【セッションNo. 3】

知って得する！現役ヘルプデスクが答える Delphiテクニカルエッセンス4.0

株式会社ミガロ.
RAD事業部 技術支援課 課長
吉原 泰介

【アジェンダ】

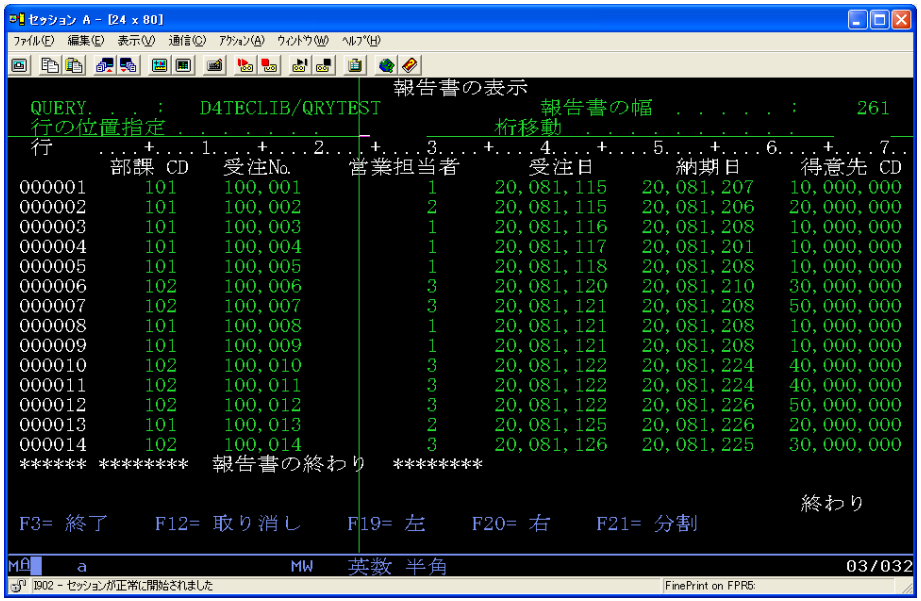
【1】RUNQRYをDelphi/400から利用しよう！
【2】ファンクションキー/Tabキー入力を制御しよう！
【3】ClientDataSetでグループ制御しよう！
【4】CSVファイルをTableで読み込もう！
【5】ExcelファイルをADOTableで読み込もう！

