尾崎 浩司

株式会社ミガロ.

システム事業部 プロジェクト推進室

DataSnapを使用した3層アプリケーション構築技法

多層型アプリケーションのためのフレームワーク「DataSnap」。
これを活用し、Delphi/400 のスキルだけで構築を実現する。

略歴
1973 年 08 月 16 日生
1996 年三重大学工学部卒
1999 年 10 月株式会社ミガロ . 入社
1999 年 10 月システム事業部配属

現在の仕事内容
ミガロ . 入社以来、主に Delphi/400 を利用した受託開発を担当している。

1.3層アプリケーションとは

一般的にクライアントサーバ型アプリケーションというと、画面処理＋ビジネスロジックを担当する「クライアント(アプリケーション層)」とデータを保持する「データサーバ（データ層)」の 2 階層で構築することが多いだろう。

対して、3 層アプリケーションとは、画面処理を担当する「クライアント（プレゼンテーション層)」、ビジネスロジックを担当する「アプリケーションサーバ(アプリケーション層)」およびデータを保持する「データサーバ（データ層)」の 3 階層に分割して構築するクライアントサーバ型アプリケーションのことを指す。【図 1】

3 階層の特徴は、ビジネスロジック部分をアプリケーションサーバ上のプログラムとして、クライアント PC 上で実行される画面処理部分のプログラムと完全に分離することにある。

3 層アプリケーションとしてシステムを構築するメリットとしては、次のようなことが挙げられる。

(1) ビジネスロジックの変更が行いやすい
(2) 大量データを取り扱うアプリケーションが構築しやすい
(3) クライアント PC の環境構築がシンプルになる

(1) 2 層アプリケーションのケースでは、ビジネスロジックがクライアント PC 上にあるため、仕様変更が必要となった場合、都度クライアント PC へ最新モジュールの再配布が必要となる。

対して、3 層アプリケーションとしたケースでは、ビジネスロジックがアプリケーションサーバ上にあるプログラムとなるため、画面処理部分の変更がない限り最新モジュールはアプリケーションサーバにのみ適用すればよい。

(2) 3 層アプリケーションとしたケースでは、ビジネスロジックを記述したプログラムとデータベースとのやり取りがサーバ間でのネットワーク通信となり、サーバ⇔クライアント間のやり取りは、画面処理に必要な情報のみに絞り込むことができる。

これは、ビジネスロジックが大量のデータを処理する必要がある場合に、遠隔拠点に配置したクライアント PC から実行指示を行う際の、処理レスポンス向上に大きく寄与するだろう。

(3) 通常 2 層アプリケーションのケースでは、クライアント PC 上にデータベース通信用の専用モジュールをセットアップしなければいけないが、3 層アプリケーションとしたケースでは、クライアント PC は直接データベースを参照しないため、クライアント PC にデータベース通信用モジュールが不要となる。

よって、Delphi/400 の特徴であるランタイム不要な実行モジュールが作成できる点を最大限に活用することが可能になる。つまり、クライアント PC には、実行モジュール（Exe）だけ配布すれば

図1

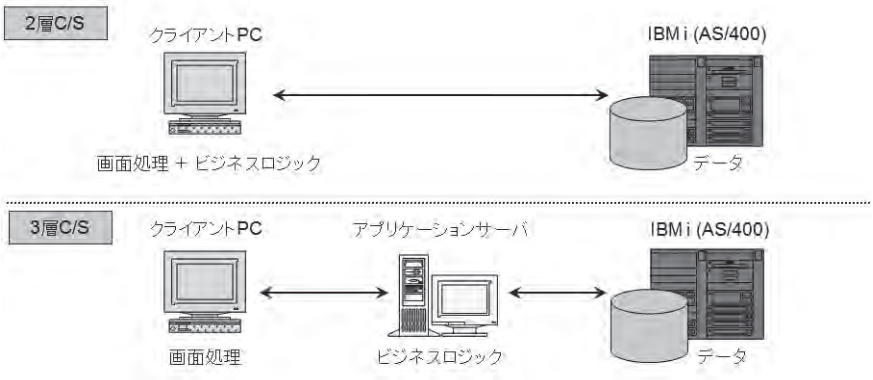
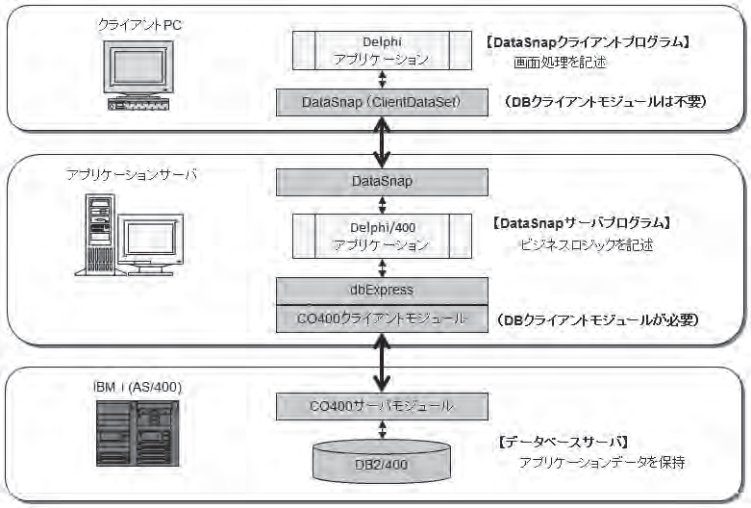


