MyStringEchoClient		Prof. Dr.-Ing. habil. L. Winkler	HOCHSCHULE MITTWEIDA UNIVERSITY OF APPLIED SCIENCES
CPP-Project mit TClientSocket	2015-01	Fakultät Informations- und Elektrotechnik	

1 Ziel des Projektes

- Grundlegender Umgang mit einem Socket der Klasse TClientSocket aus der RAD-Studio-IDE.
- Es soll eine Client-Anwendung programmiert werde, die einen String an einen String-Echo-Server (TCP-Port 6666) sendet.
- Der Echo-Server wendet auf den Text die Funktion UpperCase() an und sendet den String zurück. Der Echo-Server steht zum Download bereit.

2 Die Klasse TClientSocket[1]

Wesentliche Eigenschaften, Methoden und Ereignisse dieser Klasse, werden nachfolgend vorgestellt. Wertet man alle Ereignisse aus und zeigt diese an, wird der Ablauf einer Socket-Nutzung gut sichtbar.

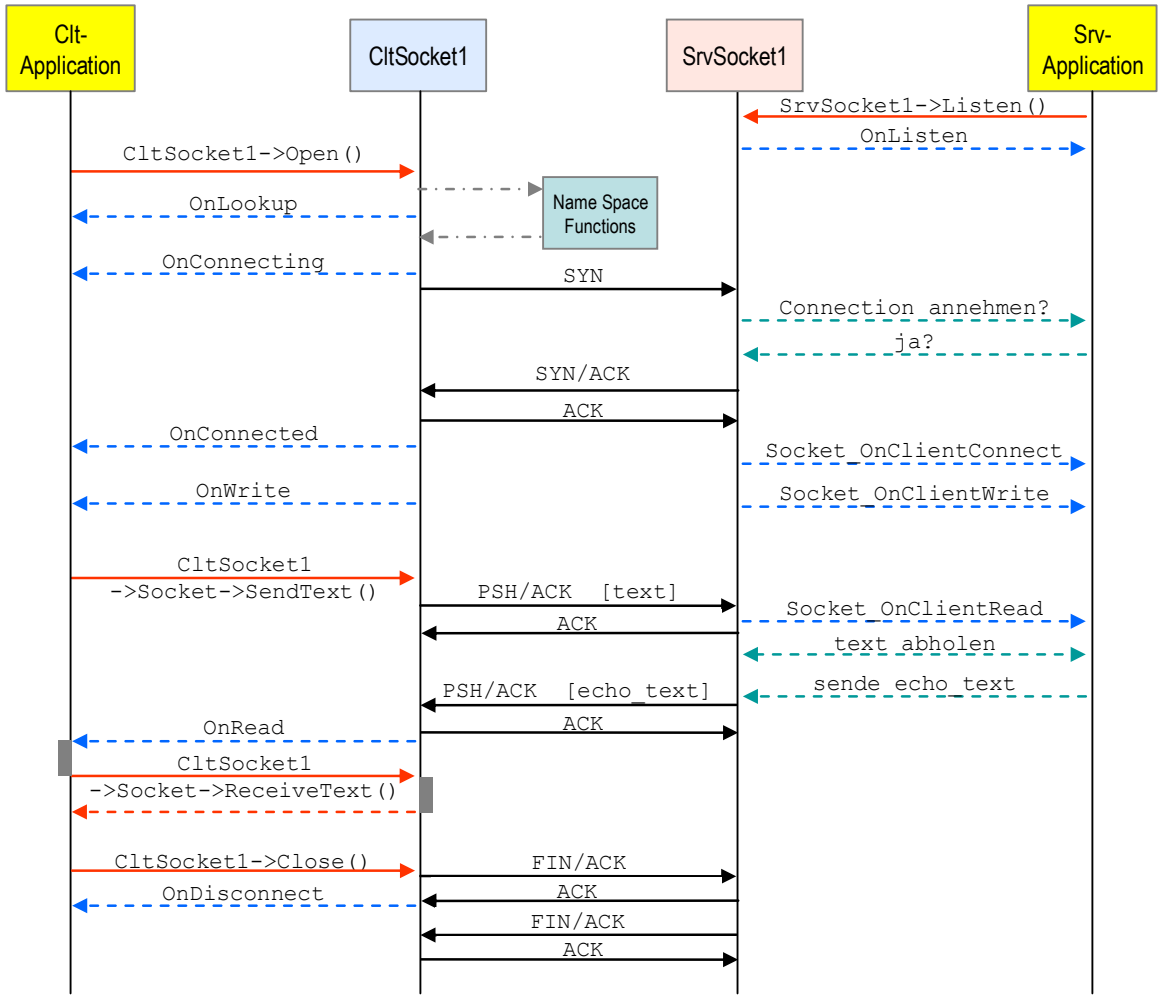
Properties der Klasse TClientSocket (Palette Internet)		
ClientType	ctBlocking|ctNonBlocking	Bei **ctBlocking** blockiert der Socket eine Funktion solange, bis er ein Ergebnis liefern kann. Wendet man z.B. die Methode Socket->ReceiveText() an und sind keine Daten zum Lesen da, blockiert der Socket solange bis Daten zum Lesen eintreffen. Bei **ctNonBlocking** liefert die Methode Socket->ReceiveText() entweder den vorhanden Text oder nichts, aber die Anwendung wird nicht blockiert. Will man das Socketereignis **onRead** nutzen, ist ctNonBlocking erforderlich.
Address oder	141.55.xxx.xxx	IP-Address des RemoteHost, zu dem die TCP-Verbindung hergestellt werden soll. Hat Host auch einen Wert, dominiert der Address-Eintrag.
Host	xxx.hs-mittweida.de	Hier kann man den Namen des RemoteHost angeben. Vor der Verbindungsherstellung wird dieser in eine IP-Adresse aufgelöst (DNS). Hat die Eigenschaft Address auch einen Wert, dominiert dieser!
Port oder	80 | 21 | 25	Hier kann man die RemotePortnummer des Dienstes angeben, zu dem eine Verbindung hergestellt werden soll. Gibt man zusätzlich den Service an, dominiert der Port.
Service	http|ftp |smtp	Anstelle der Portnummer kann man auch das Anwendungsprotokoll angeben. Vor der Verbindungsherstellung wird daraus der Port ermittelt.
Active	False|True	Standardwert ist False. Setzt man Active auf True, versucht der Socket sofort beim Start der Anwendung die Verbindung zum RemoteHost herzustellen.
Socket	*Nur zur Laufzeit*	Bezeichnet das TClientSocket-Objekt, welches eine Verbindung zu einem Server hergestellt hat.
Methoden der Klasse TClientSocket (Palette Internet)		
ClientSocket1->**Open()**		Die Verbindung zum Remote Host wird hergestellt. Property ClientSocket->Active hat anschließend den Wert True. Hat Property ClientSocket1->ClientType den Wert ctNonBlocking, erzeugt der Socket in Reihenfolge folgende Ereignisse: OnLookup, OnConnecting, OnConnected, OnWrite.
ClientSocket1->**Close()**		Die Verbindung zum Remote Host wird beendet. Property ClientSocket->Active hat anschließend den Wert False. Hat Property ClientSocket1->ClientType den Wert ctNonBlocking, erzeugt der Socket das Ereignis: OnDisconnect.
ClientSocket1->**SendText**(AnsiString)		Zum entfernten Socket einen Text vom Typ AnsiString senden
AnsiString=ClientSocket1->**ReceiveText()**		Vom entfernten Socket einen Text vom Typ AnsiString empfangen
Weitere Schreib- und Lesemethoden →siehe Hilfe		