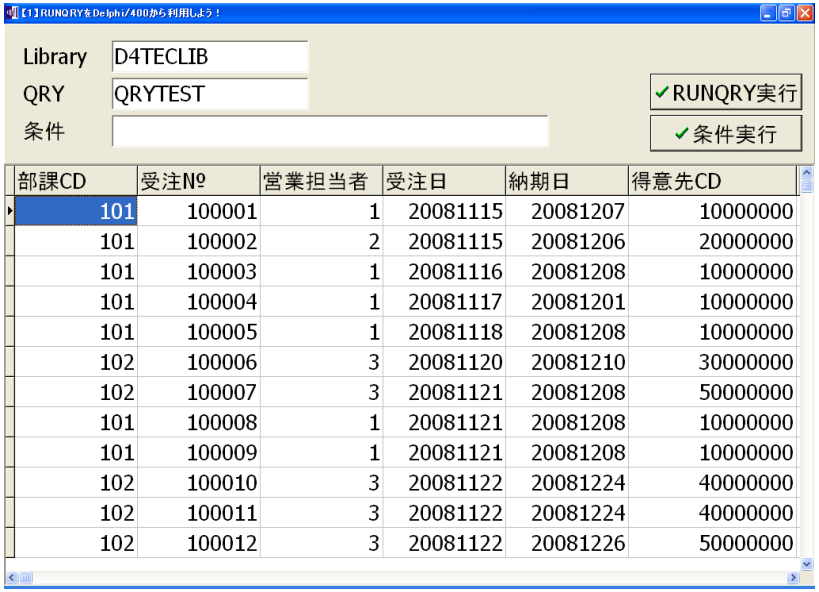
お持ち帰りトピック：開発で役立つショートカット機能

【1】RUNQRYをDelphi/400から利用しよう！

【1】RUNQRYをDelphi/400から利用しよう！

- IBM i（AS/400）上にあるQueryを活用するには





【1】RUNQRYをDelphi/400から利用しよう！

- 問題点

Queryは対話型で実行/結果表示されるが
Delphi/400は対話型ジョブではない。

