

2004/7/20

Series2

Team Developer の生産性におけるレポート

Gupta サポート

A.WAKAMATSU

今回は、前回簡単にご説明した SQL 自動生成の機能を持った関数について、詳しく説明していきたいと思います。

これから、説明していく SQL 自動生成関数は、TeamDeveloper の最大の特徴である、オブジェクト指向、第 4 世代言語を利用して作成した関数です。

この関数を実作業で取り入れることができれば、何倍もの生産性と保守性を向上させることができます。

TeamDeveloper アプリケーションは SAL (Scalable Application Language) という独自の言語で記述します。SAL 言語はインデントにより処理ブロックを規定していて、コーディングを自動生成してくれます。これからコーディングを紹介していくにあたって、SAL が自動生成してくれる部分は細字、プログラマー自ら記述、もしくはコーディングアシスタントで選択する部分は太字で記述していきます。

1. 通常の SQL 作成

VisualBasic では ADO (ActiveX Data Object) を利用して、データを取得し、画面に表示させる方法を紹介しています。直接 Oracle のデータをフィールドにセットしていますが、実際は Access 等のデータベースを介することの方が多いと思います。

'VisualBasic

```
Private Sub Command1_Click()  
  
    Dim Cnn      As New ADODB.Connection      'ADO コネクション オブジェクト  
    Dim Rec      As New ADODB.Recordset      'ADO レコードセット オブジェクト  
    Dim sSql     As String                    'SQL 格納用  
  
    'SQL の作成  
    sSql = ""  
    sSql = sSql & "SELECT 顧客番号, 会社名, 会社名_カナ, 氏名,  
                    氏名_カナ, 部署名, 役職名, 役職名,  
                    郵便番号, 住所, 電話番号, FAX 番号, eMail  
                    FROM CUSTOMER "  
  
    'レコードセットオープン (SQL 実行)  
    Rec.Open sSql, Cnn, adOpenKeyset, adLockOptimistic  
  
    'データがある時  
    If Not Rec.EOF Then  
        With Rec  
            .MoveLast  
            .MoveFirst  
            '画面表示  
            Text1.Text = !顧客番号  
            Text2.Text = !会社名  
            Text3.Text = !会社名_カナ  
            Text4.Text = !氏名  
            Text5.Text = !氏名_カナ  
            Text6.Text = !部署名  
        End With  
    End If  
End Sub
```

```
Text7.Text = !役職名
Text8.Text = !郵便番号
Text9.Text = !住所
Text10.Text = !電話番号
Text11.Text = !FAX番号
Text12.Text = !eMail

End With
End If
'レコードセット開放
Rec.Close
Set Rec = Nothing

End Sub
```

VisualBasic でデータを取得し、画面に表示させる一連の流れは次の通りです。まず SQL を文字列として発行します。MicroSoft のオブジェクトモデル、レコードセットを使用して SQL を実行し、テキストボックスと言われるフィールドに値をセットしていきます。最後にレコードセットをメモリーから開放します。またデータに NULL が入っている場合など、IIF 関数等を使用しなければなりません。VisualBasic 開発者には、あまりにも見慣れたコーディングです。

TeamDeveloper は第4世代言語です。コーディングの中で接続モデルを使用することはありません。下記の通りデータを取得します。

!TeamDeveloper

```
PushButton: pb1
  Message Actions
    On SAM_Click
      If SqlConnect( hSql )
        ! SQL 作成
        Set sSql = "
        Set sSql = sSql || 'SELECT 顧客番号, 会社名,
              '会社名_加, 氏名, 氏名_加, 部署名,
              役職名, '郵便番号,
              住所, '電話番号, FAX番号, eMail
              INTO :df1, :df2,:df3, :df4,:df5,:df6,
              :df7,:df8,:df9,:df10,:df11,:df12
              FROM CUSTOMER'
        ! SQL 実行
        Call SqlPrepareAndExecute( hSql, sSql )
        ! データフェッチ
        Call SqlFetchNext( hSql, nNum )
```

TeamDeveloper は SQL を発行して、関数にあずけます。INTO 句をフィールド名にすることで、SQL 実行時に画面に表示させることができます。メモリーの解放は必要ありません、SQL 関数が処理してくれます。VisualBasic と流れとしてはあまり変わりありませんが、ステップ数が格段に違います。VisualBasic の約 1/3 です。関数で実行できるということは、プログラムコードの品質を一定に保つことができ、より高級な言語だと言えます。

このように、TeamDeveloper ではデータ取得が汎用的にできます。この処理をさらに部品化して、上記のコーディングすらなくすことができます。

2. SQL 自動生成のコーディング

これから今回のメインである、SQL を自動生成させることができる部品（関数）を紹介致します。SQL 文自体ノンコーディングな為、全ての画面で使用でき、画面のフィールドが追加されようと、削除されようと、SQL 文を意識することはありません。これほど開発者のニーズにあった関数は他にないでしょう。