図2



ソース1

```
SEU==>                                                          DTMSCP
FMT A* ....A*. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
       ********** データの始め **********
0001.00      A*=============================================================*
0002.00      A*       ファイル名       : DTMSCP                             *
0003.00      A*       ファイル記述     : 得意先マスタ                       *
0004.00      A*=============================================================*
0005.00      A                                    UNIQUE
0006.00      A          R MCR00                   TEXT('得意先マスタ')
0007.00      A*
0008.00      A            MCTRCD         8S 0     COLHDG('得意先コード')
0009.00      A            MCTRNM         52O      COLHDG('得意先名称')
0010.00      A            MCTRKN         20A      COLHDG('カナ名')
0011.00      A            MCYUBN          8A      COLHDG('郵便番号')
0012.00      A            MCADR1         42O      COLHDG('住所１')
0013.00      A            MCADR2         42O      COLHDG('住所２')
0014.00      A            MCTEL          14A      COLHDG('電話番号')
0015.00      A            MCFAX          14A      COLHDG('ＦＡＸ番号')
0016.00      A          K MCTRCD
```

よいのである。これは、OS/400のバージョンアップやDelphi/400運用版ミドルウェアのバージョンアップをより容易にするだろう。

2.DataSnapとは

こういったメリットを持つ3層アプリケーションだが、あまり利用されていないのはなぜだろうか。それは、アプリケーションサーバとクライアントPCとの間の通信を実装する手順が容易ではないからである。

技術的には「CORBA」と呼ばれる分散オブジェクト技術の仕様があるのだが、これをアプリケーションに導入しようとすると、どうしてもCORBAに関する専門知識が不可欠となる。つまり、実現は可能だが敷居が高いというのが本音だろう。

ところが、Delphi/400には、「DataSnap」と呼ばれる多層型アプリケーションを構築するためのフレームワークが搭載されている。このDataSnapを利用すれば、Delphi/400のスキルだけで3層アプリケーションが構築できるのである。

DataSnapを使用して構築する3層アプリケーションは、図2のような構成となる【図2】。一見すると複雑そうなイメージではあるが、これらは、通常のアプリケーション同様、コンポーネントを使用したビジュアル開発で作成可能である。

特に、Delphi/400 VersionXEでは、便利なウィザードが使用できるため、より容易に作成できるようになっている。今回は、このDataSnapを使用した3層アプリケーションの構築技法を紹介しようと思う。

なお、今回作成するサンプルプログラムは、ソース1のDDSより生成された「得意先マスタ（ファイル名：DTMSCP）」を使用する。【ソース1】

3.DataSnapサーバプログラム作成手順

アプリケーションサーバに配置する、DataSnapサーバプログラムの作成手順を見ていこう。

まずDelphi/400を起動したら、［ファイル｜新規作成｜その他］より新規作成メニューを開き、「DataSnap Server」を選択する。【図3】

するとウィザードが始まるので、次のとおり指定していこう。

①第1画面の「プロジェクトの種類」では通常の「VCLフォームアプリケーション」を選択する。今回はサーバプログラムも通常のフォームアプリケーションとして作成する。【図4】

②第2画面目の「サーバの機能」画面では、使用するサーバ機能を選択する。ここでは初期値のまま「プロトコル-TCP/IP」「サーバメソッドクラス」にチェックを付けておこう。もし通信手段にHTTPを使用する場合は、ここで選択すればよい。【図5】

③第3画面目は、通信に使用するポートを選択する。ここも通常は初期値である「211」を指定しておけばよいだろう。もし同じサーバ上に複数のDataSnapサーバプログラムを配置するような場合には、それぞれのプログラムで異なるポート番号を指定すればよい。【図6】

④最後の第4画面目の「サーバメソッドクラスの上位クラス」では、メソッド等を定義する親クラスを指定する。通常ここではTDSServerModuleを選択しよう。これを選択すると、2層アプリケーション開発においても多用するデータモジュールと同様の開発手順が可能となる。【図7】

以上で、設定が完了である。

ウィザードが完了すると、次の3つのユニットを持つプロジェクトが生成される。

(1) ServerControllerUnit1.pas（DataSnap本体）
(2) ServerMethodsUnit1.pas（サーバ実装用モジュール）
(3) Unit1.pas（メインフォーム）

プロジェクトが生成されたら、ビジネスロジックにあたる処理を(2) のServerMethodsUnit1.pasにあるTServerMethods1に作成すればよい。ここでは、得意先マスタの内容を取得できるサーバを作成していこう。

ServerMethodsUnit1.pasのデザイナを立ち上げ、TSQLConnection、TSQLTable、TDataSetProviderを貼り付けて、データベース接続設定を行ってみよう。なお、この手順は、データモジュールにdbExpress接続のデータベースアプリケーションを作成するのと同じであることも分かるであろう。【図8】