**コマンド発行は可能なので実行して
結果をファイルで出力で受け取る。**

【1】RUNQRYをDelphi/400から利用しよう！

- 実行コマンド：RUNQRY

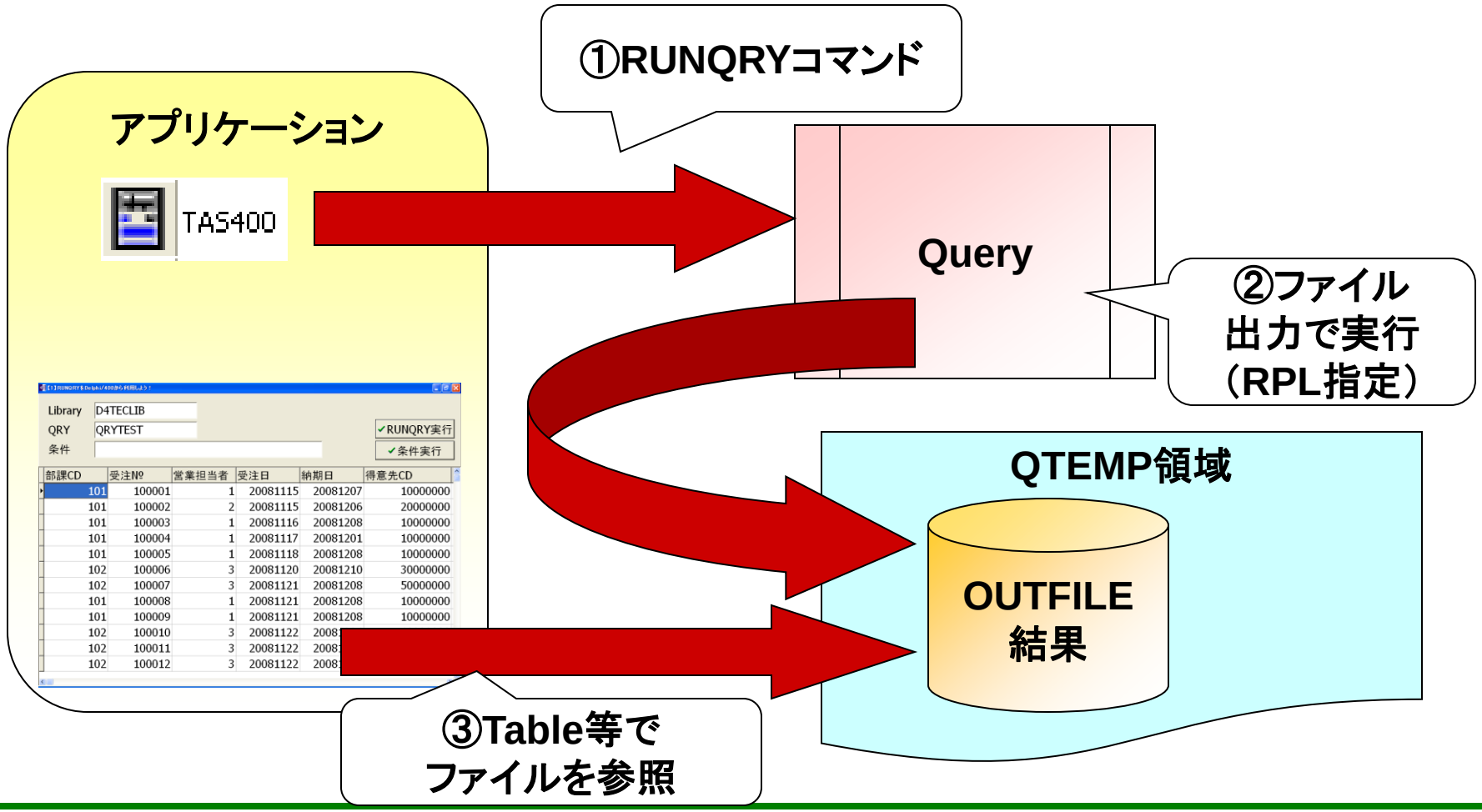
Queryの実行

```
RUNQRY QRY(ライブラリ名/Query名)
 OUTTYPE(*OUTFILE)
 OUTFILE(QTEMP/出力ファイル名 *FIRST *RPLFILE)
```