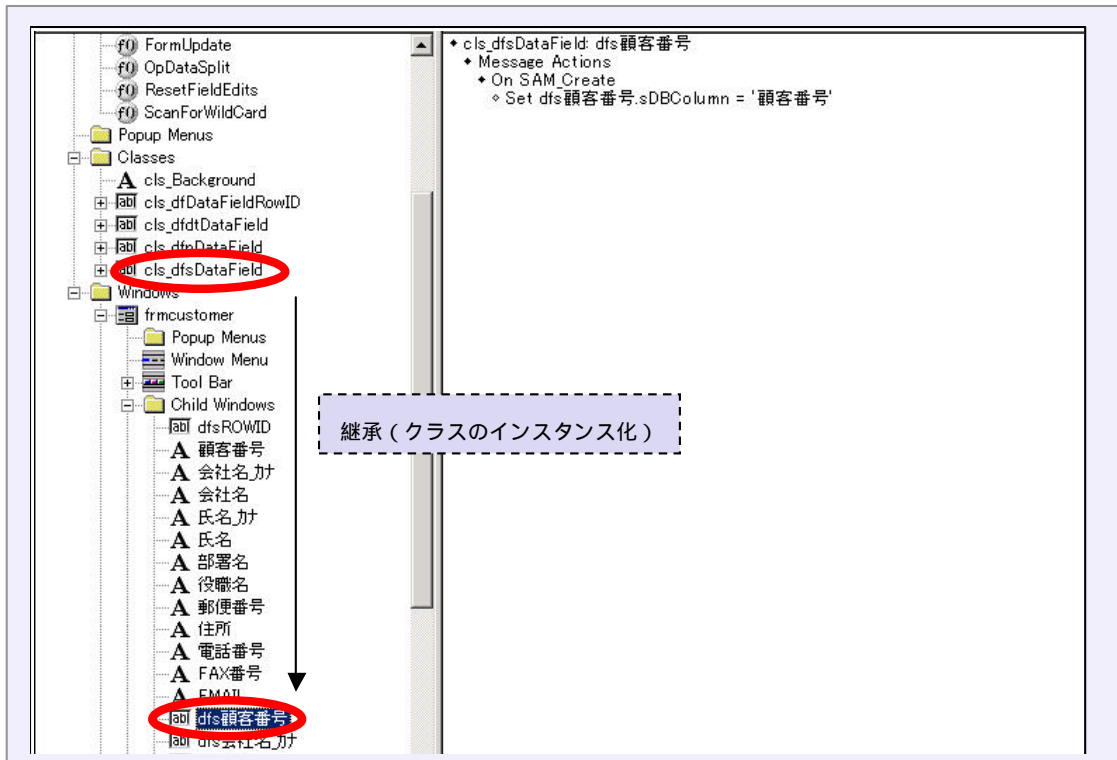
2.1 クラスのコーディング

まず汎用的な処理を記述したクラスを作成します。

```
Data Field Class: cls_dfsDataField
Description: データベースアクセスをする為のデータフィールド
              (文字列データを読み込むフィールド)
Derived From
Class Variables
Instance Variables
String: sDBCColumn !データベースの項目名
Functions
Message Actions
On PAM_SQL 2.3 ユーザー定義メッセージ
            Call BuildSQL ( wParam, hWndItem, sDBCColumn )
```

このクラスで、SQL 文を組み立てる関数（BuildSQL）を Call します。これを継承（2.2 データフィールド、クラスの継承（クラスのインスタンス化））するデータフィールドは全て、この関数を処理することになります（このロジックを通ります）。もちろん他クラスを作成して、処理を切り分けることもできます。

2.2 データフィールド、クラスの継承（クラスのインスタンス化）



TeamDeveloper アプリケーション内のツリー図

継承したフィールド（インスタンス化したフィールド）にデータベースの項目名をセットします。



間違えないデータベース項目名でなければなりません。この名前が、今後生成される SQL の SELECT 句になります。

2.3 ユーザー定義メッセージ

ユーザー定義のメッセージ（イベント）を作成します。

User ! P A Mメッセージ Number: $PAM_SQL = SAM_User + 1$	各画面に1度定義
--	----------

メッセージを定義することで、SalSenMsg 関数などで、イベントとして駆動させることができます。

2.4 ボタン押下時

ボタンクリック時に Function を Call します。

Pushbutton: pbSelect Message Actions On SAM_Click Call FormSelect(hWndForm,hSqlMain)

FormSelect は、上記で作成したクラス（2.1クラスのコーディング）のメッセージ（ユーザー定義メッセージ）をイベントとして駆動させるようコーディングされています。

2.5 FormSelect (Function)

Function : SalSendMsgToChildren でユーザー定義関数 PAM_SQL を呼び出すことで、各クラスの PAM_SQL が動作し SQL が発行され、実行します。

Function: **FormSelect**

Description: 検索をして検索結果を画面に表示する。

Returns

Boolean: !TRUE:OK FALSE:NG OR NODATA

Parameters

Window Handle: **hWndTopLevel** !検索結果を読み込ウィンドウハンドル

Sql Handle: **hSqlSelect** !SQL ハンドル

Static Variables

Local variables

Actions

! 変数を初期化します。

Set sColumns=""