[1] Die Komponente ist nicht standardmäßig in die IDE eingebunden.
Sollte sie die Komponente TClientSocket in der Palette „Internet“ nicht finden , geht man zum Einbinden der Komponente wie folgt vor:
(1) Menü: Komponenten→Packages installieren (2)→Hinzufügen
(3)C|D|X:\Programme\Embarcadero\RAD Studio\9.0\bin\dclsockets160.bpl wählen →Öffnen
(4)Die Komponentenpalette „Internet“ hat jetzt zwei neue Komponenten: TClientSocket, TServerSocket

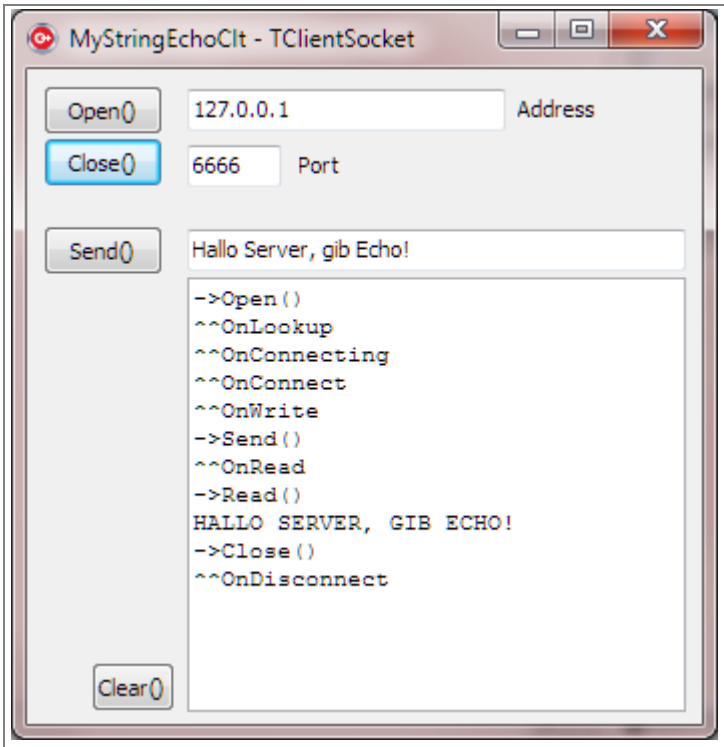
Ereignisse der Klasse TClientSocket (Palette Internet)	
Nur wenn ClientSocket1->ClientType=ctNonBlocking	
ClientSocket1**OnLookup**	Start von Routinen zur Adressenauflösung Host zu Address, der Diensteauflösung Service zu Port.
ClientSocket1**OnConnecting**	Wenn die Auflösung erfolgreich oder nicht erforderlich war, wird unmittelbar vor dem Versuch eine Verbindung aufzubauen, dieses Ereignis ausgelöst.
Clientsocket1**OnConnect**	Mit diesem Ereignis wird die erfolgreiche Herstellung einer connection zu RemoteAdress+RemotePort angezeigt.
ClientSocket1**OnWrite**	Mit diesesm Ereignis wird die erfolgreiche Herstellung einer Connection zu RemoteAdress+RemotePort angezeigt.
Clientsocket1**OnRead**	Dieses Ereignis wird ausgelöst, wenn Daten zum Lesen aus dem Socket bereitstehen. Der Anwendung wird damit eine einfache Möglichkeit gegeben aus den Socket zu lesen.
ClientSocket1**OnDisconnect**	Dieses Ereignis wird ausgelöst, wenn die Verbindung zum RemoteHost z.B. mit Close() beendet wurde.
ClientSocket1**OnError**	Diese Ereignis tritt ein, wenn ein Socket eine Verbindung nicht erstellen, verwenden oder schließen kann. Setzen Sie den Parameter ErrorCode auf 0, um das Entstehen einer ESocketError-Exception zu verhindern.