結果をファイル出力

出力先ファイルの指定
※上書きできるように*RPLFILEを指定

【1】RUNQRYをDelphi/400から利用しよう！



【1】RUNQRYをDelphi/400から利用しよう！

RUNQRYの実行

```
procedure TfrmT1.Button1Click(Sender: TObject);
var
  LibraryName, QryName, RUNQRY : String; //Library名, QRY名, RUNQRY実行文
begin
  tblQRY.Close;                    // 使用ファイルをClose
  LibraryName  := EdtLIB.Text;     // Library名
  QryName      := EdtQRY.Text;     // QRY名
  //RUNQRY実行文の編集
  //RUNQRYを実行してQTEMPにQRY名の結果ファイルを作成
  RUNQRY :=  ('RUNQRY QRY(' + LibraryName + '/' + QryName + ') ' +
             'OUTTYPE(*OUTFILE) ' +  'OUTFILE(QTEMP/' + QryName + ' *FIRST *RPLFILE)');
  //RUNQRYの実行
  DMmain.As400.RemoteCmd(RUNQRY);

  //RUNQRYの実行結果ファイルをTableで取得
  tblQRY.TableName := 'QTEMP/' +  QryName;
  tblQRY.Open;
end;
```

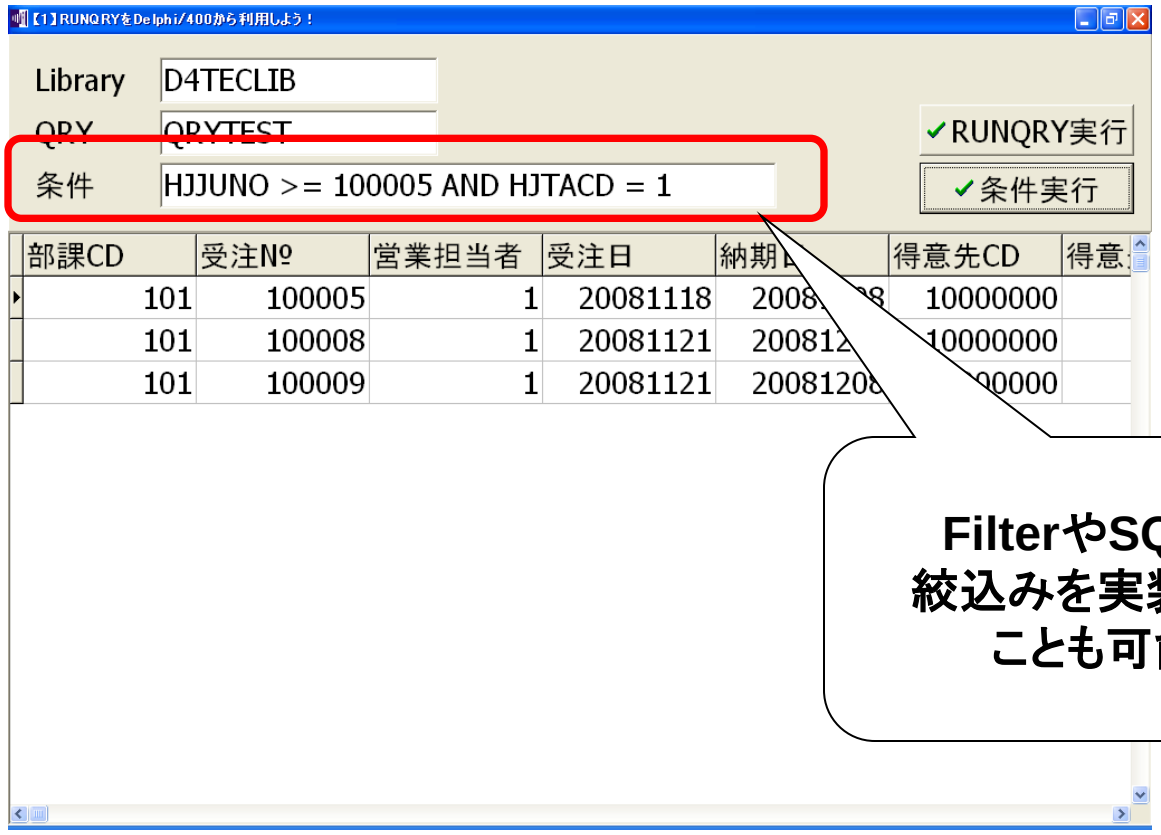
RUNQRY QRY(ライブラリ名/Query名)
OUTTYPE(*OUTFILE)
OUTFILE(QTEMP/出力ファイル名 *FIRST *RPLFILE)

