OpenEdge DataProvider für List & Label

Progress User Group – 10. April 2014 in München

Taste IT Consulting
Thomas Wurl
thomas.wurl@taste-consulting.de
www.taste-consulting.de

List & Label Anbindung an Open Edge

- Klassische Druckschleifen
  - Nur eine Tabelle
  - Druckschleife sendet Daten
  - Trick: „Satzformat“ Kennzeichen in den Daten zum Ansteuern verschiedener Zeilenformate.
  - Sehr unübersichtlich im Designer. Sehr aufwendig ☹
- Berichtcontainer
  - Tabellenmodell mit mehreren, auch geschachtelten Tabellen, Kreuztabellen und Grafiken etc.
  - Übersichtliche Darstellung im Designer.
  - Das Layout bestimmt wann welche Daten benötigt werden.
  - Der moderne Weg ☺.

Berichtscontainer mit DLL

List & Label Anbindung über DLL

- Definition
  - LIDbAddTable()
  - LIDbAddTableSortOrder()
  - LIDbAddTableRelation()
- Druck
  - LIPrintDbGetCurrentTable()
  - LIPrintDbGetCurrentTableSortOrder()
  - LIPrintDbGetCurrentTableRelation()
  - Besondere Druckschleife erforderlich.

Berichtscontainer mit .Net

- List & Label Anbindung über .Net Komponente

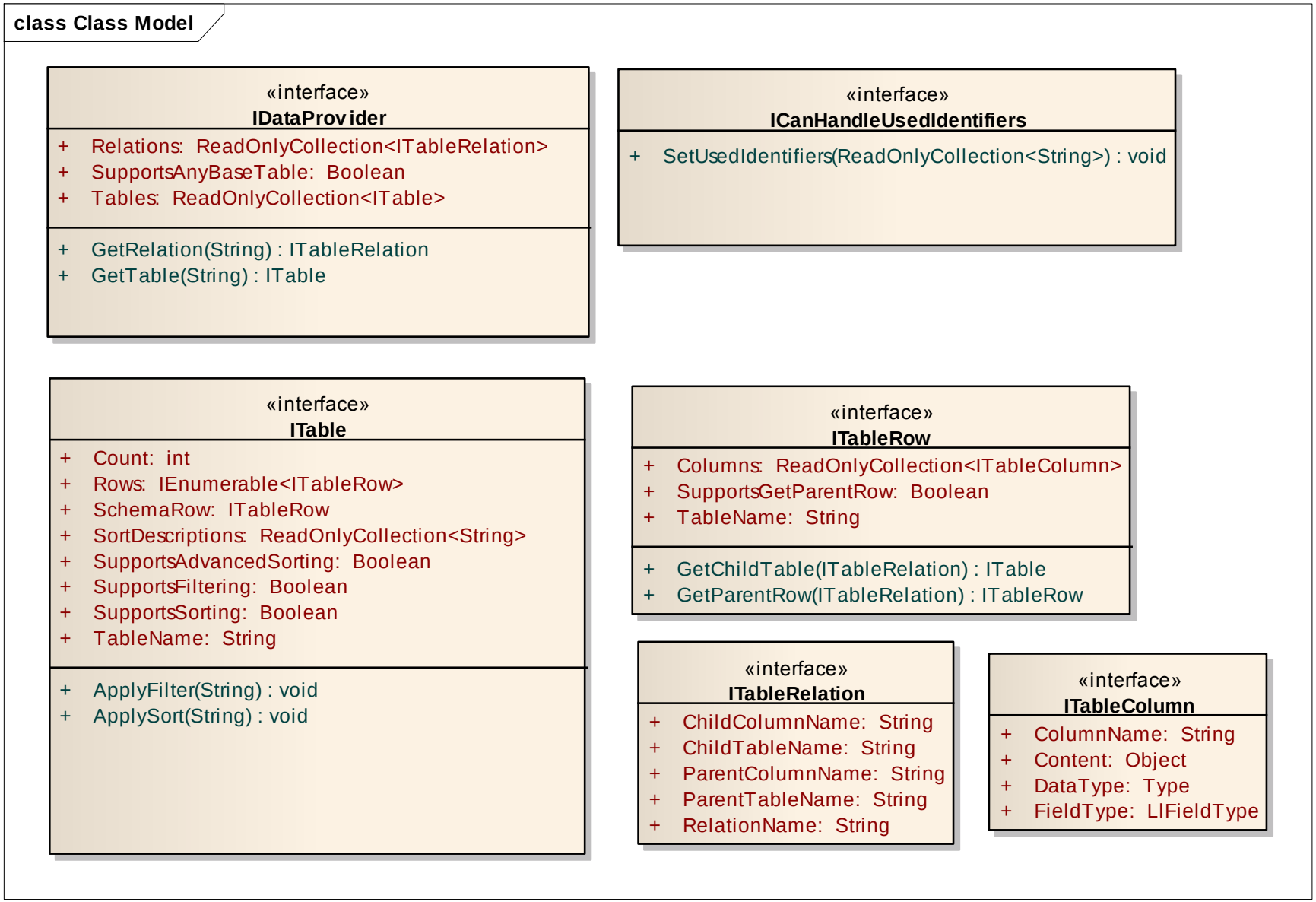
```
oLL = NEW combit.ListLabel19.ListLabel().
oLL:ForceSingleThread = TRUE.  /* OpenEdge - leider */
oLL:DataDataSource = <DataSource>.
oLL:Print().
```