ところで、設定が完了したら、一度tblDTMSCPのActiveプロパティをTrueに変更して、データベースに正しく接続できるか確認しておくとよい。無事に接続確認ができたら、いったんSQLConnection1のConnectedプロパティをFalseに戻して、接続を切断しておこう。

これだけで、DataSnapサーバプログラムの作成は完了である。

完了したらプロジェクトに名前を付けて保存してからコンパイルのうえ、完成したExeファイルを実行してみよう。すると、フォームが1つ立ち上がるアプリケーションが実行されるだろう。実は、このプログラムが、DataSnapを使用したアプリケーションサーバとなっているのである。

では、このサーバプログラムを立ち上げた状態のまま、次項からクライアントプログラムを作成していこう。

4.DataSnapクライアントプログラム作成手順

クライアントプログラムは、VCLフォームアプリケーションとして作成しよう。

新規作成後、生成されたForm1にTClientDataSet、TDataSource、TDBGridおよびTDBNavigatorを貼り付けて、各コンポーネントの紐付けを行う。この手順も、dbExpress接続アプリケーションで、お馴染みであろう。

● TSQLConnectionコンポーネント

次に、TSQLConnectionコンポーネントを貼り付け、ConnectionNameプロパティにDataSanpCONNECTIONを選択しよう。実はDataSnapクライアントプログラムは、dbExpress接続プログラム同様、TSQLConneciton で接続を行うのである。Driverプロパティにはサーバ情報を指定すればよい。【図9】

図3

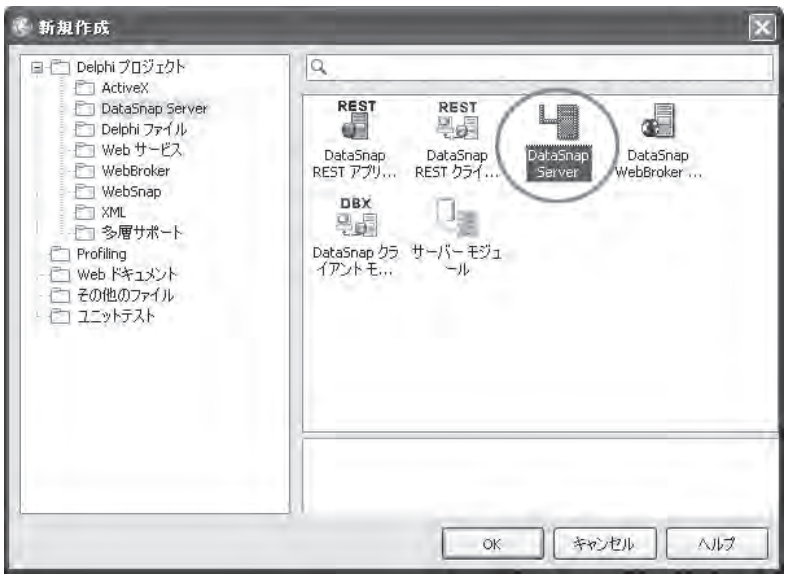

新規作成
Delphi プロジェクト
ActiveX
DataSnap Server
Delphi ファイル
Web サービス
WebBroker
WebSnap
XML
多層サポート
Profiling
Web ドキュメント
その他のファイル
ユニットテスト
REST
DataSnap REST アプリ...
REST
DataSnap REST クライ...
DataSnap Server
DataSnap WebBroker ...
DBX
DataSnap クライアント モ...
サーバー モジュール
OK
キャンセル
ヘルプ

図4


DataSnap Server
プロジェクトの種類
このアプリケーションの種類を指定します
VCL アプリケーションは VCL フォームを表示します
VCL フォーム アプリケーション(V)
コンソール アプリケーション(C)
サービス アプリケーション(S)
ページ 1/4
<< 戻る(B)
次へ(N) >>
完了
キャンセル
ヘルプ

図5


DataSnap Server
サーバーの機能
DataSnap サーバーに追加する機能のチェック ボックスをオンにします
詳細については、機能を選択してください
タイトル
プロトコル
TCP/IP
HTTP
認証
権限付与
サーバー メソッド クラス
サンプル メソッド
すべて選択/選択解除
ページ 2/4
<< 戻る(B)
次へ(N) >>
完了
キャンセル
ヘルプ

具体的には、今回は同じ端末上にDataSnapサーバが稼働しているため、HostNameプロパティはlocalhostのままでよい。（DataSnapサーバプログラムがリモートサーバの場合は、サーバのIPアドレスを指定することとなる。）また、DataSnapサーバプログラムで指定したポート番号「211」を、Portプロパティに設定しよう。

DataSnapサーバプログラムへの接続設定が完了した。

● TDSProviderConnectionコンポーネント

続いて、TDSProviderConnectionコンポーネントを貼り付けよう。これは、DataSnapサーバプログラムで作成したサーバメソッドクラスを指定するものだ。