【1】RUNQRYをDelphi/400から利用しよう！

- レコード選択が必要な場合

対話で処理できないので、レコード選択の指定を
行うことはできません。
条件が必要な場合は、結果ファイルを読み込む際に
次の方法で可能です。
① Filterプロパティで絞込む。
② QueryコンポーネントなどSQLの条件付きで取得する。

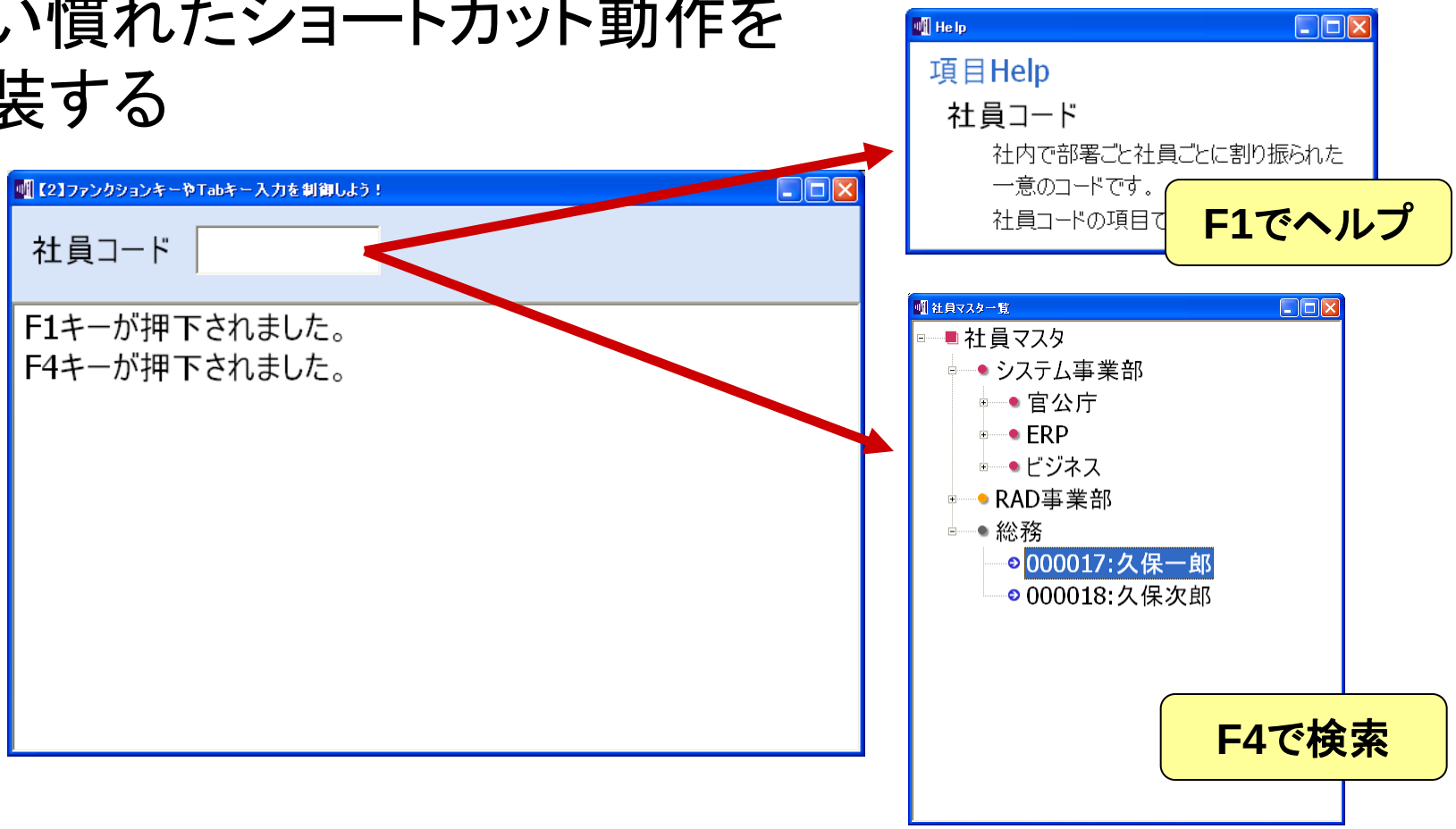
【1】RUNQRYをDelphi/400から利用しよう！



【2】ファンクションキーやTabキー入力を制御しよう！

【2】ファンクションキー入力を制御しよう！

- 使い慣れたショートカット動作を実装する



【2】ファンクションキー入力を制御しよう！

- ファンクションキーを画面全体で受け取るには

画面全体でファンクションキー入力を受け取るにはFormのOnKeyDownイベントが利用できる。

× OnKeyPressイベントは文字キー入力で発生するのでファンクションキーは取得でません。

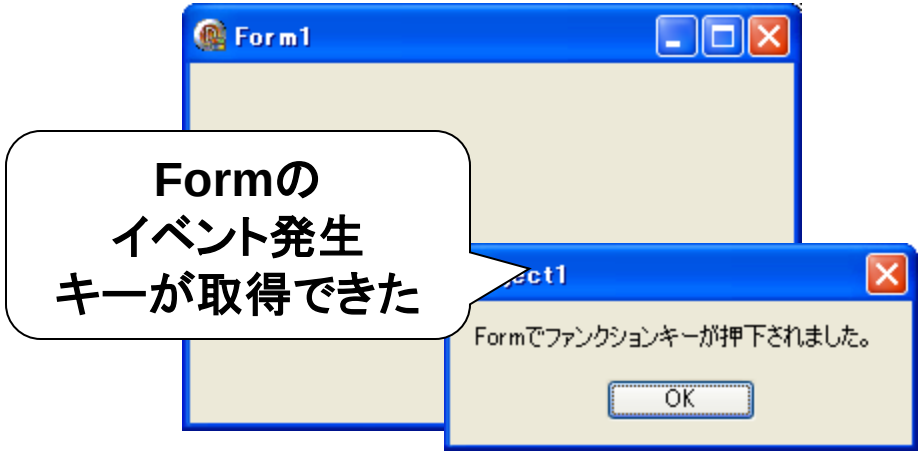
```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  ShowMessage('Formでファンクションキーが押下されました。');
end;
```