- Data Source:
  - .Net Dataset
  - OpenEdge ProBinding Source
  - Data Provider – verfügbar für alle gängigen Datenbanken, XML, CSV, JSON, ….
  - Leider bislang keiner für OpenEdge
  - DataProvider Collection – Mehrere verschiedene Datenquellen in einem Report.

Was macht ein Data Provider?

- Registriert Tabellen, Spalten, Datentypen, Relationen, Sortierungen.
- Empfängt Tabellenname, Filter und Sortierungen und liefert Daten.
- Besteht aus Klassen, die List & Label Interfaces implementieren müssen.
- Die Programmierung erfordert .Net Generics. (OpenEdge 10, OpenEdge 11)

List & Label Data Provider Interfaces



Beispiel Table – ABL mit .Net Generics

```
USING combit.ListLabel19.DataProviders.*.
CLASS Table IMPLEMENTS ITable:
DEFINE PRIVATE VARIABLE moRows AS "List<ITableRow>" NO-UNDO.
DEFINE PUBLIC PROPERTY Rows AS "IEnumerable<ITableRow>" NO-UNDO
GET:
        openQuery().
        RETURN moRows:AsReadOnly().
END GET.
CONSTRUCTOR PUBLIC Table (…)
        SUPER().
        moRows = NEW List<ITableRow>().
        …
END CONTRUCTOR.
METHOD PRIVATE VOID openQuery(…):
        moRows:Clear().
        …
        moRows:Add (…).
        …
END.
END CLASS.
```

Probleme bei der Entwicklung des ABL Data Providers

- Klassen generiert – ließen sich manchmal Compilieren.
- Beim Implementieren kommen die ersten Probleme und Open Edge stürzt beim Complieren ab. (Architect, AppBuilder).
- Es gab dann aber eine Version, die sich Compilieren ließ.
- Funktionalität stimmte, aber immer Probleme mit der Anzahl der Rows. Ab ca. 1500 Rows –l und –s Fehler.
- -s stieg über alle Grenzen und harte Abstürze.

Probleme bei der Entwicklung des ABL Data Providers

- Code Review – sieht alles gut aus.
- Log-Manager Log – nicht auffälliges zu sehen.
- Progress Tech Support – keine Idee.
- Progress Development – bislang keine Antwort.
- Combit – Herr Bartlau – gemeinsames Debuggen der List & Label und OpenEdge Seite – kein Erfolg!
- Auslagern des Table Row Enumerators in .Net
  - -l Probleme reduziert
  - -s Probleme bleiben – magische Grenze ca. 1500 Rows.
- FRUST ☹!

Probleme bei der Entwicklung des ABL Data Providers

- Open Edge hat anscheinend Probleme mit Generics, die aber erst bei einer großen Anzahl Iterationen auftreten.
- Mir war es unmöglich, den Data Provider nur mit Open Edge zu implementieren.
- Dauer erste Implementierung – ca. 3 Tage.
- Dauer Fehlersuche, weitere Versuche, Tech Support etc. – mehrere Wochen.