ここでは、ServerClassNameプロパティにTServerMethods1と入力し、SQLConnectionプロパティにSQLConnection1を選択すればよい。これで、DataSnapサーバ上のサーバメソッドをクライアントから使用できるようになる。

最後に、ClientDataSet1を選び、RemoteServerプロパティにDSProviderConnection1を選択する。すると、ProviderNameプロパティに、DataSnapサーバプログラム上で定義したdspDTMSCPが選択できるはずである。ここまでのところを一通り設定したのが図10である。【図10】

以上で、設定が完了である。

設定が完了したら、ソース2のようなプログラムを記述しよう。画面起動時（FormのOnCreate時）にクライアントデータセットを開く処理と、データセットPost後（ClientDataSetのOnAfterPost時）にクライアントデータセットの変更内容をサーバに適用するロジックのみである。【ソース2】

完成したら、クライアントアプリケーションを実行してみよう。クライアントプログラムには、IBM i（AS/400）に接続する処理は一切記述されていないが、データが表示されているのが分かるであろう。【図11】

ここまで、DataSnapを使用することで、簡単に3層アプリケーションを構築できることをお伝えした。DataSnapの仕組みと設定ポイントがお分かりいただけたかと思う。

5.Windowsサービスを使用したDataSnapサーバの作成

先ほど作成したDataSnapサーバプログラムは、VCLフォームアプリケーションであった。つまり、クライアントプログラム実行前に、あらかじめDataSnapサーバプログラムの画面を起動しておく必要がある。しかし、実際のアプリケーションサーバでは、常にログインしたままプログラムを実行しておくというのは現実的ではない。

そこでここからは、DataSnapサーバプログラムを「Windowsサービス」として実行できるようにしてみよう。

また、先ほどは単純に得意先マスタの内容を表示するだけであったが、今度はクライアントアプリケーション側から絞り込み条件として「得意先カナ」を指定することで、対象データを検索できるように拡張してみたい。

● Windowsサービス

Windowsサービスプログラムとは、Windows動作中にバックグラウンドで常に実行させることが可能なプログラムのことである。これはログインしなくても実行させることができるため、DataSnapサーバプログラムに最適であろう。

Windowsサービスを使用したDataSnapサーバプログラムは、前述の「3. DataSnapサーバプログラム作成手順」の項と同様の手順でプロジェクトが作成できる。

ウィザードの第1画面目（図4）に出る「プロジェクトの種類」で「サービスアプリケーション」を指定すればよい。ただし、作成されたプロジェクトは、前述の「VCLフォームアプリケーション」と同様だが、Unit1.pas（Form1）だけがない点に注意されたい。今回の「サービスアプリケーション」には通常画面が存在しないからだ。

●データ検索処理

また今回は、クライアントからデータ検索処理が呼び出せるようサーバメソッドも追加しよう。

宣言部に、GetDataDTMSCPという名前の関数を追加する。【ソース3】

この関数の引数（AMCTRKN）に指定されたカナ名をもとに得意先マスタをカナ名で検索し、条件に合致するデータのみ抽出できるようにしたい。データの抽出には、SQL文を発行するためにTSQLQueryコンポーネントを使用しよう。ServerMethodsUnit1には図12のように設定を行う。【図12】

準備ができたら、ソース4のようなロジックを作成しよう。ここでは、サービス起動時にデータベースへ接続する処理と、引数に指定された「カナ」を条件にデータを抽出するSQL文を発行し、検索結果が1件以上存在した場合、戻り値にTrueをセットするメソッドを作成している。【ソース4】

なお、ここまで見てきた手順も、データモジュールに関数（メソッド）を定義する手順と同様であることがお分かりいただけるであろう。

以上、ソースが完成したらプロジェクトに名前を付けて保存のうえ、コンパイルを行おう。加えて、完成したプログラム（Exe）はそのまま実行するのではなく、サービスに登録する必要がある。これは通常コマンドプロンプトで、プログラム名の後ろに「/install」を付加して実行すればよい。その後サービス管理画面で、登録されたサービスを「開始」にする。【図13】

今回は実行しても、上述したようにプロジェクトにフォームがないため、画面が何も表示されないが、サービス実行状態が「開始」となっていれば完了である。

6.サーバメソッドを呼び出すクライアントプログラムの作成

今度は、サービス登録したDataSnapサーバを使用するクライアントプログラムを作成していこう。

まず、検索条件となる「カナ」入力を行うTEditと、検索を実行するTBitBtnを貼り付けて画面を作成しよう。なお、TSQLConnection、TDSProviderConnection、TClientDataSet

