

Delphi Web Start



Ein Application Loader ohne
Browser Technologie als mobilen
Thin Client implementieren...

Kleiner
K o m m u n i k a t i o n

Rethink Design



- Was ist der Vorteil gegenüber GET/POST oder dem altgebackenen RPC? HTTP ist und bleibt zustandslos.
- Ein zusätzliches Fenster ist im Browser nur ein html Vorschlag. CSS ist abhängig der Auflösung und Popups sind deaktivierbar.
- Ein `on_exit` sendet zurück an den Server. Jedes Form ist eigentlich modal. Undo wird vielfach für Session Tricks missbraucht. Security wird aufgebohrt.

DWS: Worum geht es ?

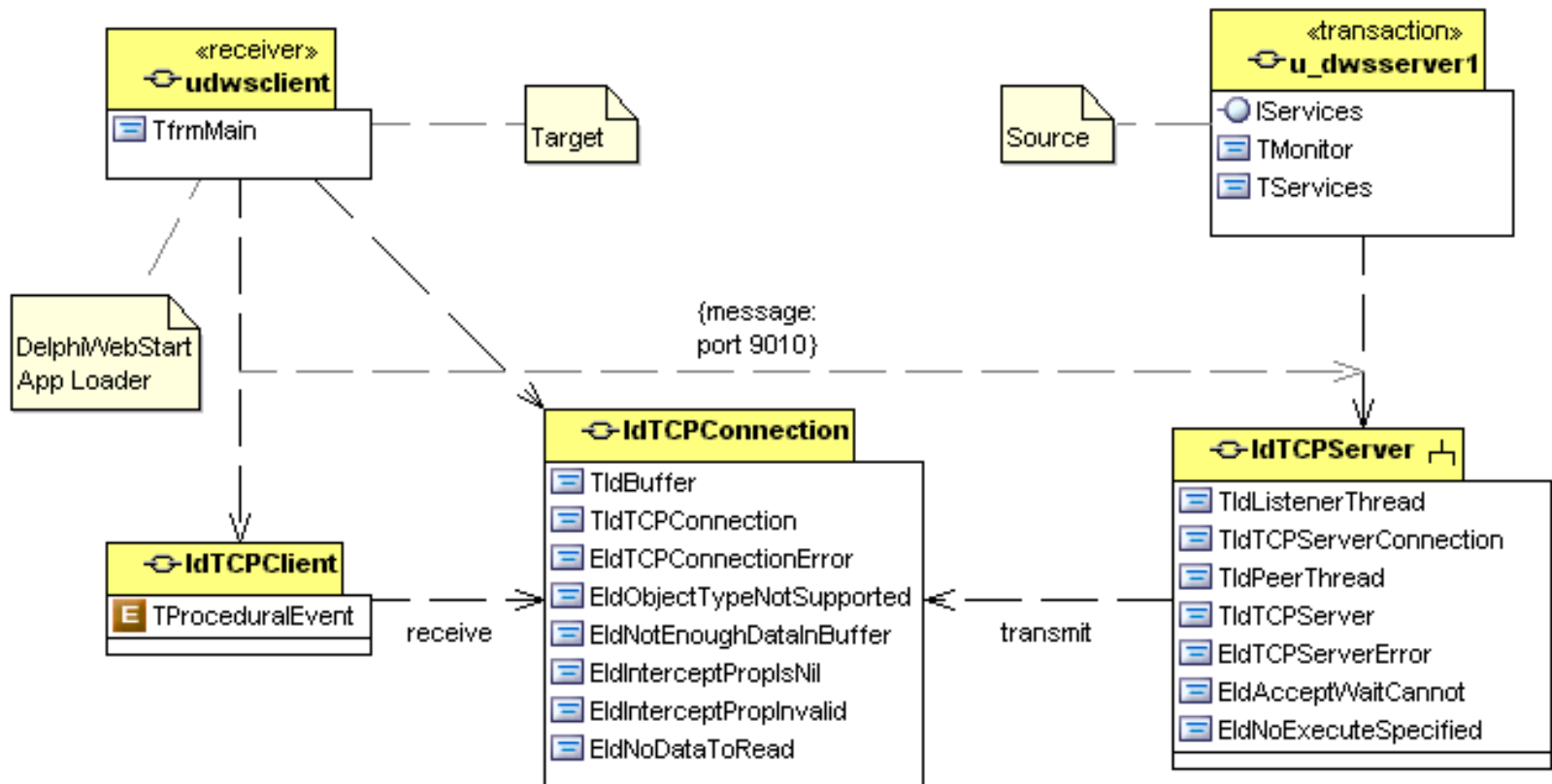


- dwsClient.exe als Client
- DWSServerWiz.exe als TCPServer
- qtintf.dll als Package
- InterBase / DBExpress als Logger