Nächster Ansatz – eigene DLL

- Ziel des neuen Ansatzes mit .Net DLL:
  - Die List & Label Interfaces komplett aus Open Edge raushalten.
  - Klassen in .Net, die die Data Provider Interfaces implementieren.
  - Abstrakte Properties und Methoden in diesen Klassen, die in Open Edge implementiert werden müssen.
  - Performance- und Speicheroptimierung.
    - nur eine TableRow zu einer Zeit, eigener Enumerator
    - Used Identifiers berücksichtigen
    - usw.

Klassen des DataProviders

- .Net DLL

public abstract class AbstractDataProvider : IDataProvider, ICanHandleUsedIdentifiers

public abstract class AbstractTable : ITable, IEnumerable<ITableRow>

public class **TableRelation** : ITableRelation

public abstract class AbstractTableRow : ITableRow

public class **TableColumn** : ITableColumn

internal class TableRowEnumerator : IEnumerator<ITableRow>

- OpenEdge

CLASS **OpenEdgeDataProvider** INHERITS AbstractDataProvider

CLASS **OpenEdgeTable** INHERITS Abstract Table

CLASS **OpenEdgeTableRow** INHERITS Abstract TableRow

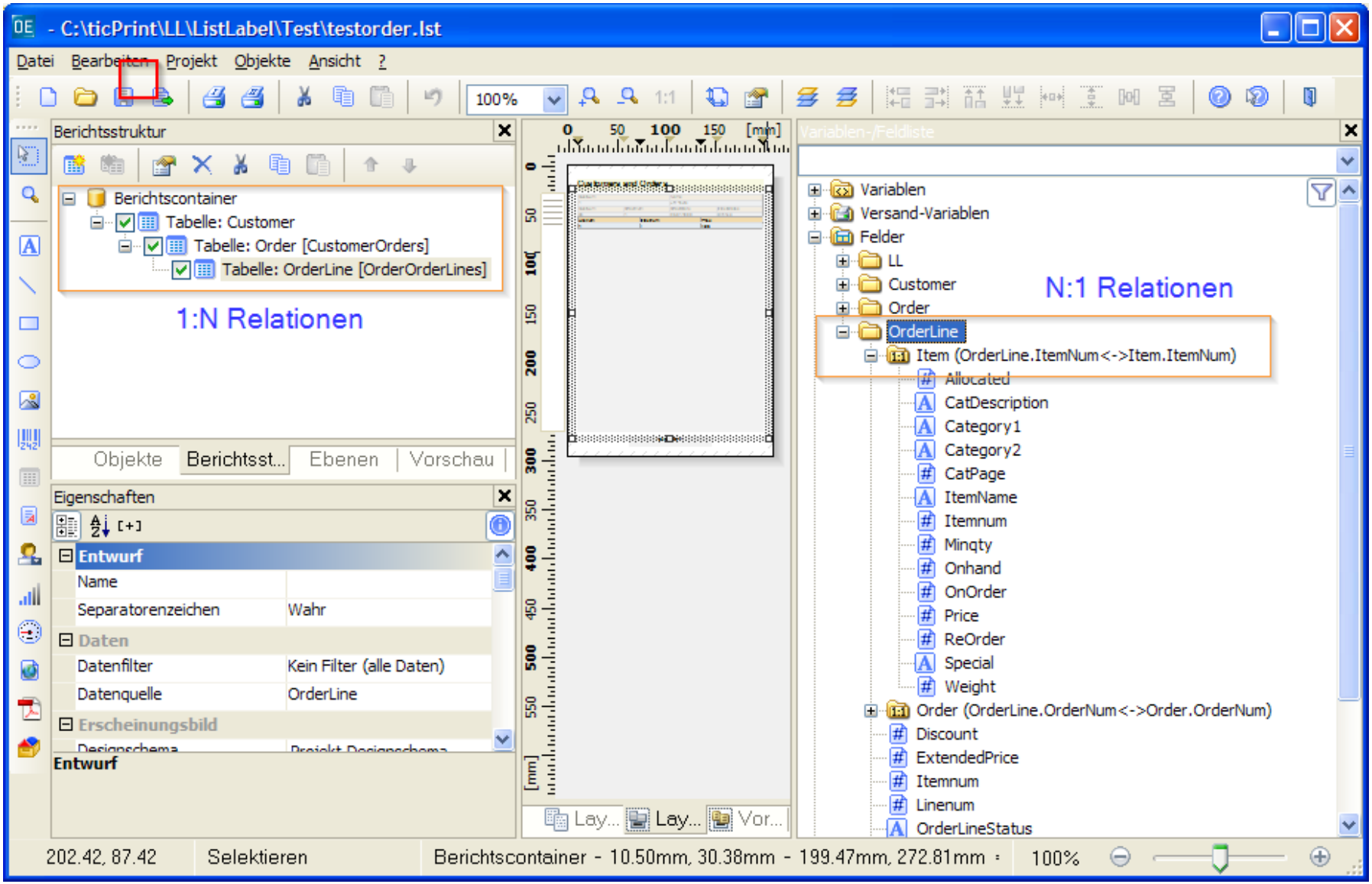
Data Provider – Eine Datenbank Tabelle

```
USING Progress.Lang.*.
USING combit.ListLabel19.*.
USING ListLabel.DataProviders.Base.*.

DEFINE VARIABLE oLL              AS CLASS ListLabel NO-UNDO.
DEFINE VARIABLE oProvider        AS CLASS OpenEdgeDataProvider NO-UNDO.
DEFINE VARIABLE oTable           AS CLASS OpenEdgeTable NO-UNDO.

oLL       = NEW ListLabel().
oProvider = NEW OpenEdgeDataProvider().

oTable = NEW OpenEdgeTable (oProvider, BUFFER Customer:HANDLE).
oTable:BaseQueryWhere = "CustNum < 10".
oProvider:addTable( oTable ).

oLL:DataSource = oProvider.
oLL:ForceSingleThread = TRUE.

oLL:Design().
```

