24.1.1 Socket (gb.net)

Aus der Sicht einer Gambas-Anwendung ist ein Socket ein Zugang zum Netzwerk und kapselt die Details der Netzwerkkommunikation. Die Methoden der Socket-Klasse stellen sowohl die Funktionen zum Aufbau von Verbindungen als auch zur Kommunikation zwischen mindestens zwei Netzwerk-Teilnehmern bereit. TCP-, UDP- und lokale (Unix-Sockets) Verbindungen sind implementiert.

Beachten Sie: Den Unix-Sockets ist ein eigenes Kapitel in '24.1.4 UnixSocket' gewidmet.

Diese Klasse führt ihre Arbeit asynchron aus. Programme werden beim Verbinden, Senden oder Empfangen von Daten nicht angehalten. Diese Klasse ist von der Klasse Stream abgeleitet, so dass Sie deren Standard-Methoden zum Senden und Empfangen von Daten sowie zum Schließen des Sockets verwenden können.

So erzeugen Sie einen neuen Socket:

```
Dim hSocket As Socket
hSocket = New Socket() As "EventName"
```

24.1.1.1 Eigenschaften

Die Klasse Socket verfügt über diese Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
Host	String	Der Host, zu dem sich der Client verbinden möchte (IP-Adresse oder Host-Name).
LocalHost	String	Sobald eine TCP-Verbindung zu einem Server hergestellt worden ist, spiegelt diese Eigenschaft die lokale IP-Adresse wider, die von dieser Verbindung verwendet wird.
LocalPort	Integer	Sobald eine TCP-Verbindung zu einem Server hergestellt worden ist, spiegelt diese Eigenschaft den lokalen Port (1-65535) wider, der von dieser Verbindung verwendet wird.
Port	Integer	Der Port mit dem Sie sich auf dem Remote-Host verbinden möchten
RemoteHost	String	Sobald eine TCP-Verbindung zu einem Server hergestellt worden ist, spiegelt diese Eigenschaft die von dieser Verbindung verwendete entfernte IP-Adresse wider.
RemotePort	Integer	Sobald eine TCP-Verbindung zu einem Server hergestellt worden ist, spiegelt diese Eigenschaft den von dieser Verbindung verwendeten Remote-Port (1-65535) wider.
Server	ServerSocket	Wenn ein Socket mit der Accept()-Methode eines Server-Sockets erzeugt wurde, wird mit dieser Eigenschaft ein ServerSocket-Objekt zurückgegeben. Andernfalls wird NULL zurückgegeben.
Status	Integer	Spiegelt den aktuellen Status eines Socket-Objekts wider.
Timeout	Integer	Liefert oder setzt den Wert für ein Timeout des Sockets in Millisekunden.

Tabelle 24.1.1.1.1 : Eigenschaften der Klasse Socket

Hinweise zu ausgewählten Werten (→ Kapitel 24.1.6 Net-Konstanten) der Eigenschaft Socket.Status:

- Net.CannotCreateSocket → Das System erlaubte es nicht, einen neuen Sockel zu erzeugen.
- Net.SocketRefused → Der Server verweigert die angeforderte Verbindung.
- Net.CannotRead → Daten können nicht gelesen werden, Verbindung unterbrochen.
- Net.CannotWrite → Daten können nicht geschrieben werden, Verbindung unterbrochen.
- Net.Searching → Es wird versucht, den Hostnamen in die IP-Adresse aufzulösen.
- Net.Connecting → Es wird versucht, eine Verbindung zu einem Remote-Server herzustellen.
- Net.Connected → Mit dem Remote-Server verbunden, bereit zum Senden und Empfangen.
- Net.Inactive → Socket geschlossen.

24.1.1.2 Methoden

Die Klasse Socket stellt die folgenden Methoden zur Verfügung:

Methode	Beschreibung
Begin ( )	Beginn der Pufferung der in den Stream geschriebenen Daten im Socket-Puffer, so dass beim Aufruf der Send()-Methode alles gesendet wird.
Send ( )	Sendet alle Daten auf einen Schlag, die seit dem letzten Aufruf von Begin()-Methode im Socket-Puffer gepuffert wurden.
Drop ( )	Löscht alle Daten, die seit dem letzten Aufruf der Begin()-Methode im Socket-Puffer gepuffert wurden.
Peek ( ) As 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]) As String	Liest eine Textzeile aus dem Datenstrom, wie bei der Anweisung LINE INPUT.
Watch (Mode As Integer, Watch As Boolean)	Erlaubt die Steuerung der Überwachung des Stream-Datei-Deskriptors zum Lesen oder Schreiben, nachdem er geöffnet wurde.

Tabelle 24.1.1.2.1 : Methoden der Klasse Socket

Hinweise:

- Peek(): Die Zeichenfolge wird nicht aus dem Socket-Puffer gelöscht. Wenn Sie also das nächste Mal Daten empfangen, erhalten Sie die gleiche Zeichenfolge, ergänzt durch neue Daten, die im 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 für das Beobachten des Lesens und gb.Write für das Beobachten des Schreibens. Mit Watch = True wird die Überwachung aktiviert und mit Watch = False deaktiviert.

24.1.1.3 Ereignisse

Die Klasse Socket verfügt über diese Ereignisse:

Ereignis	Beschreibung
Closed ( )	Dieses Ereignis tritt ein, nachdem die entfernte Seite des Sockets die Verbindung geschlossen hat. Die Status-Eigenschaft wird auf den Wert Net.Inactive gesetzt.
Error ( )	Dieses Ereignis tritt auf, wenn Anweisungen fehlgeschlagen sind. Zum Beispiel dann, wenn der Remote-Server die Verbindung verweigert hat. Die Status-Eigenschaft wird zu einem negativen Wert, der den Fehlertyp widerspiegelt und der Socket wird automatisch geschlossen.
Found ( )	Bei Verbindungen über TCP-Sockets besteht der erste Schritt darin, den Hostnamen in die Host-IP-Adresse aufzulösen oder die IP-Adresse zu bestätigen. Dieses Ereignis wird ausgelöst, wenn der Hostname erfolgreich in die Host-IP-Adresse aufgelöst wurde oder die IP-Adresse bestätigt werden konnte.
Ready ( )	Dieses Ereignis wird ausgelöst, nachdem eine Verbindung zwischen Client und Server erfolgreich hergestellt wurde. Die Status-Eigenschaft wird auf den Wert Net.Connected gesetzt.
Read ( )	Dieses Ereignis wird ausgelöst, wenn Daten vom Kommunikationspartner der anderen Seite der IP-Verbindung im Socket angekommen sind.
Write ( )	Dieses Ereignis wird ausgelöst, wenn die internen Socket-Sendepuffer weitere Daten aufnehmen können.

Tabelle 24.1.1.3.1 : Ereignisse der Klasse Socket

Hinweis:

- Write(): Dieses Ereignis ist am nützlichsten, wenn Sie eine große Datei senden wollen. Das

Write-Ereignis wird nicht kontinuierlich ausgelöst, da ein Socket fast immer zur Aufnahme von Daten bereit ist. Sie müssen Gambas signalisieren, dass Sie einige Daten zu schreiben haben, indem Sie einen ersten Schreibvorgang in den Socket durchführen. Nach dem anfänglichen Schreiben wird das Write()-Ereignis kontinuierlich ausgelöst, solange Sie weitere Daten innerhalb der Write-Ereignisbehandlungsroutine schreiben.