【2】ファンクションキー入力を制御しよう！

- 問題点

画面に入力コンポーネントがある場合、
FormのOnKeyDownイベントが発生しない。

例えば

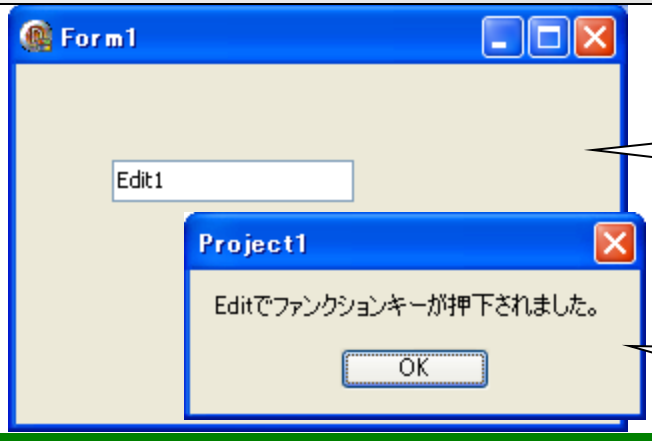


Editコンポーネントを置くと

【2】ファンクションキー入力を制御しよう！

- これは次のように、フォーカスを持つEdit側でOnKeyDwonイベントを取得しているため。

```
procedure TForm1.Edit1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  ShowMessage('Editでファンクションキーが押下されました。');
end;
```



Form:OnKeyDown
イベント発生なし
キーが取得できない

Edit:OnKeyDown
イベント発生
キーが取得できた

【2】ファンクションキー入力を制御しよう！

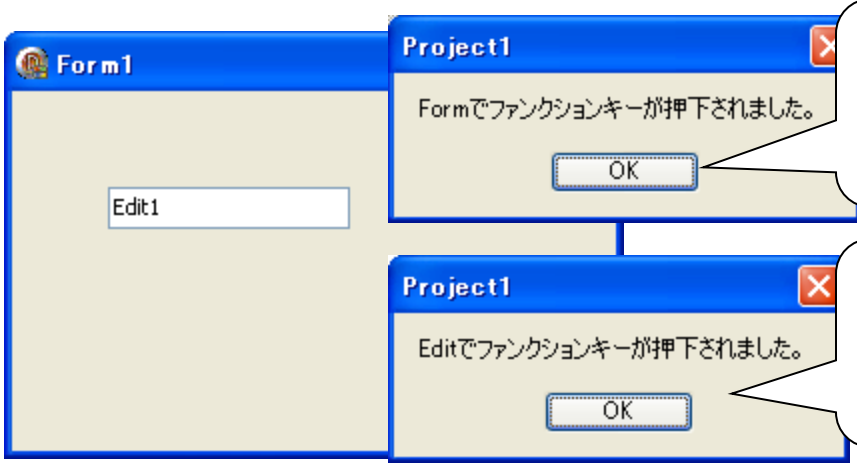
- ただし画面のコンポーネントごとに入力キーの制御を組み込むと
  - ○ 画面のコンポーネントごとに制御したい場合有効
  - × 画面全体で同じ動作の場合プログラムが煩雑になる

ファンクションキー取得のイベントを一箇所に集約するためには次の手法があります。
①FormのKeyPreviewプロパティをTrueに設定する。
②ActionListコンポーネントを利用する。

【2】ファンクションキー入力を制御しよう！

- 手法①　FormのKeyPreviewプロパティ

KeyPreviewプロパティがTrueの場合、キーボードイベントは
アクティブコントロールで発生する前にフォームで発生する。
両方で発生するので順番に注意（Formが先に発生）