Data Provider – Mehrere DB Tabellen und Relationen

```
USING Progress.Lang.*.
USING combit.ListLabel19.*.
USING ListLabel.DataProviders.Base.*.

DEFINE VARIABLE oLL            AS CLASS ListLabel NO-UNDO.
DEFINE VARIABLE oProvider      AS CLASS OpenEdgeDataProvider NO-UNDO.
DEFINE VARIABLE oTable         AS CLASS OpenEdgeTable NO-UNDO.

oProvider = NEW OpenEdgeDataProvider().

oTable = NEW OpenEdgeTable (oProvider, BUFFER Customer:HANDLE).
oTable:BaseQueryWhere = "CustNum = 1".
oTable:addSortOrder("CustNum","BY CustNum").
oTable:addSortOrder("Name","BY Name").

oProvider:addTable( oTable ).

oTable = NEW OpenEdgeTable (oProvider, BUFFER Order:HANDLE).
oProvider:addTable( oTable ).
oTable = NEW OpenEdgeTable (oProvider, BUFFER orderline:HANDLE).
oProvider:addTable( oTable ).
oTable = NEW OpenEdgeTable (oProvider, BUFFER Item:HANDLE).
oProvider:addTable( oTable ).

oProvider:addTableRelation("CustomerOrders","Customer","Order","CustNum","CustNum").
oProvider:addTableRelation("OrderOrderLines","Order","OrderLine","OrderNum","OrderNum").
oProvider:addTableRelation("OrderLineItem","Item","OrderLine","ItemNum","ItemNum").

oLL = NEW ListLabel().
oLL:ForceSingleThread = TRUE.
oLL:DataSource = oProvider.
oLL:Design().
oLL:Dispose().
```