3 Ablauf einer Client-Server-Verbindung

Auf der Clientseite sieht man exakt alle Methoden, Ereignisse zwischen der Client-Anwendung und dem Socket sowie den damit im Zusammenhang stehenden TCP-Protokoll-Ablauf.
Die Serverseite zeigt nicht alle Details, sondern nur das Prinzip.



4 Realisierung des Projektes MyStringEchoClt



- Erzeugen Sie ein neues Projekt "MyStringEchoClt" im gleichnamigen Order.
- Falls die Klasse TClientSocket noch nicht vorhanden ist, installieren Sie die Komponente dclsockets100.bpl entsprechen der Anleitung Fußnote 1, Seite1!
- Editieren Sie die Oberfläche entsprechend der nebenstehenden Abbildung.Verwenden Sie folgene Klassen und Objektnamen:
  - TButton: Open_, Close_, Send_, Clear_
  - TEdit: Edit1, Edit2, Edit3
  - TLabel: Label1, Label2
  - TRichEdit: RichEdit1
- Erzeugen Sie Schritt für Schritt den Programmcode, entsprechend dem Beispiel-Code.

```
//==Notwendiger Programmtext==
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Open_Click(TObject *Sender)
{
      RichEdit1->Lines->Add("->Open()");
      ClientSocket1->Address=Edit1->Text;
      ClientSocket1->Port=Edit2->Text.ToInt();
      ClientSocket1->Open();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Close_Click(TObject *Sender)
{
      RichEdit1->Lines->Add("->Close()");
      ClientSocket1->Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Send_Click(TObject *Sender)
{
      RichEdit1->Lines->Add("->Send()");
      ClientSocket1->Socket->SendText(Edit3->Text);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Clear_Click(TObject *Sender)
{
      RichEdit1->Clear();
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::ClientSocket1Read(TObject *Sender,
      TCustomWinSocket *Socket)
{
     RichEdit1->Lines->Add("^^OnRead");
     RichEdit1->Lines->Add("->Read()");
     RichEdit1->Lines->Add(ClientSocket1->Socket->ReceiveText());
}

//==Nicht notwendiger Programmtext, dient nur zur Ereignisanzeige==

//---------------------------------------------------------------------------
void __fastcall TForm1::ClientSocket1Lookup(TObject *Sender,
      TCustomWinSocket *Socket)
{
     RichEdit1->Lines->Add("^^OnLookup");
}

//---------------------------------------------------------------------------
void __fastcall TForm1::ClientSocket1Connecting(TObject *Sender,
      TCustomWinSocket *Socket)
{
     RichEdit1->Lines->Add("^^OnConnecting");
}

//---------------------------------------------------------------------------
void __fastcall TForm1::ClientSocket1Connect(TObject *Sender,
      TCustomWinSocket *Socket)
{
     RichEdit1->Lines->Add("^^OnConnect");
}

//---------------------------------------------------------------------------
void __fastcall TForm1::ClientSocket1Write(TObject *Sender,
      TCustomWinSocket *Socket)
{
     RichEdit1->Lines->Add("^^OnWrite");
}

//---------------------------------------------------------------------------
void __fastcall TForm1::ClientSocket1Disconnect(TObject *Sender,
      TCustomWinSocket *Socket)
{
     RichEdit1->Lines->Add("^^OnDisconnect");
}

//---------------------------------------------------------------------------
void __fastcall TForm1::ClientSocket1Error(TObject *Sender,
      TCustomWinSocket *Socket, TErrorEvent ErrorEvent, int &ErrorCode)
{
     RichEdit1->Lines->Add("^^OnError "+IntToStr(ErrorCode));
     ErrorCode=0;
}
//---------------------------------------------------------------------------
```