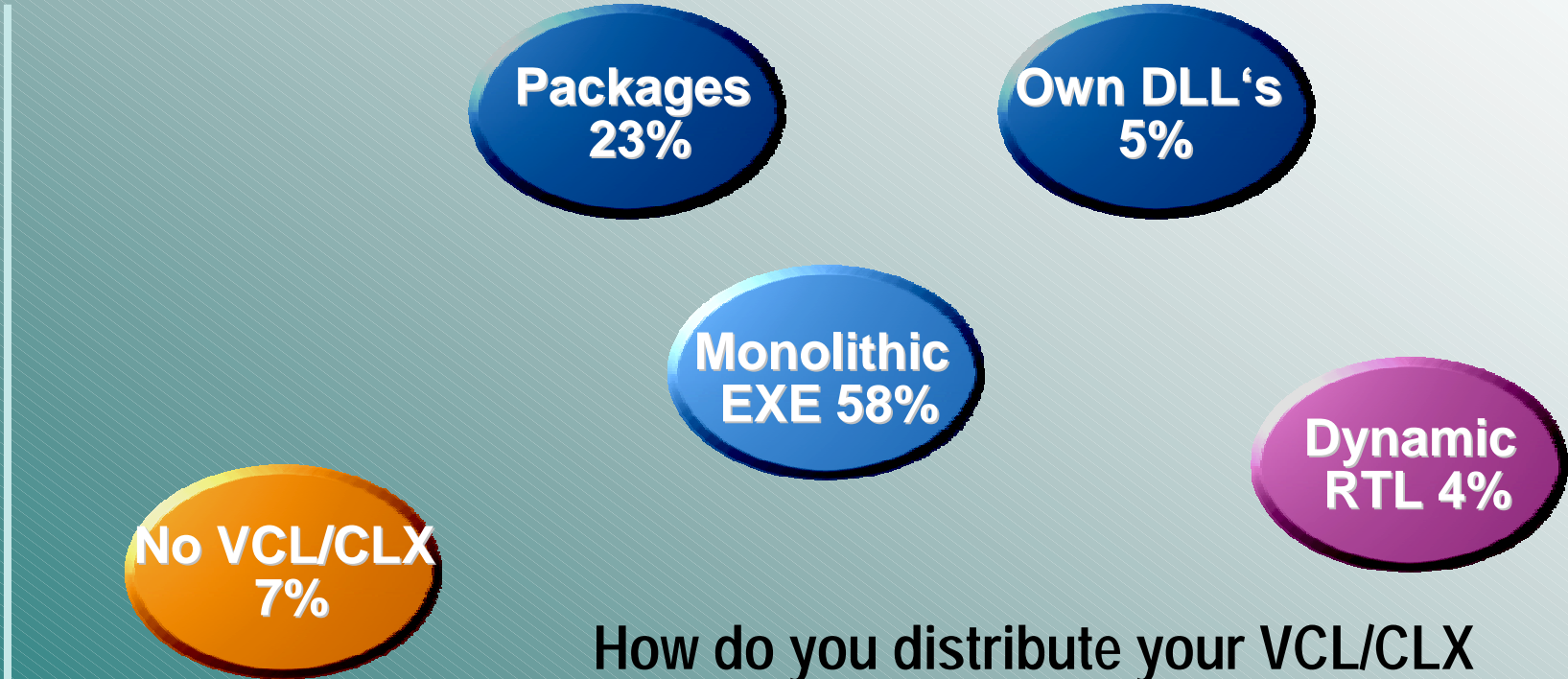
Die Elemente des DWS

- - IdTCPClient
- - IdTCPConnection
- - IdTCPServer

Struktur als Class Diagram

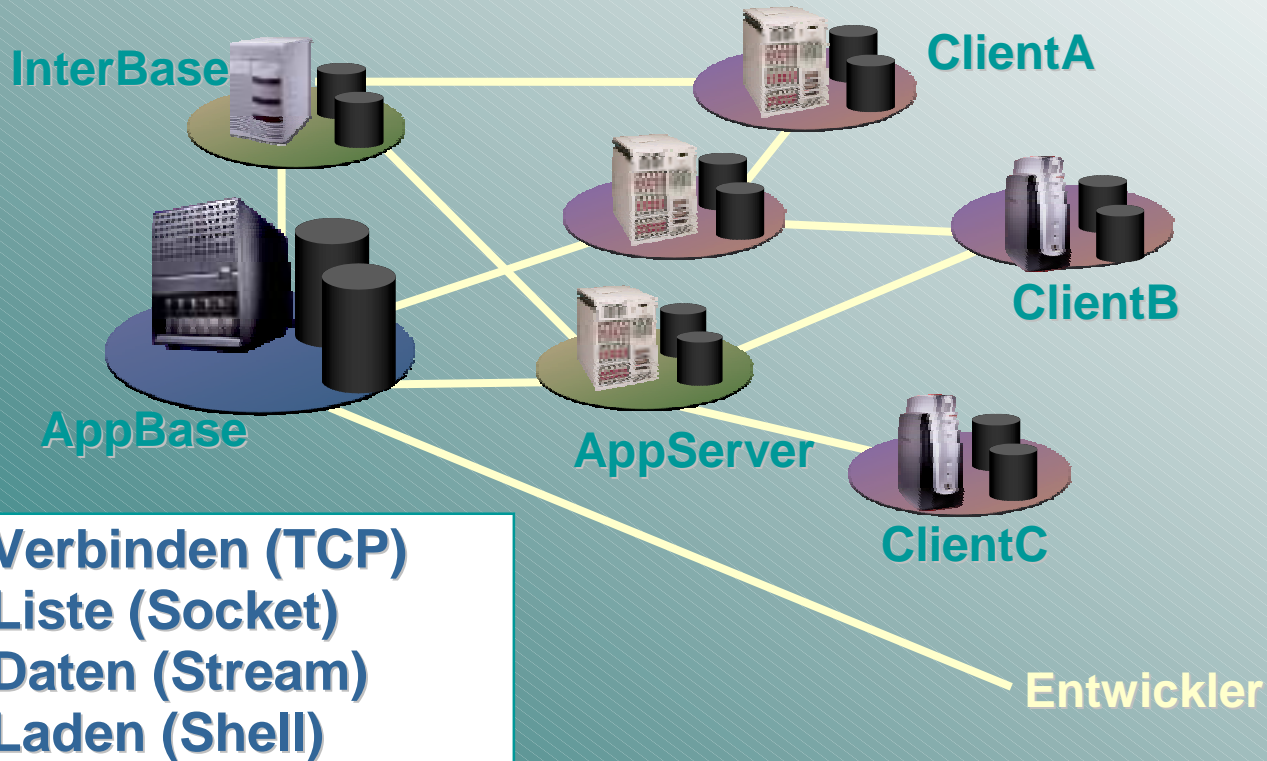


DWS Services



How do you distribute your VCL/CLX applications ?

Web Services konkret



Sockets



- *Dessen Socket ist durch eine Internet-Adresse (IP) und eine Port-Nummer nach dem Akzeptieren im System eindeutig identifiziert.*
- Der Socket Dämon startet einen **Serverprozess** und übergibt diesem nach dem `bind()` und `accept()` einen Handler auf einen Socket. Danach können Client und Server per `send()` und `recv()`-Funktionen Datenblöcke austauschen.
- Anwendungen können in einer Ereignisbehandlungsroutine für `OnGetSocket` einer Server-Socket-Komponente eigene Socket-Objekte erzeugen.

Socket Create()



- *Nach dem Aufruf des geerbten Konstruktor nimmt Create die folgenden Initialisierungen vor:*
 - *Die Initialisierung von WINSOCK.DLL wird überprüft.*
 - *Hilfsobjekte für das Multithreading werden zugewiesen.*
 - *ASyncStyles wird mit [asRead, asWrite, asConnect, asClose] initialisiert.*
 - *Eigenschaft SocketHandle mit ASocket initialisiert.*
 - *Eigenschaft Connected initialisiert und prüft ob ASocket Handle eines gültigen, geöffneten Socket ist.*
 - *Die Eigenschaft Addr wird initialisiert.*

Sockets Blocked – Non Blocked ?