Relationen im Designer

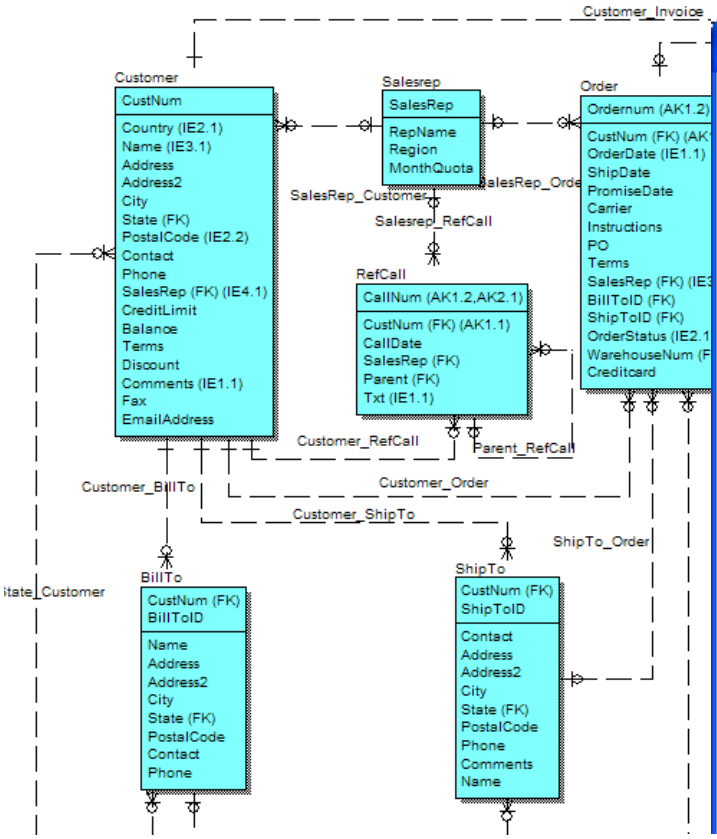


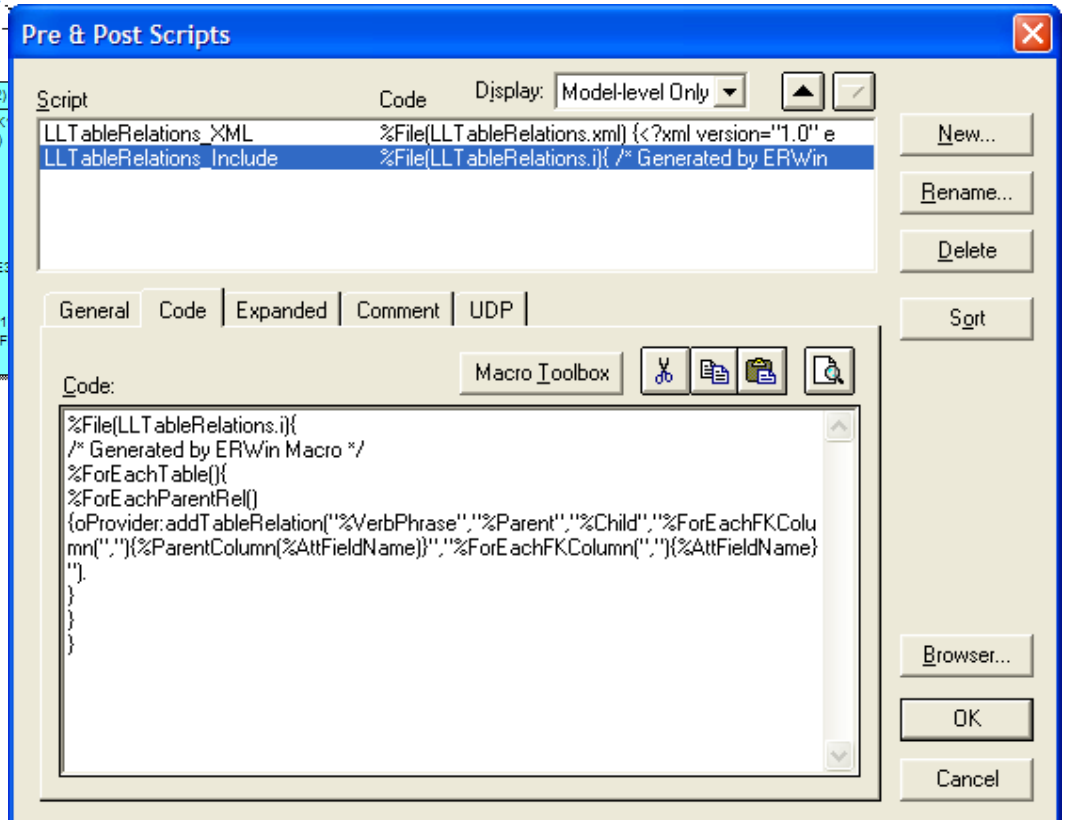
DataProvider und ProDataset

- Bei einem DATASET können List & Label Relationen aus den DATA-RELATIONS ermittelt werden.
- Eine DATA-RELATION hat keinen Typ wie „1:N“ oder „N:1“.
- Man kann den Typ aber über RELATION-FIELDS und <Buffer>:KEYS ableiten.
- Eigene Klasse **OpenEdgeTableSchema**, die man dem Provider übergibt und die der Provider auch intern verwendet.

```
/* Schema extern definiert */
oTableSchema = NEW OpenEdgeTableSchema().
oTableSchema:registerDataset(hDataset).
oProvider:attachDataset(hDataset,oTableSchema).

/* Schema wird vom Provider erstellt */
oProvider:attachDataset(hDataset,oTableSchema).
```

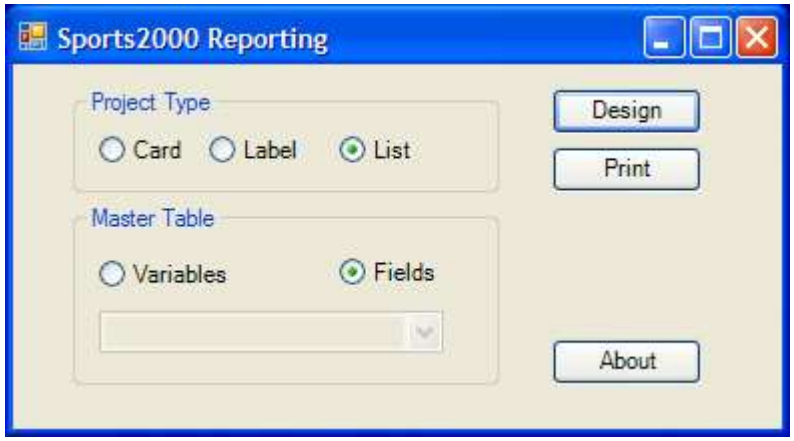
DB Reverse Engineering mit ERWin






```
/* Generated by ERWin Macro */
addTableRelation("BillTo_Order","BillTo","Order","CustNum,BillToID","CustNum,BillToID").
addTableRelation("Bin_InventoryTrans","Bin","InventoryTrans","BinNum","BinNum").
addTableRelation("Customer_RefCall","Customer","RefCall","CustNum","CustNum").
addTableRelation("Customer_Invoice","Customer","Invoice","CustNum","CustNum").
addTableRelation("Customer_ShipTo","Customer","ShipTo","CustNum","CustNum").
