sockets als Konstante der Net-Klasse zurück.
Timeout	Integer	Gibt den Server-Socket-Timeout in Millisekunden zurück oder setzt ihn.

Tabelle 24.1.2.0.1 : Eigenschaften der Klasse ServerSocket

Hinweise:

- Synonym für Net.Local wird Net.Unix verwendet.
- Ein Socket-Timeout begrenzt die Zeitspanne, die für die Kommunikation zwischen Verbindungen zulässig ist. Dies kann der Aufbau einer Verbindung oder das Senden von Daten sein. Sie wird verwendet, um zu verhindern, dass störende Verbindungen Ressourcen beanspruchen, und um die Sicherheit durch Schließen abgebrochener Verbindungen zu erhöhen. Diese Eigenschaft kann geändert werden, wann immer dies notwendig oder sinnvoll ist. Ein Fall, in dem Sie neue Verbindungen unterbrechen möchten, liegt vor, wenn Sie einen umfangreichen Datentransfer durchführen müssen und diesem Vorrang einräumen möchten. So können Sie verhindern, dass neue Verbindungen dieselbe Art von Datentransfer-Anforderung stellen und Ihren Server überlasten.

```
If sBuf = "start-massive-transfer" Then
   btnPause.Enabled = False
   Print "Doing massive transfer."
   MyServerSocket.Pause()
'-- Prevent this priority connection from keeping the others waiting too long
'-- Verhindern Sie, dass diese Prioritätsverbindung die anderen zu lange warten lässt
   MyServerSocket.Timeout = 10 * 1000
Else If sBuf = "end-massive-transfer" Then
   Print "Massive transfer done."
   MyServerSocket.Timeout = 0
'-- Retain btnPause control of Pause/Resume state
'-- btnPause-Steuerung des Pause/Fortsetzungs-Status beibehalten
   If Not Waiting Then MyServerSocket.Resume()
   btnPause.Enabled = True
Endif
```

24.1.2.0.3 Methoden

Die Klasse *ServerSocket* besitzt folgende Methoden:

Methode	Rückgabetyp	Beschreibung
Accept ( )	Socket	Verwenden Sie diese Methode, um eine Verbindungsanforderung von einem Client zu akzeptieren. Sie müssen diese Methode immer im Ereignis Connection verwenden. Die Funktion gibt ein mit dem Client verbundenes Socket-Objekt zurück, mit dem Sie diese Verbindung verwalten und die Kommunikation durchführen können. Das neue Socket-Objekt wird automatisch an den Ereignisbeobachter "Socket" angehängt.
Close ()	-	Verwenden Sie diese Methode, um alle vom Server eingerichteten Verbindungen zu schließen und den Lausch-Vorgang zu beenden.
Listen ([MaxConn As Integer])	-	Startet das Abhören am ausgewählten TCP-Port oder am lokalen Pfad. Optional können Sie den Parameter 'MaxConn' übergeben.
Pause ()	-	Verwenden Sie diese Methode, um alle bestehenden Verbindungen aufrechtzuerhalten. Sie nehmen aber keine weiteren an – bis Sie die Resume()-Methode verwenden.
Resume ()	-	Verwenden Sie diese Methode, um das Abhören auf neue Verbindungsanfragen wieder zu starten, wenn Sie diesen Prozess vorher mit der Pause()-Methode gestoppt haben.

Tabelle 24.1.2.0.2 : Methoden der Klasse ServerSocket

Hinweise:

- Der Parameter *MaxConn* kann 0 – keine Begrenzung der Anzahl der Verbindungen oder größer als 0 sein und damit die maximale Anzahl der gleichzeitig aktiven Verbindungen vorgeben.
- Wenn Sie den optionalen Parameter *MaxConn* nicht verwenden, dann gibt es keine Begrenzung der Anzahl von Verbindungen, weil die Eigenschaft *MaxConn* dann intern automatisch auf 0 gesetzt.
- Nach dem Aufruf der Close()-Methode wird der Server-Status auf *Net.Inactive* gesetzt. Alle Socket-Clients, die zuvor erzeugt wurden um die einzelnen Verbindungen zwischen Client und Server zu verwaltenund über die die Kommunikation zwischen Client und Server läuft, werden in den Status 'inaktiv' versetzt.
- Sie können die Pause()-Methode auch verwenden, wenn der Socket aktuell nicht auf Verbindungen wartet. Wenn er dann mit dem Warten auf Verbindungen beginnt, verweigert er aber alle eingehenden Verbindungen bis Sie die Resume()-Methode aufrufen.

24.1.2.0.4 Ereignisse

Die Klasse ServerSocket hat nur diese beiden Ereignisse:

Ereignis	Beschreibung
Connection (RemoteHostIP As String)	Das Ereignis ausgelöst, wenn ein Client versucht, eine Verbindung zum Server herzustellen (Request). Das ist die einzige Stelle im Quelltext, an der Sie diese Verbindung akzeptieren oder ablehnen können.
Error ( )	Dieses Ereignis wird ausgelöst, wenn innerhalb der Listen()-Methode Fehler auftraten. Die Status-Eigenschaft nimmt einen Wert kleiner Null an.

Tabelle 24.1.2.0.3 : Ereignisse der Klasse ServerSocket

Hinweise:

- Connection(…)
  Um eine Verbindung anzunehmen und mit ihr zu arbeiten, verwenden Sie die Accept()-Methode. Um sie abzulehnen, tun Sie einfach nichts – lassen Sie das Ereignis unbehandelt. Die Verbindung mit diesem Client wird geschlossen. Der Parameter 'RemoteHostIP' ist die IP-Adresse des Clients, der versuchte, eine Verbindung zum Server aufzubauen.
- Error()
  Der Wert der Fehler-Konstante kann Ihnen helfen, den Grund für einen Fehler zu erkennen: Net.CannotCreateSocket (-2), Net.CannotBindSocket (-10) oder Net.CannotListen (-14)