cSqlExec

presentation

Scanduc | 21

The cSqlExec class

“Stand alone” package cSqlExec.pkg

Developed by DAE for internal use

Enhanced for use with the Plato sample project

Purpose:

1. Make embedded SQL calls to the database backend
2. Method framework for building SQL syntax

Scanduc

Declare a cSqlExec object (SqlExecPlato.pkg)

```
Use cSqlExec.pkg

Open Order // known to reside in the Plato database

Global_Variable Handle ghoSqlPlato // <- Access object via this name

{Visibility=private}
Object oPlatoSqlExec is a cSqlExec
    Set phFollowTable to Order.File_Number // connect to database housing the Order table
    Move Self to ghoSqlPlato
End_Object
```

Scanduc

Executing SQL - Get ExecuteSql

```
Procedure HelloSqlExec
    Integer iRow iMax
    String[][] aCities

    Get ExecuteSql of ghoSqlPlato "SELECT DISTINCT [Customer].[City] FROM [Customer]" to aCities

    Move (SizeOfArray(aCities)-1) to iMax
    For iRow from 0 to iMax
        Showln aCities[iRow][0]
    Loop
End_Procedure
```

Scanduc

Executing SQL - Send ExecuteSql

```
Function OnFetchRow Handle hoStmt String[] aColumnNames Returns tWebRow
    tWebRow stRow

    Get SQLColumnValue of hoStmt 1 to stRow.sRowId // hoStmt is a cSQLStatement object.
    Get SQLColumnValue of hoStmt 1 to stRow.aCells[0].sValue
    Get SQLColumnValue of hoStmt 2 to stRow.aCells[1].sValue

    Function_Return stRow
End_Function

Procedure OnManualLoadData tWebRow[] ByRef aTheRows String ByRef sCurrentRowID
    String sSql
    Move "SELECT [Customer].[Number], [Customer].[Name] from [Customer]" to sSql
    Send ExecuteSql of ghoSql sSql Self (RefFunc(OnFetchRow)) (&aTheRows)
End_Procedure
```

this can be an array of anything

Scanduc

Another way

Hello cSqlExec - the other way

```
Procedure HelloSqlExec
    Handle hoSql
    tSqlStmt stStmt // Represents a SELECT statement
    String sSql
    String[][] aCities

    Move ghoSqlPlato to hoSql

    // "SELECT DISTINCT [dbo].[Customer].[City] FROM [dbo].[Customer]"
    Get NewStmt of hoSql (SqlTableName(hoSql,Customer.File_Number)) to stStmt
    Send AddStmtColumn of hoSql (&stStmt) File_Field Customer.City
    Move True to stStmt.bDistinct

    Get StmtToString of hoSql stStmt to sSql

    Get ExecuteSql of hoSql sSql to aCities
    ....
End_Procedure
```

Scanduc

cSqlExec - Filters and Select statements

- Filter Expressions
  - Represented as string
  - Used for boolean expressions
- Select statements
  - Represented as a tSqlStmt value
  - Statements can be converted to strings
    - The string can be executed (ExecuteSQL)
    - And thereby used as part of expressions
- General expressions
  - Used for expressions of any type
  - Reverse Polish notation kind of interface
- Utility functions

Scanduc

SqlFilterColumn function

Returns SQL syntax for a simple filter on a column:

Definition:

```
Function SqlFilterColumn Integer iTable Integer iColumn String sComp String sConstant Returns String
```

```
where sComp is “=” “<=” “like” “in” ...
```

Example:

```
Get SqlFilterColumn File_Field Customer.Customer_Name “<>” “Sture ApS” to sFilter
```

will return

```
[dbo].[Customer].[Customer_Name] <> N’Sture ApS’
```

Scanduc

Other functions returning a filter expression

Filter by expression:

```
Function SqlFilterColumn Integer iTable Integer iColumn String sComp String sConstant Returns String

Function SqlFilterColumnExpr Integer iTable Integer iColumn String sComp String sExpr Returns String

Call: Get SqlFilterColumnExpr file_field Order.Order_Date ">=" "getdate()" to sFilter

Result: [dbo].[Order].[Order_date] >= getdate()
```

Filter by parent dataflex relation:

```
Function SqlFilterParentConstrain Integer iChild Integer iParent Boolean bUseParentRecordBuffer Returns String

Call: Get SqlFilterParentConstrain Order.file_number Customer.file_number False to sFilter

Result:                                  [dbo].[Order].[Customer_Number] = [dbo].[Customer].[Customer_Number]

Result if bUseParentRecordBuffer is True: [dbo].[Order].[Customer_Number] = 78
```

Scanduc

Combining filter expressions

Append two filters

```
Procedure SqlFilterConcatExpr String ByRef sFilter String sLogicOp String sExpr

        Send SqlFilterConcatExpr (&sFilter) “OR” sFilter2

        [dbo].[Customer].[Zip] = N’4000’ OR [dbo].[Customer].[Zip] = N’8000’
```