- Sockets: Mit TCP/IP-Sockets können Sie extrem kleine Clients erstellen. Die von den Sockets bereitgestellten Verbindungen basieren zwar auf dem TCP/IP-Protokoll, sind aber so universell gehalten, dass auch verwandte Protokolle wie User Datagram Protocol (UDP), Xerox Network System (XNS) oder das DECnet von Digital verwendet werden können.

Alternativen



- Bei den TCP/IP-Sockets handelt es sich um eine Socket-Dispatcher-Anwendung namens *SCKTSRVR.EXE*.

- Für HTTP-Verbindungen wird *HTTPSRVR.DLL* verwendet, eine ISAPI/NSAPI-DLL, die zusammen mit dem Webserver installiert werden muss.

- Wenn das `ClientDataSet` mehr Daten benötigt, wird im Grunde die Methode `InternalGetRecords` des Providers aufgerufen und die Anzahl Records gemäß `PacketRecords` in `RecsOut` auf die Reise geschickt.

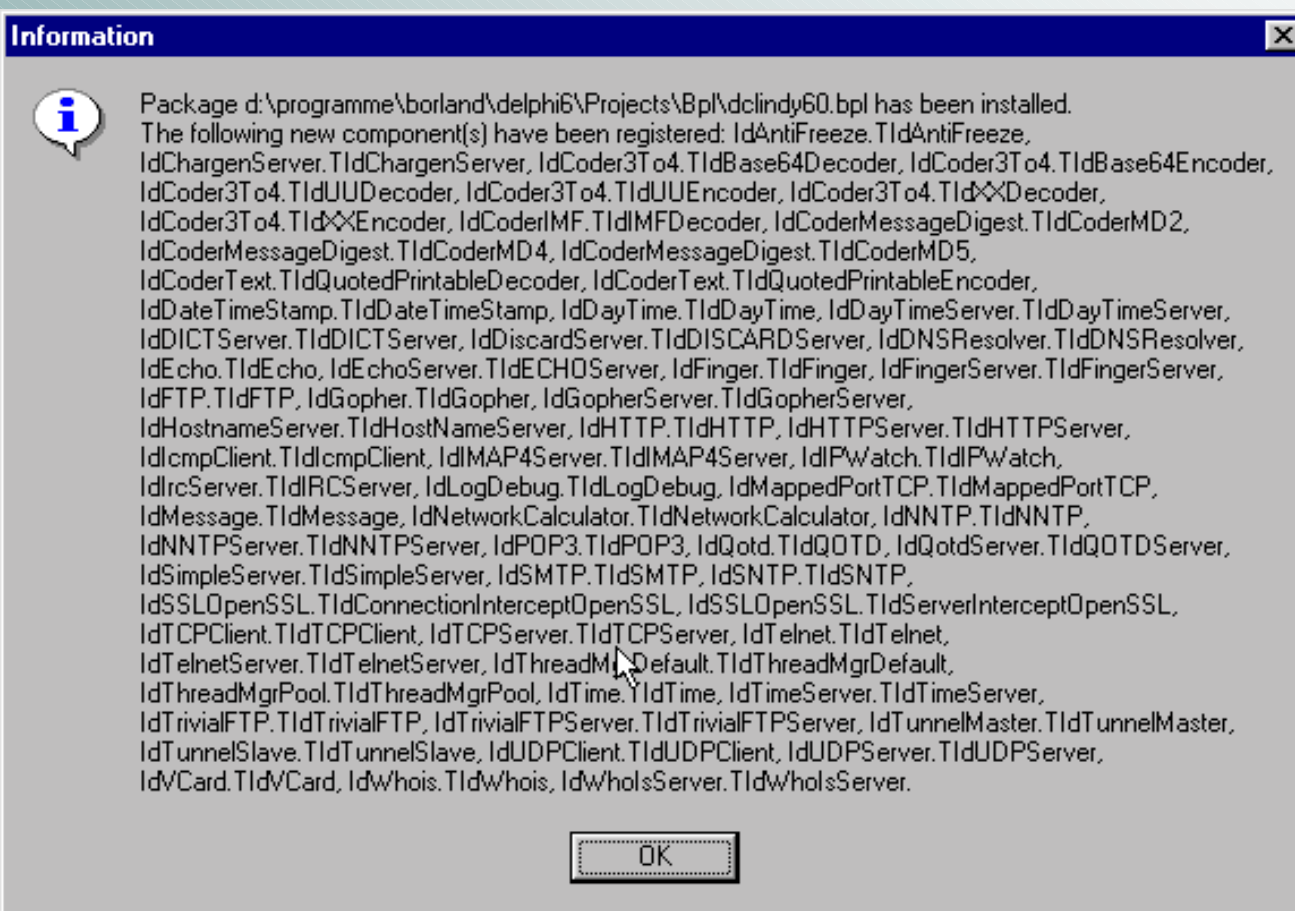
Indy Sockets - the Real Thing



Ich realisiere DWS mit den Indy-Komponenten, vor allem den TCPServer. Diese Komponente kapselt einen vollständigen TCPServer mit Multi-Thread-Unterstützung. Der TCPClient implementiert die