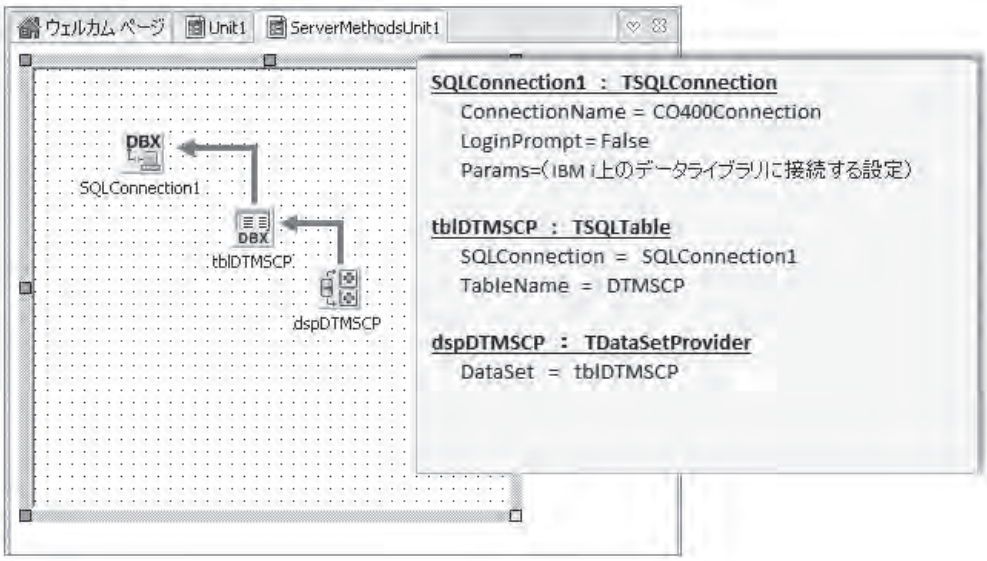
図6


DataSnap Server
ポート番号
DataSnap サーバーでクライアント要求のリスンに使用されるポートを指定します。[テスト] ボタンを使用すると、ポート番号がこのコンピュータでまだ使用されていないかどうかを確認できます。
TCP/IP 用のポートです。DataSnap のデフォルトの TCP/IP ポート番号は 211 です。
TCP/IP ポート(T):
211
ポートのテスト
オープン ポートの検索(E)
ページ 3/4
<< 戻る(B)
次へ(N) >>
完了
キャンセル
ヘルプ

図7


DataSnap Server
サーバー メソッド クラスの上位クラス
サーバー メソッド クラスの上位の型を選択します。
TDSServerModule を上位型とする場合は、設計画面のほかに、IAppServer インターフェイスの実装が提供されます。
TComponent(C)
TDataModule(D)
TDSServerModule(S)
ページ 4/4
<< 戻る(B)
次へ(N) >>
完了
キャンセル
ヘルプ

図8


ウェルカム ページ
Unit1
ServerMethodsUnit1
DBX
SQLConnection1
DBX
tblDTMSCP
dspDTMSCP
SQLConnection1 : TSQLConnection
ConnectionName = CO400Connection
LoginPrompt = False
Params=(IBM i上のデータライブラリに接続する設定)
tblDTMSCP : TSQLTable
SQLConnection = SQLConnection1
TableName = DTMSCP
dspDTMSCP : TDataSetProvider
DataSet = tblDTMSCP

の部分は、前述の「4. DataSnap クライアントプログラム作成手順」の項で設定した手順と同様でよい。【図 14】

● TSQLServerMethod コンポーネント

次に、TSQLServerMethod コンポーネントを貼り付けよう。これは、DataSnap サーバに定義したメソッドを呼び出すことができるクライアントコンポーネントである。具体的には、SQLConnection プロパティに SQLConnection1 を指定した後、ServerMethodName プロパティを選ぶとメソッドの一覧が表示される。ここで、先ほど作成した GetDataDTMSCP を選択すればよい。【図 15】

これで、クライアントプログラムからサーバメソッドが使用可能になるわけだ。プログラムの記述例は、ソース 5 のようになる。

メソッドの引数（今回の場合、AMCTRKN）および処理結果の戻り値（Result）は、TSQLQuery におけるパラメータクエリー同様、ParamByName メソッドでアクセスできることが分かる。また、戻り値（Result）は、パラメータ名の ReturnParameter にて取得可能である。なお、サーバメソッドの実行は ExecuteMethod メソッドを使用する。【ソース 5】

今回作成したクライアントプログラムを実行すると、図 16 のようになる。クライアント PC で指定した条件により、DataSnap サーバが検索処理を行い、結果のデータセットを返却していることが分かる。【図 16】

7.最後に

今回は DataSnap を使用した、3 層アプリケーションの構築技法を紹介した。この技法を用いれば、例えば GUI アプリケーションと Web アプリケーションとで同じビジネスロジックを使用するような場合にも有益そうである。

図 17 は、DataSnap サーバプログラムにアクセスするクライアントプログラムを、Web アプリケーション用フレームワークである VCL for the Web を使用して作成したものである。【図 17】

Web サーバは通常、社内 LAN 環境とは別に DMZ（非武装地帯）に配置することが多い。しかし、DataSnap を使用すれば、Web サーバとアプリケーションサーバとの間にデータベース接続用の特別なポートを開放することなくアプリケーションを公開できるため、安全性を向上させることが可能である。【図 18】

このように、DataSnap を使用した 3 層アプリケーションとしてシステムを構築しておくと、たとえ当初はクライアントプログラムを GUI アプリケーション用に作成したとしても、将来 Web アプリケーション化等を検討する際に容易に応用がきくし、安全性の向上というメリット獲得にもつながる。そういったことも勘案し、ぜひ一度 DataSnap を使用した 3 層アプリケーションにチャレンジしてみてほしい。