Form:OnKeyDown
①イベント発生
キーが取得できた

Edit:OnKeyDown
②イベント発生
キーが取得できた

【2】ファンクションキー入力を制御しよう！

- 入力されたファンクションキーを判断する

仮想キー(VK_XX)を使用して、押されたキーを判断すると便利。

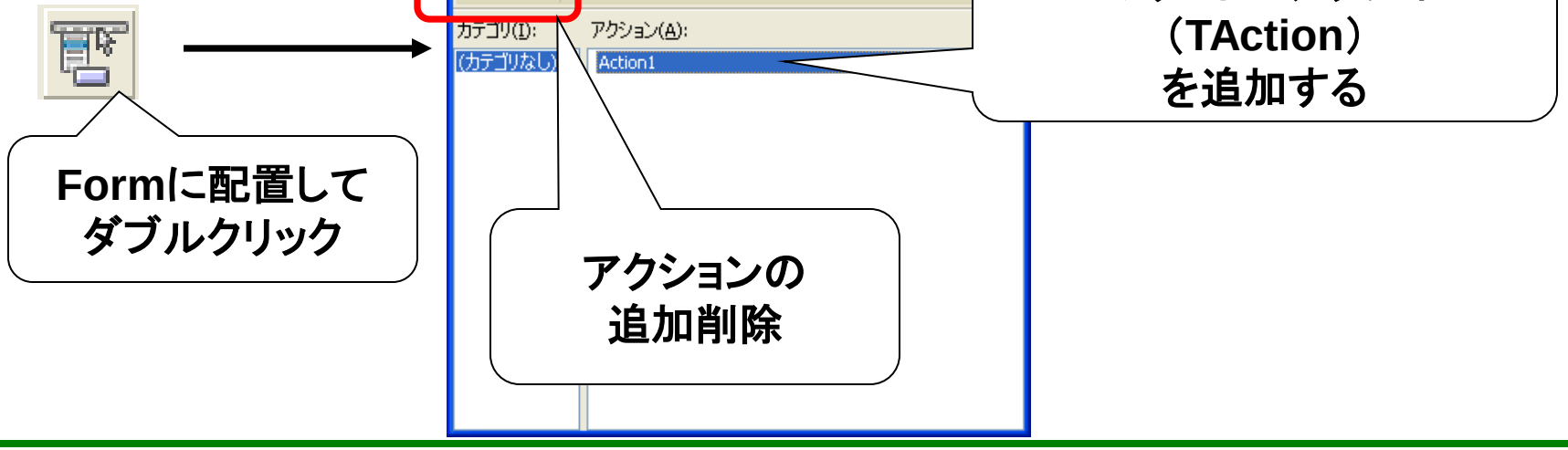
```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  //仮想キー(VK_XX)で判断
 case Key of
  VK_F1:begin
          //F1で実行したい処理を記述
          ShowMessage('F1キーが押下されました。');
        end;
  VK_F2:begin
          //F2で実行したい処理を記述
          ShowMessage('F2キーが押下されました。');
        end;
  end;  //同様に使用したいキー判断を追記する。
end;
```