Client-Funktionen des TCP-Protokolls einschließlich Sockets-Unterstützung. Dies kann man direkt oder in abgeleiteter Form für angepaßte Client-Anwendungen verwenden.

Die Indy Client-Komponenten sind einfach in der Handhabung, weil Sie Ihre Transaktionen in einer Folge schreiben und die Server multithread-fähig sind.

Indy Install

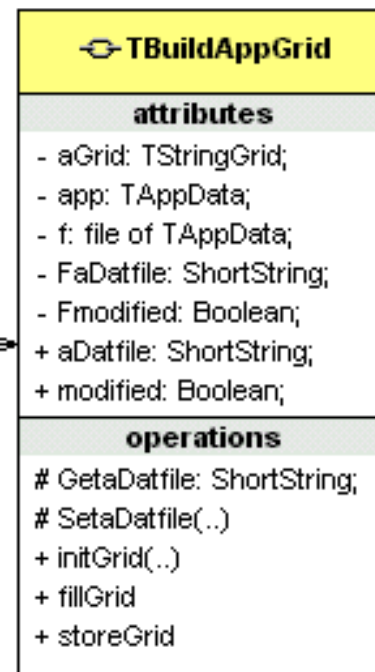
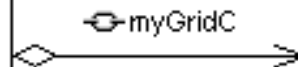
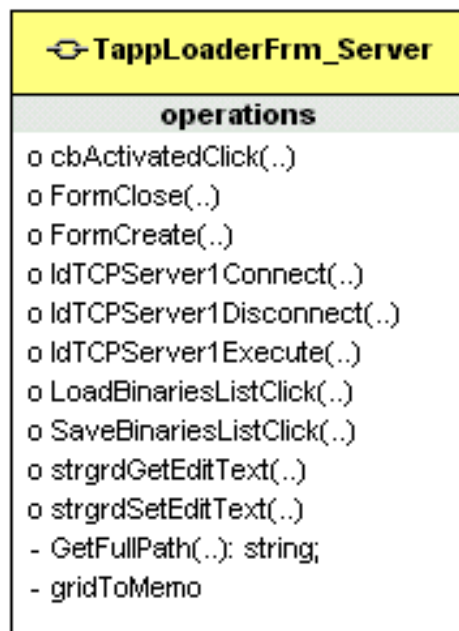


Die Implementierung



- Der DWS-Client erhält eine Liste der startfähigen Applikationen. Nach der Auswahl der gewünschten Anwendung startet der Benutzer die Transaktion, bis die Applikation auf dem Client transportiert ist und sich selbst startet.
- Ein Marshaling zwischen Source und Target wird nicht benötigt, da DWS den Source ja nach dem Laden auf dem Target startet und betreibt.
- *Unter Marshaling wird der Mechanismus verstanden, der es einem Client ermöglicht, Aufrufe von Schnittstellenfunktionen von Remote-Objekten in einem anderen Prozessraum oder auf einem anderen Computer durchzuführen.*