M

図9

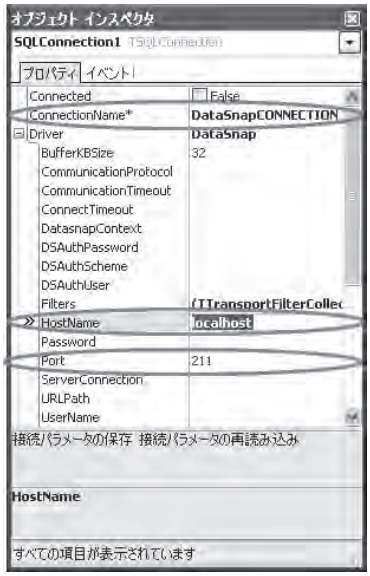
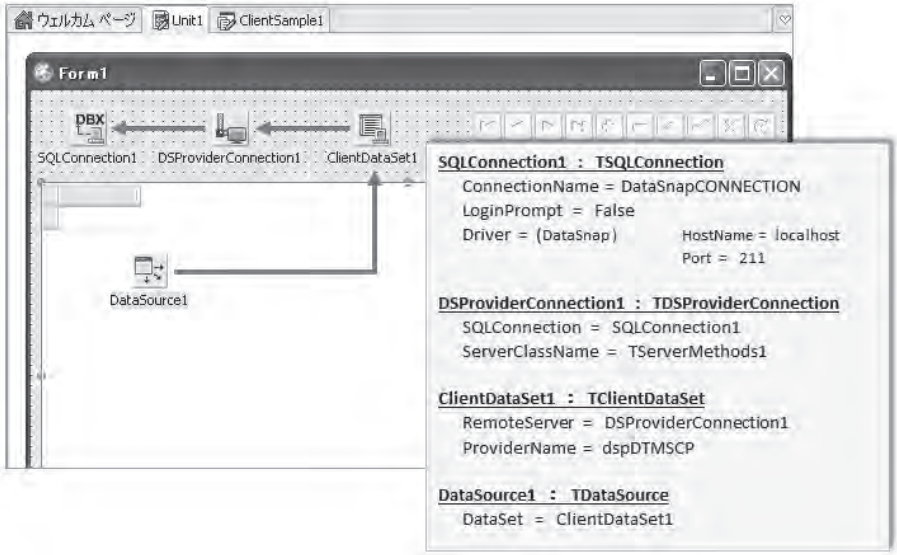


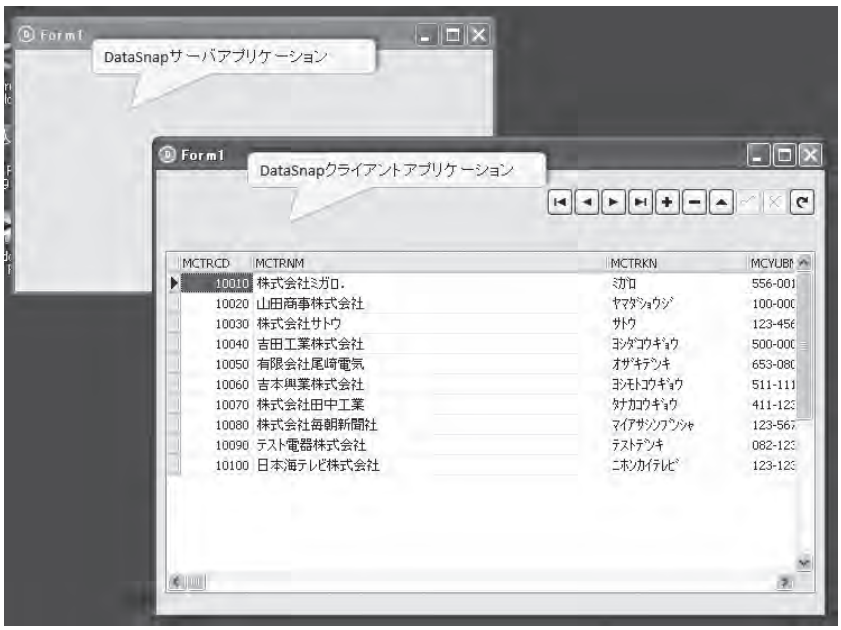
図10



ソース2

```
implementation

{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
begin
  // クライアントデータセットを開く
  ClientDataSet1.Active := True;
end;

procedure TForm1.ClientDataSet1AfterPost(DataSet: TDataSet);
begin
  //クライアントデータセット上でPost時にサーバーへ適用する
  ClientDataSet1.ApplyUpdates(0);
end;

end.
```

