Winsock in SDI Anwendungen

1. Neue Klasse: CMeinSocket (Client + Server)

MFC-Klasse: CMeinSocket, Basisklasse: **CAsyncSocket**
private Membervariable: **CDocument*** m_pDoc;
public Memberfunktion: **void** CMeinSocket::SetParent(**CDocument***pDoc){m_pDoc=pDoc;}

Socket- Nachrichten umleiten nach C...Doc mit virtuellen Funktionen:

A) Client

```
void CMeinSocket::OnConnect(int nErrorCode){
      if(nErrorCode == 0)
            ((C...Doc*)m_pDoc)->OnConnect();
}

void CMeinSocket::OnReceive(int nErrorCode){ ...->OnReceive(); }

void CMeinSocket::OnClose(int nErrorCode){ ...->OnClose(); }
```

B) Server

```
void CMeinSocket::OnAccept(int nErrorCode){ ...->OnAccept(); }

void CMeinSocket::OnReceive(int nErrorCode){ ...->OnReceive(); }

void CMeinSocket::OnClose(int nErrorCode){ ...->OnClose(); }
```

2. m_Socket

A) Client

Membervariable in C...Doc:

```
CMeinSocket m_Socket;
```

In OnNewDocument:

```
m_Socket.SetParent(this);
```

B) Server

Membervariable in C...Doc:

```
CMeinSocket m_SocketListen;
CMeinSocket m_Socket;
```

In OnNewDocument:

```
m_SocketListen.SetParent(this);
m_Socket.SetParent(this);
```

3. Verbindung herstellen

A) Client

```
CString sServer = „127.0.0.1“;
//                “loopback”
int iPort = 4711;
m_Socket.Create();
m_Socket.Connect(sServer, iPort);
```

B) Server

```
int iPort = 4711;
m_SocketListen.Create(iPort);
m_SocketListen.Listen();
```

4. Auf Nachricht warten, dass Verbindung steht

A) Client

erhält Nachricht **OnConnect:**

```
C...Doc::OnConnect()
{
      ...
}
```

B) Server

erhält Nachricht **OnAccept:**

```
C...Doc::OnAccept()
{
  m_SocketListen.Accept(m_Socket);
}
```

5. Datenaustausch (Client + Server)

Daten senden

a) Text

```
CString sNachricht = “blabla“;
int n;
n = sNachricht.GetLength();
m_Socket.Send(sNachricht, n);
```

b) Zahl

```
long nZahl = 13;
m_Socket.Send(&nZahl,sizeof(nZahl));
```

Daten empfangen

erhält Nachricht **OnReceive:**

```
C...Doc::OnReceive()
{
```

a) Text

```
CString sNachricht;
char Puffer[100];
int n, nMax = 99;
n=m_Socket.Receive(Puffer,nMax);
Puffer[n]= ‘\0’;
sNachricht = Puffer;
     ...
}
```

b) Zahl

```
long nZahl;
m_Socket.Receive(&nZahl,sizeof(nZahl));
     ...
```

6. Verbindung schliessen (Client + Server)

```
m_Socket.Close();
```

Wenn die Verbindung vom **Client** beendet wird, erhält der **Server** die Nachricht **OnClose,** wenn die Verbindung vom **Server** beendet wird, erhält der **Client** die Nachricht **OnClose**:

```
C...Doc::OnClose()
{
      ...
}
```