addTableRelation("Customer_BillTo","Customer","BillTo","CustNum","CustNum").
addTableRelation("Customer_Order","Customer","Order","CustNum","CustNum").
addTableRelation("Department_Employee","Department","Employee","DeptCode","DeptCode").
addTableRelation("Employee_Timesheet","Employee","TimeSheet","EmpNum","EmpNum").
addTableRelation("Employee_Vacation","Employee","Vacation","EmpNum","EmpNum").
addTableRelation("Employee_Benefits","Employee","Benefits","EmpNum","EmpNum").
```

DEMO - Sports 2000 Reporting Spielwiese



- Sports2000DataProvider
  - Registriert alle Tabellen über Meta Schema
  - Registiert für alle Felder SortOrders (ASC / DESC)
  - Registiert alle Relationen. Reverse Engineering Sports 2000 mit ERWin, Relationen eingetragen, Makro zum Generieren von Code für Relationen.
  - Gut zum Testen von List & Label Features ohne eine Zeile Code…

Wie kommt man an den Data Provider?

- Freie Open Source Version – kein Support.
  - Download von www.taste-consulting.de.
- Unterstützung im Rahmen von Consulting
  - Individuelle Versionen.
  - Integration in ihre Applikation.
  - …
- ticPrint by Taste IT Consulting
  - Professionelle Drucklösung für OpenEdge.

Fragen