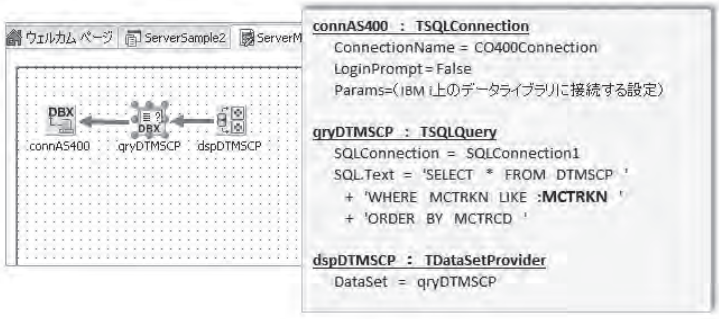
図11



ソース3

```
unit ServerMethodsUnit1;

interface

uses
  SysUtils, Classes, DSServer, DBXDynalink, FMTBcd, Provider, DB, SqlExp

type
  TServerMethods1 = class(TDSServerModule)
  private
    { private 宣言 }
  public
    { public 宣言 }
    function GetDataDTMSCP(AMCTRKN: String): Boolean; //追加メソッド
  end;
```

図12



ソース4

```
procedure TServerMethods1.DSServerModuleCreate(Sender: TObject);
begin
  //データベースへ接続する
  connAS400.Connected := True;
  //SQLQueryのパラメータに初期値をセット
  qryDTMSCP.ParamByName('MCTRKN').AsAnsiString := '';
end;

function TServerMethods1.GetDataDTMSCP(AMCTRKN: String): Boolean;
begin
  //初期化
  Result := False;

  //SQLQueryの実行
  with qryDTMSCP do
  begin
    Active := False;

    //パラメータの設定
    ParamByName('MCTRKN').AsAnsiString := '%' + AMCTRKN + '%'; //中間一致

    //データセットを開く
    Active := True;

    //対象データが存在する場合、結果にTrueを返す
    if not (Eof and Bof) then
      Result := True;
  end;
end;
```

図13

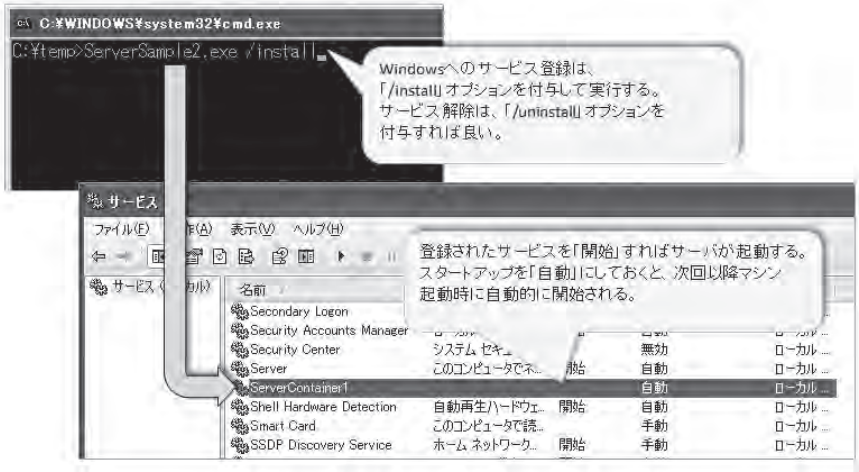


図14

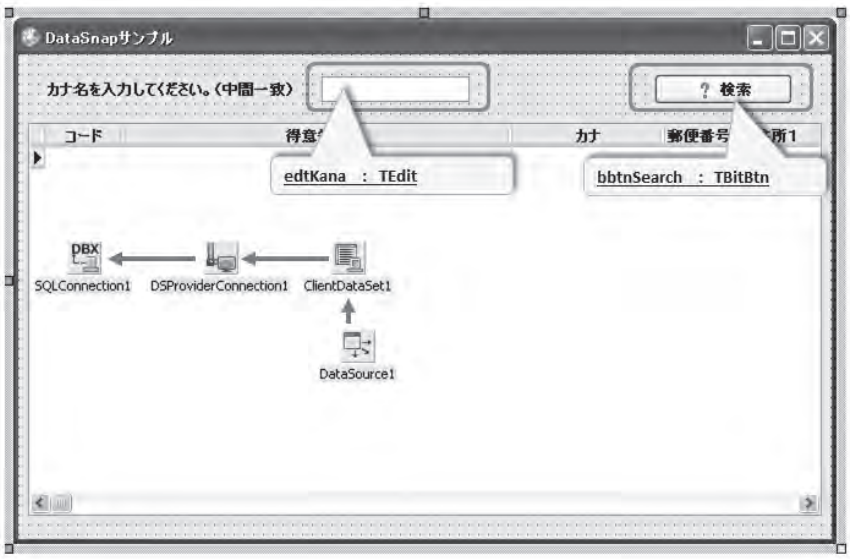
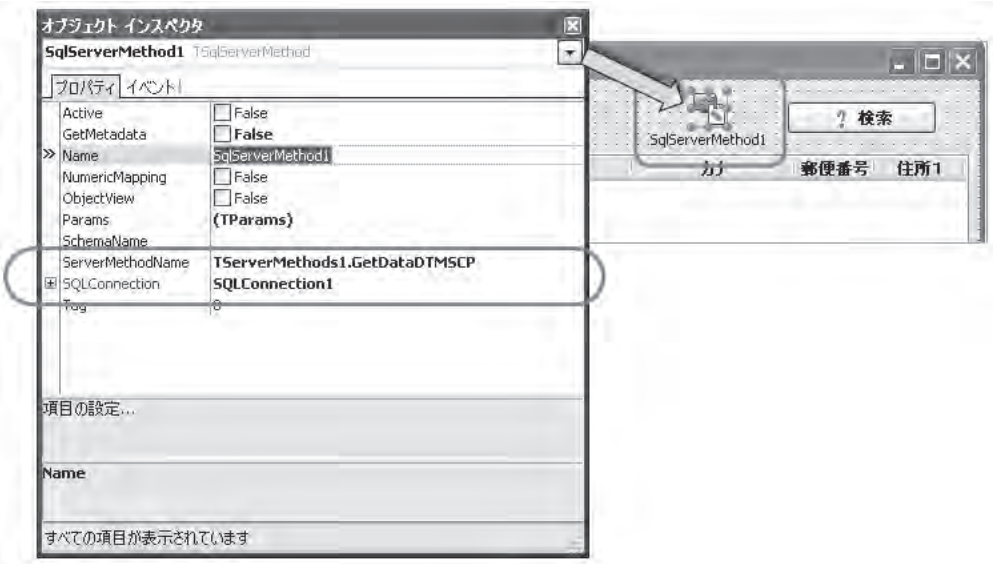