Append filter to string

```
Procedure SqlFilterConcat String ByRef sFilter String sLogicOp Integer iTable Integer iColumn String sComp String sValue

        Send SqlFilterConcat (&sFilter) “OR” File_Field Customer.Zip “=” “8000”

        [dbo].[Customer].[Zip] = N’4000’ OR [dbo].[Customer].[Zip] = N’8000’
```

Scanduc

Filtering IRL

```
Procedure OnConstrain // DataDictionary event
     ...
       Get EscapeLikeString of hoSql sSearchVal to sSearchVal // \ % _ [
       Move ("%"+sSearchVal+"%") to sPattern

       Send SqlFilterConcat of hoSql (&sFilter) "or" File_Field CarOwner.Adresse "like" sPattern
       Send SqlFilterConcat of hoSql (&sFilter) "or" File_Field CarOwner.Zipcode "like" sPattern
       Send SqlFilterConcat of hoSql (&sFilter) "or" File_Field CarOwner.CVR_Num "like" sPattern
       Send SqlFilterConcat of hoSql (&sFilter) "or" File_Field CarOwner.Name    "like" sPattern

       Set psSQLFilter to sFilter
       Set pbUseDDSQLFilters to True
End_Procedure
```

Scanduc

Filtering IRL - 2 (from SecMod log list)

```
If (not(IsNullDateTime(stFilter.dtFrom))) Begin
    Get SqlFilterColumn of hoSql File_Field SecModLog.LogDT ">=" stFilter.dtFrom to aFilters[SizeOfArray(aFilters)]
End
If (not(IsNullDateTime(stFilter.dtTo))) Begin
    Get SqlFilterColumn of hoSql File_Field SecModLog.LogDT "<=" stFilter.dtTo to aFilters[SizeOfArray(aFilters)]
End
If (stFilter.sProgramFilter<>"") Begin
    Get SqlFilterColumn of hoSql File_Field SecModLog.Program "like" ("%"+stFilter.sProgramFilter+"%") to aFilters[SizeOfArray(aFilters)]
End

If (SizeOfArray(stFilter.aTargetUsers)>0) Begin
    Get SqlFilterColumnExpr of hoSql File_Field SecModLog.TargetUserUuid "in" (SqlArray(hoSql,stFilter.aTargetUsers,typeString)) to aFilters[SizeOfArray(aFilters)]
End
If (SizeOfArray(stFilter.aEditedByUser)>0) Begin
    Get SqlFilterColumnExpr of hoSql File_Field SecModLog.EditByUserUuid "in" (SqlArray(hoSql,stFilter.aEditedByUser,typeString)) to aFilters[SizeOfArray(aFilters)]
End
If (SizeOfArray(stFilter.aEventTypes)>0) Begin
    Get SqlFilterColumnExpr of hoSql File_Field SecModLog.EventType "in" (SqlArray(hoSql,stFilter.aEventTypes,typeString)) to aFilters[SizeOfArray(aFilters)]
End
If (SizeOfArray(stFilter.aStatusValues)>0) Begin
    Get SqlFilterColumnExpr of hoSql File_Field SecModLog.AuxValue "in" (SqlArray(hoSql,stFilter.aStatusValues,typeString)) to aFilters[SizeOfArray(aFilters)]
End

Get SqlAppendExpressions of hoSql aFilters "AND" True to sFilter
```

Scanduc

utility functions

- Function **SqlColumnName** Integer iTable Integer iColumn Returns String
  - “[dbo].[customer].[customer_name]”
- Function **SqlTableName** Integer iTable Returns String
  - same as above
- Function **AddParenthesis** String sExpr Returns String
  - If sExpr is not empty parenthesis will be added to the returned value
- Function **SqlConstant** String sValue Integer eType Returns String
  - aTypr = typeString, typeDateTime, typeDate, typeNumber

- Function **SqlArray** String[] aOperands Integer eType Returns String
  - eType = typeString, typeDateTime, typeDate, typeNumber
  - (3.14, 2.72, 9.81)
- Function **SqlAppendExpressions** String[] aExpr String sJoinByOperator Boolean bParenthesis Returns String
  - expr1 AND expr2 AND expr3 AND expr4 …
  - sJoinByOperator should be AND, OR, +

Scanduc

SQL Select statements

SELECT inventory.Item_ID, inventory.Description, vendor.Name from inventory

LEFT JOIN vendor ON inventory.Vendor_ID = vendor.ID

WHERE Vendor.Zip <> N'9500'

```
tSqlStmt stStmt
Get NewStmt of hoSql (SqlTableName(hoSql,Inventory.File_Number)) to stStmt
Send AddStmtJoinAuto of hoSql (&stStmt) Inventory.File_Number Vendor.File_Number C_SQL_LEFT_JOIN
Send AddStmtColumn of hoSql (&stStmt) File_Field Inventory.Item_ID
Send AddStmtColumn of hoSql (&stStmt) File_Field Inventory.Description
Send AddStmtColumn of hoSql (&stStmt) File_Field Vendor.Name
Send AddStmtFilter of hoSql (&stStmt) File_Field Vendor.Zip "<>" "9500"
Get StmtToString of hoSql stStmt to sSql
Send ExecuteSQL of hoSql sSql ...
```

