AdsClass++

Arbeiten mit dem Advantage Database Server

AdsClass++ ist eine objektorientierte Anbindung des ADS DBMS an eine Xbase++Applikation. Die Kommunikation erfolgt auf Basis der ADS API und benötigt keine Alaska ADSDBE. Alle Tabellen oder SQL Abfragen werden in Objekten verwaltet.

Vorteile von AdsClass++:

- schnell, da auf der ADS API basierend
- unterstützt alle ADS Versionen, auch local engine
- unterstützt alle ADS Tabellenformate: DBF,CDX,VFP,ADT
- unterstützt alle Datentypen
- unterstützt Rawkey Indices
- unterstützt alle weiteren ADS Funktionalitäten
- einfachste Integration von SQL
- Thread save oder Thread übergreifend verwendbar
- Zugriff ausschließlich durch Objekte ohne Workarea
- Developer Version mit Sourcecode
- Englische und deutsche Dokumentation

Die Grundidee dieser Bibliothek ist die Kapselung einer Tabelle in einem Objekt. Xbase++ kann zur Laufzeit dynamisch Klassen definieren und dazu Objekte er-zeugen. Die Tabellenstruktur wird dabei in ACCESS ASSIGN Methoden abgebildet. Für die Navigation, Suche, Filterung, Scopes, etc. stehen Methoden zur Verfügung. Jeder Xbase++ Dateibefehl wird durch eine gleichlautende Methode ersetzt. Zusätzlich wird der Befehlsumfang von Xbase++ um produktive Methoden erweitert.

Ein weiterer Vorteil der Objekte ist die Typisierung. Sie können wie alle anderen Variablen eingesetzt werden, als Parameter weitergegeben oder validiert werden, etc.
Einzig durch das Tabellenobjekt wird mit dem ADS kommuniziert, ohne dass eine Workarea benötigt wird.
Daher bietet sich diese Klasse weniger für die Konvertierung alten CLIPPER Codes als vielmehr für Neuentwicklungen an.

Unsere Erfahrung mit Tabellenobjekten zeigt, dass der Code leichter lesbar, besser pflegbar und weniger fehleranfällig in Bezug auf Tabellenzugriffe ist. Insbesondere in einer MDI Anwendung wird hier die Zurodnung einer Tabelle zu einem Dialog erleichtert.

Dieses Konzept wird in **XClass++** für alle Xbase++ DBEs und Arrays realisiert. **AdsClass++** integriert sich nahtlos in **XClass++,** kann auch unabhängig davon eingesetzt werden.
Tabellenobjekte entkoppeln eine Applikation somit auch von der Datenzugriffs-Technik.

Hier die wichtigsten Komponenten im Überblick:

- **dsAceSession**, eine Verbindung zu einem ADS Server herstellen und verwalten, für weitere Verbindungen zum gleichen oder verschiedenen ADS Servern muss ein neues Objekt erzeugt werden.

```
adsConnection := dsAceSession():New()
adsConnection:connect(„\\myServer“)
? adsConnection:IsConnected()
AdsSetDefault(„\\myServer\myShare\myDir“)

dbTable :=
OpenAceTable(„Kunde“,adsConnection )
dbTable:close()
adsConnection:disconnect()
```

- **dsAceDD**, Verbindung zu einem ADS Datadictionary aufbauen.

```
adsDD := dsAceDD():New()
adsDD:connect(„\\myServer\myShare\myDir“)
? adsDD:IsConnected()
dbTable := adsDD:OpenTable(„Kunde“ )
dbTable:gotop()
```

Hiermit wird auch ein ADD gepflegt:

- Anlegen von Gruppen und Benutzern
- Transaktionsverwaltung
- Pflege von Tabellen, Indices, Views, SQL-Funktionen und stored procdures
- Öffnen applikationsweit verfügbarer Tabellen

➢ **Transaktion**

Je Verbindung (dsAceSession oder dsAceDD) kann eine Transaktion gestartet werden.

```
adsConnection:BeginnTransaction()
beginn sequence
   ...
   adsConnection:CommitTransaction()
recover
   adsConnection: RollBackTransaction()
end sequence
```

➢ **dsAceTable**, Ableitungen davon werden tabellenspezifisch erzeugt und repräsentieren eine Tabelle. Tabelle kann dabei eine physikalische Tabelle, eine SQL Abfrage, ein View oder das Ergebnis einer stored procedure sein. Es erfolgt hier keine Unterscheidung in der Erzeugung oder Verwendung.