図15

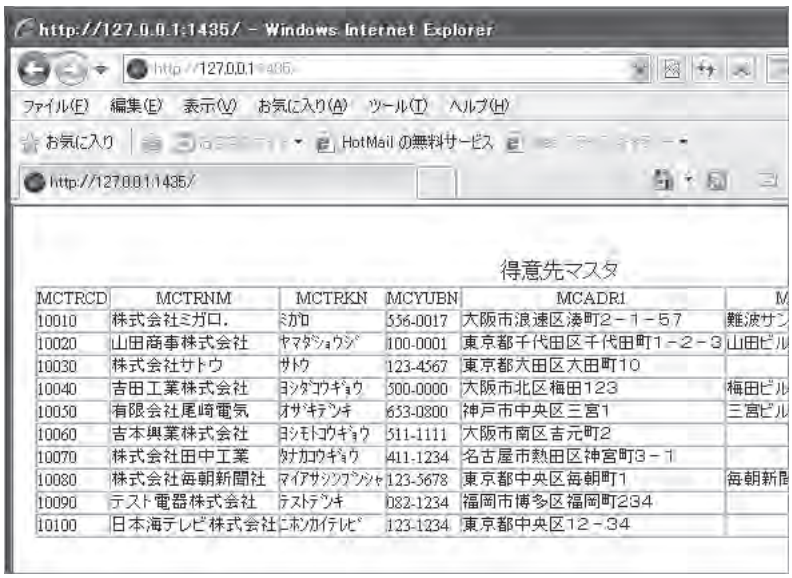


ソース5

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  //DataSnapサーバに接続する
  SQLConnection1.Connected := True;
end;

procedure TForm1.bbtnSearchClick(Sender: TObject);
begin
  //データセットを閉じる
  ClientDataSet1.Active := False;

  //画面のカナ項目値をパラメータにセット
  SqlServerMethod1.ParamByName('AMCTRKN').AsString := edtKANA.Text;

  //検索用メソッド(GetDataDTMSCP)を実行する
  SqlServerMethod1.ExecuteMethod;

  //実行結果がFalseの場合、エラーメッセージを出力する
  if not SqlServerMethod1.ParamByName('ReturnParameter').AsBoolean then
  begin
    MessageDlg('対象データが存在しません。', mtError, [mbOK], 0);
    Exit;
  end;

  //データセットを開く
  ClientDataSet1.Active := True;
```

図16

図17



得意先マスタ

MCTRCD	MCTRNM	MCTRKN	MCYUBN	MCADR1	M
10010	株式会社ミガロ.	ミガロ	556-0017	大阪市浪速区湊町2－1－57	難波サン
10020	山田商事株式会社	ヤマダショウジ	100-0001	東京都千代田区千代田町1－2－3	山田ビル
10030	株式会社サトウ	サトウ	123-4567	東京都大田区大田町10	
10040	吉田工業株式会社	ヨシダコウギョウ	500-0000	大阪市北区梅田123	梅田ビル
10050	有限会社尾崎電気	オザキデンキ	653-0800	神戸市中央区三宮1	三宮ビル
10060	吉本興業株式会社	ヨシモトコウギョウ	511-1111	大阪市南区吉元町2	
10070	株式会社田中工業	タナカコウギョウ	411-1234	名古屋市熱田区神宮町3－1	
10080	株式会社毎朝新聞社	マイアサシンブンシャ	123-3678	東京都中央区毎朝町1	毎朝新聞
10090	テスト電器株式会社	テストデンキ	082-1234	福岡市博多区福岡町234	
10100	日本海テレビ株式会社	ニホンカイテレビ	123-1234	東京都中央区12－34	

図18