Set sIntoVars=""

Set sWhere=""

! SELECT 文を作成します。

Call SalSendMsgToChildren(hWndTopLevel, PAM_SQL, nSELECT, 0)

! SQL 文を実行します。

If SqlPrepareAndExecute(hSqlSelect, 'SELECT ' || sColumns ||
' INTO ' || sIntoVars || ' FROM ' || sTable)

Call SqlGetResultSetCount(hSqlSelect, nWork1)

If nWork1=0

Return FALSE

Else

Call SqlFetchNext(hSqlSelect, nWork1)

Return TRUE

Else

Return FALSE

Cls_dfsDataField を継承している (インスタンス化された) 各フィールドにメッセージが送信され、BuildSQL がフィールドの数ぶん呼び出される

グローバル変数
sColumns: DB 項目
sIntoVars: 画面のフィールド名
SQL 文を実行します。

SQL を自動生成させ、実行します。

2.6 BuildSQL (Function)

Function : 各フィールドに設定された、データベース項目の文字列を結合します。

Function: **BuildSQL**

Description: 動的な SQL 文の部品を生成

Returns

Parameters

Number: **nStatementType** !SQL コマンドを作成する時のステータス

Window Handle: **hWnd** !ウィンドウハンドル

String: **sDbColumn** !データベースの項目名

Static **Variables**

Local variables

String: **sParentName**

String: **sChildName**

Actions

! 完全修飾のバインド変数を生成するために、親及びフィールド名を取得

親フォーム名取得

Call GetItemName(SalParentWindow(hWnd), sParentName)

フィールド名取得

Call SalGetItemName(hWnd, sChildName)

! SELECT COMMAND

If sColumns != ""

Set sColumns = sColumns || ', '

DB 項目の文字結合

Set sIntoVars = sIntoVars || ', '

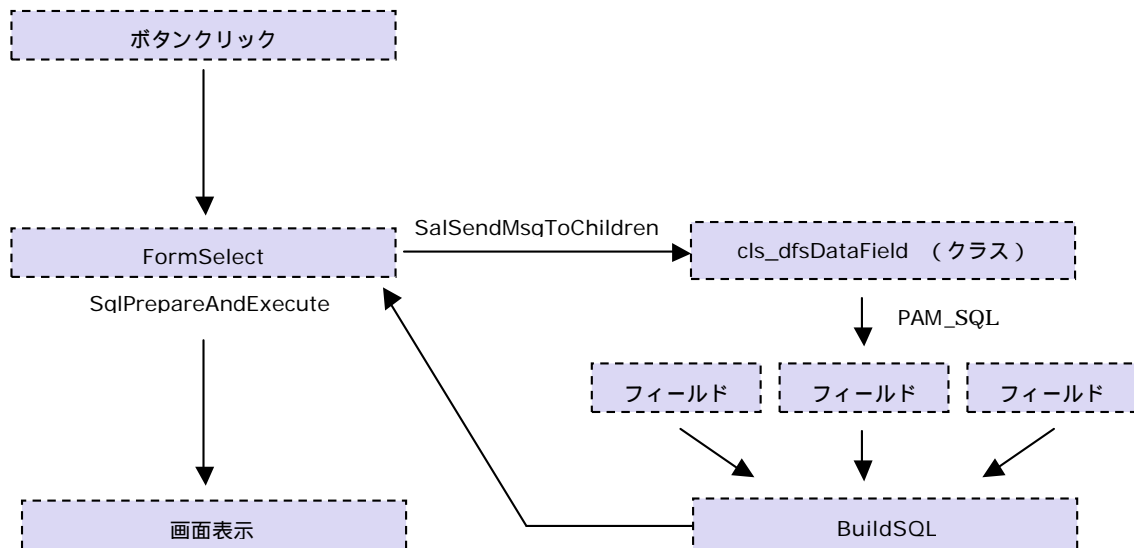
フィールド名の文字結合

Set sColumns = sColumns || sDbColumn

**Set sIntoVars = sIntoVars || ':' || sParentName || '.' ||
sChildName**

SELECT するデータベース項目、INTO 句のデータフィールド項目名を格納します。

2.7 SQL 自動生成時の処理フロー



FormSelect と BuildSQL は FUNCTION です。これは全画面で汎用的に使用することができ、もちろんクラスとして作成することもできます。クラスと FUNCTION を汎用的に使用することで、コーディングを画期的に少なくすることができます。各フィールドにデータベース項目を設定しておくだけで、SQL 文が発行され、実行されます。SELECT したいデータベース項目を各フィールドに設定しておくだけなのです。上記の処理フローを完全に把握すれば、SQL 作成だけにとどまらず、各種処理に応用することは容易です。

次回

今回のコーディングを CDK (TeamDeveloper に Include して使うライブラリ) 化して、より生産性を向上させることができる方法をご説明します。