o Dateiöffnung

```
dbTable := OpenAceTable(„Kunde“,
     adsConnection/adsDD )
dbTable:gotop()
? dbTable:KDNR                  // 4711
if dbTable:rlock()
   dbTable:KDNR := 4712
   dbTable:unlock()
endif
? dbTable:alias                 // KUNDE01
? dbTable:tablename()           // KUNDE
do while !dbTable:eof()
   ….
   dbTable:skip()
enddo
dbTable:cose()
```

o Indizieren

```
// NTX
dbTable:ordcreate(„NANME“,,;
   “upper(name)+Upper(vorname)“)
// CDX
dbTable:ordcreate(,“NAME“,;
   “upper(name)+Upper(vorname)“)
// ADT, Feldtyp CiCharacter, Rawkey Index
dbTable:ordcreate(,“NAME“, „name;vorname“)
// absteigend sortiert
dbTable:ordcreate(,“NAME“,;
   „name;vorname“,,ADS_DESCENDING )
// oder mit spezieller Collation
dbTable:ordcreate(,“NAME“,„name;vorname“;
   ,,,,“GERMAN_VFP_CI_AS_1252“ )
```

Rawkey Index sind binär und können alle Datentypen mixen, sind dadurch aber extrem performant!

o und die entprechende Suche

```
// NTX, CDX
dbTable:ordsetfocus(„NAME“)
dbTable:seek(upper(padr(„Müller“,20))+;
   upper(padr(„Alfred“,20)))
// ADT, Rawkey Index
dbTable:seek({„Müller“,„Alfred“})
? dbTable:found()
```

o Filter und Scope:

```
// wird auf dem Client ausgewertet
dbTable:setfilter({|db| myFilter(db)})
// wird auf dem Server ausgwertet
dbTable:setfilter(,“Kunde=‘Maier‘“)
// wird über Index optimierzt
dbTable:setaof(“Kunde=‘Maier‘“)
// Top und Bottom Scope
dbTable:Setscope(, „BERLIN“)
```

➢ **SQL** (Kunde kann ein Tabelle oder ein View sein)

```
dbTable := OpenSqlTable(,“select * from”+;
  “ kunde where name like ‘Schmid%’”,;
   adsConnection )
```

➢ **SQL** mit Parameter

```
sqlConn :=;
   adsSqlTable():New(adsConnection)
sqlConn:Prepare(;
   "SELECT * from Kunde where ort=:P1")
sqlConn:SetParam("P1", "Berlin")
dbTable := sqlConn:Execute2Server()

sqlConn:SetParam("P1", "Hamburg")
dbTable := sqlConn:Execute2Server()
sqlConn:Close()
```

Das Tabellenobjekt eines SQL Ergebnisses kann wiederum indiziert werden!

➢ **SQL stored procedures**

Gibt eine stored procedure eine Ergebnismenge zurück, wird diese wie jede andere Tabelle behandelt

```
sqlConn :=;
   adsSqlServer():New(adsConnection)
dbTable := sqlConn:SQL2Table(;
   "execute procedure sp_MyProc()")
```

➢ **dsAceMg**, ADS Management für Anzeige und Pflege der ADS Einstellungen und Parameter

o Anzeige angemeldeten Benutzer
o Anzeige der geöffneten Tabellen
o Anzeige der gesperrten Sätze mit dem dazugehörenden Benutzer
o Disconnect von Benutzern

➢ **dsAdsListening**, Klasse für Notifikationen vom ADS Server empfangen und verarbeiten. In einem Thread werden Notifikation, welche zuvor angemeldet wurden, empfangen und verarbeitet.

```
// initialisieren im MAIN
AdsListening(dsAceDD)
// Dialogfenster mit Browser erstellen
dlgWindow := xbpDialog():New(...)
...
// events registrieren, browser wird
automatisch // refreshed, wenn ein Satz
geschrieben wurde
AdsListening():CreateEvent(dlgWindow,,;
  "ARTICLE.UPDATE ",;
  {|| dlgWindow:oBrowse:RefreshAll()})
```

Mehr Information zu ADS finden Sie unter www.sybase.com/products/databasemanagement

Eine Demoversion mit vollständiger Funktionalität, Dokumentation und Beispielprogrammen kann von www.ds-datasoft.de heruntergeladen werden.

Kontakt:
DS-Datasoft GmbH&Co.KG
An der Kirche 5
D-87654 Friesenried
Tel. 08347 981370
info@ds-datasoft.de