Scanduc

Building a statement

```
Function NewStmt String sFrom Returns tSqlStmt

// Add the necessary join segments to join to iParentTable from iChildTable
Procedure AddStmtJoinAuto tSqlStmt ByRef stStmt Integer iChildTable Integer iParentTable Integer eJoinMode

// “Manual” join
Procedure AddStmtJoin tSqlStmt ByRef stStmt String sTable Integer eJoinMode
Procedure AddStmtJoinSegment tSqlStmt ByRef stStmt Integer iFromTable Integer iFromColumn ;
                                                    Integer iParentTable Integer iParentColumn
Procedure AddStmtJoinSegmentExpr tSqlStmt ByRef stStmt Integer iFromTable Integer iFromColumn String sComp String sExpr

Procedure AddStmtColumn tSqlStmt ByRef stStmt Integer iTable Integer iColumn String sOptAlias
Procedure AddStmtFilter tSqlStmt ByRef stStmt Integer iTable Integer iColumn String sComp String sValue
Procedure AddStmtOrderBy tSqlStmt ByRef stStmt Integer iTable Integer iColumn Boolean bDescending

// The ‘Expr’ suffix means that sExpr is inserted into the statement unprocessed)
Procedure AddStmtColumnExpr tSqlStmt ByRef stStmt String sExpr String sOptAlias // <-- LOOK AT ME!
Procedure AddStmtFilterExpr tSqlStmt ByRef stStmt String sExpr
Procedure AddStmtOrderByExpr tSqlStmt ByRef stStmt String sExpr Boolean bDescending

Function StmtToString tSqlStmt stStmt Returns String // Get statement as string
```

Scanduc

General expressions (Reverse Polish Notation)

```
Procedure PushOperandExpr String[] ByRef aOperands String sExpr
Procedure PushOperandConstant String[] ByRef aOperands String sConstant Integer eType
Procedure PushOperandTableColumn String[] ByRef aOperands Integer iTable Integer iColumn
Procedure ApplyOperator String[] ByRef aOperands String sOperator

Procedure AddParenthesisTopOperand String[] ByRef aOperands   // Add parenthesis to top of stack
Function TopOperand String[] aOperands Returns String         // Return top of stack
Function PopOperand String[] ByRef aOperands Returns String

String[] aOperands
Send PushOperandTableColumn of hoSql (&aOperands) File_Field Order.Amount
Send PushOperandTableColumn of hoSql (&aOperands) File_Field Order.VAT
Send ApplyOperator of hoSql (&aOperands) "+"     // top:  [Order].[Amount] + [Order].[VAT]
Send PushOperandConstant of hoSql (&aOperands) 33 typeNumber
Send ApplyOperator of hoSql (&aOperands) "+"     // top:  [Order].[Amount] + [Order].[VAT] + 33
Send AddParenthesisTopOperand of hoSql (&aOperands) // Add () to top element
Get TopOperand of hoSql aOperands to sExpr       //       ([Order].[Amount] + [Order].[VAT] + 33)
```

Scanduc

When is cSqlExec useful?

Not always.

But when you ...

- Use cSqlExec to produce fragments of SQL syntax and insert in existing strings
- Develop “flexible” query interfaces on your database.
- Develop general purpose logic without prior knowledge of the database

Scanduc

Showtime

Demo WebOrderMobile - FilterBuilder

Demo WebOrderMobile - DDSqlExtractor

Scanduc

The end

Scanduc

Bonus slide - joining same table twice

```
Get NewStmt of hoSql (SqlTableName(hoSql,ItemApproval.File_Number)) to stStmt

Send AddStmtJoin        of hoSql (&stStmt) (SqlTableName(hoSql,WebAppUser.File_Number)) C_SQL_LEFT_JOIN "A"
Send AddStmtJoinSegment of hoSql (&stStmt) File_Field ItemApproval.UserName1 File_Field WebAppUser.LoginName

Send AddStmtJoin        of hoSql (&stStmt) (SqlTableName(hoSql,WebAppUser.File_Number)) C_SQL_LEFT_JOIN "B"
Send AddStmtJoinSegment of hoSql (&stStmt) File_Field ItemApproval.UserName2 File_Field WebAppUser.LoginName

Send AddStmtColumn      of hoSql (&stStmt) File_Field ItemApproval.ItemType

Send AddStmtColumn      of hoSql (&stStmt) File_Field ItemApproval.UserName1
Send AddStmtColumnExpr  of hoSql (&stStmt) "A.FullName"

Send AddStmtColumn      of hoSql (&stStmt) File_Field ItemApproval.UserName2
Send AddStmtColumnExpr  of hoSql (&stStmt) "B.FullName"
```

Scanduc