Der Client



TAppData
Name: string[50];
Size: longint;
Release: string[30];
descript: string[80];

Liebe auf den ersten Click



- Als erstes definiere ich zwei sogenannte „handlers“, die als Nachrichten korrespondieren müssen:

- `CTR_LIST = 'return_list';`
`CTR_FILE = 'return_file';`

- Ich erhalte ich die Liste der verfügbaren Dateien:

- ```
procedure TForm1.IdTCPServer1Execute(AThread:
TIdPeerThread);
// comes with writeline from client
if sRequest = CTR_LIST then begin
 for idx:= 0 to meData.Lines.Count - 1 do
 aThread.Connection.
 WriteLn(ExtractFileName(meData.Lines[idx]));
 aThread.Connection.WriteLine('::END::');
 aThread.Connection.Disconnect;
```

.....

# Threading auf dem TCPServer



- Bezüglich Threads sind zwei durch Indy selbst in den Umlauf gekommen, erstens einen Listener, der auf eine Verbindung wartet und den Servicethread, der den Transport einleitet. Diesbezüglich muß ich mich nicht um die Threadverwaltung kümmern, da der zugehörige Thread jeweils initialisiert übergeben wird:
- `IdTCPServer1Execute(AThread: TIdPeerThread)`
- Diese Technik ist mächtig, da der Client jederzeit eine Verbindung aufbauen kann, auch bei Dutzenden von bestehenden und aktiven Verbindungen zum Server.

# Datenbahn statt Autobahn



Der zweite Befehl `CTR_FILE` transportiert nun die Anwendung via Stream zum Client:

```
if Pos(CTR_FILE, sRequest) > 0 then begin
 iPos := Pos(CTR_FILE, sRequest);
 FileName := GetFullPath(FileName);
 if FileExists(FileName) then begin
 lbStatus.Items.Insert(0, Format('%-20s %s',
 [DateTimeToStr(now), 'Transfer starts ...']));
 FileStream := TFileStream.Create(FileName, fmOpenRead +
 fmShareDenyNone);
 aThread.Connection.OpenWriteBuffer;
 aThread.Connection.WriteStream(FileStream);
 aThread.Connection.CloseWriteBuffer;
 FreeAndNil(FileStream);
 aThread.Connection.Disconnect;
```

.....

# Load & Run

Mit Fork unter Linux



- Laden und Starten erfolgt auf Win wie auf Linux, den DWS ist eine CLX-Anwendung. Unter Windows werden aber Textzeilen mit CR/LF (d.h. ASCII 13 + ASCII 10) abgeschlossen, unter Linux mit LF. Der Code-Editor kann mit diesem Unterschied zwar umgehen, aber wenn Sie Code aus Windows importieren, sollten Sie diesem Punkt besondere Aufmerksamkeit schenken:
- ```
{$IFDEF LINUX}  
  execv(pchar(filename),NIL);  
  //libc.system(pchar(filename));  
{$ENDIF}
```

Call from Client



1. Folgend sehen sie noch den Client Aufruf mit der einfachen Konfiguration von Port und Host, zusätzlich benötigt der Client noch die Datei *qtintf.dll*:
2.

```
with IdTCPClient1 do begin
  if Connected then Disconnect;
  showStatus;
  Host:= edHost.Text;
  Port:= StrToInt(edPort.Text);
  Connect;
  WriteLn(CTR_LIST);
```

...

Warum es vorderhand keinen Browser braucht ?



Target	Receiver	Sender	Source
Programm	TCPClient	TCPServer	*.exe
Film	TCPClient	TCPServer	*.avi
Telnet	TCPClient	TCPConnection	Port: 9010
Form	DWSCClient	DWSServer	Monitor

Vorteil gegenüber Scripting



- OO-Technik mit Klassenbildung
- Direkt aus dem Code instanzierbar
 - ◆ Kein Browser erforderlich
 - ◆ Keine WebServer Konfiguration, Willen zur Einbindung anderer Hersteller und Plattformen
 - ◆ Borland erlaubt eine flexible Nutzung mehrerer Ideen und Welten (Indy und CLX)
 - ◆ Dank des Listendienstes immer up-to-date



Fragen und hoffentlich Antworten?

Aktuelle Quellen:

<http://max.kleiner.com/download/dws.zip>