【2】ファンクションキー入力を制御しよう！

- 手法②　ActionListコンポーネントを利用する。

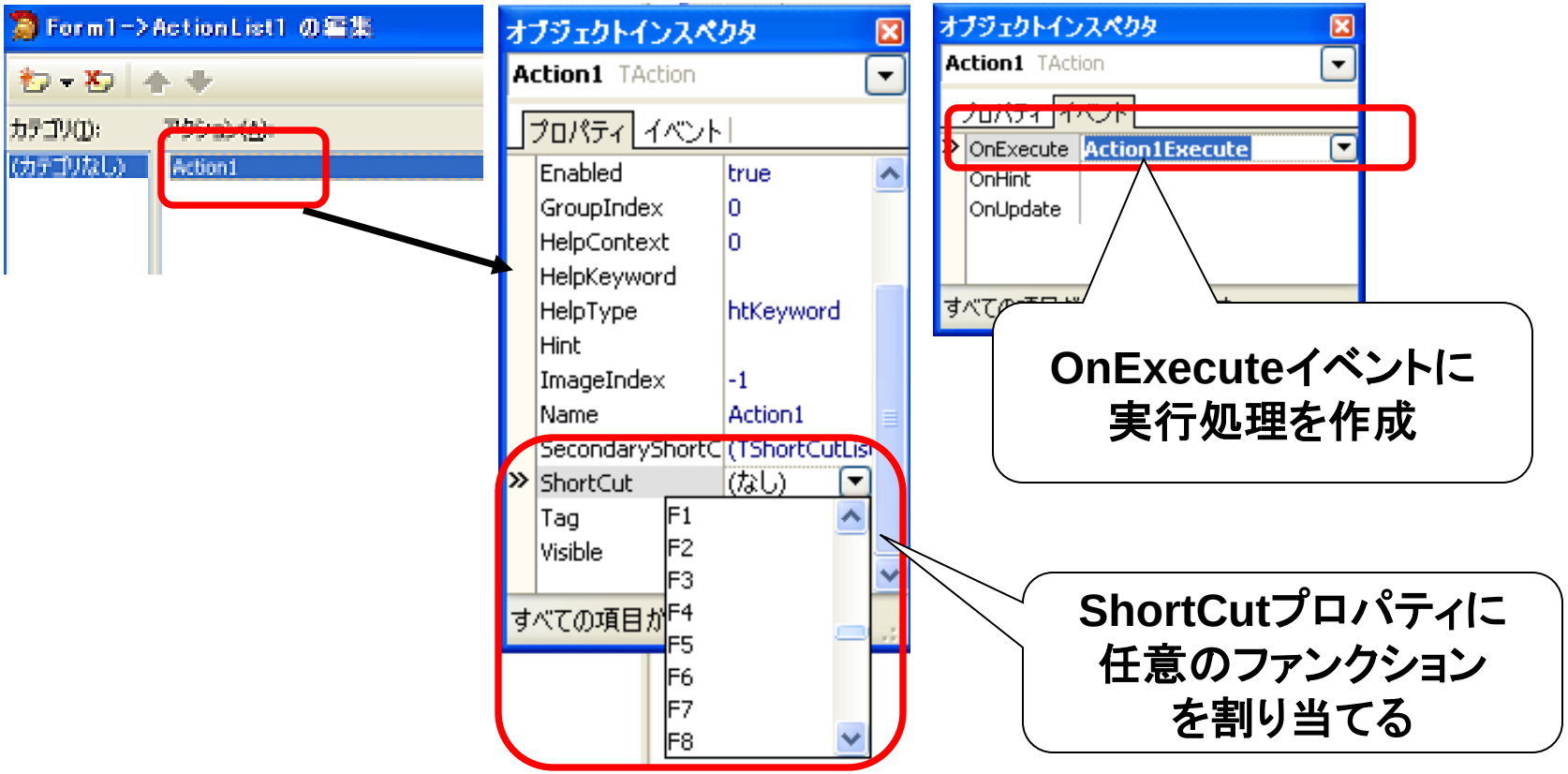
ActionListコンポーネントを使うと、ユーザー操作に
対する応答を一元化できる。

TActionList



【2】ファンクションキー入力を制御しよう！

- Actionでファンクションキーを設定



【2】ファンクションキー入力を制御しよう！

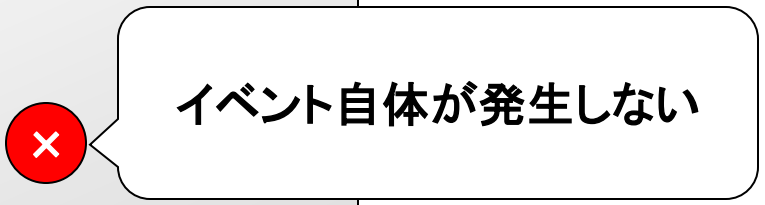
- Actionでファンクションキーを設定

```
procedure TForm1.Action1Execute(Sender: TObject);
begin
  //F1で実行したい処理を記述
  ShowMessage('F1キーが押下されました。(アクション)');
end;
```

【2】Tabキー入力を制御しよう！

- Tabキーを取得するには
  ファンクションキー同様に手法①で考えると
  仮想キーで判断して処理できそうですが・・・

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
Begin
  //仮想キー(VK_XX)で判断
 case Key of
  VK_TAB:begin
           //TABで実行したい処理を記述
           ShowMessage('TABキーが押下されました。');
         end;
  end;  //同様に使用したいキー判断を追記する。
end;
```



【2】Tabキー入力を制御しよう！

- 問題点

ファンクションキーと違ってOnKeyDownイベントで処理できない。

理由としてはOnKeyDown処理が発生する前にTabキーのメッセージが処理済になっているためなので、未処理にしてやる必要がある。

【2】Tabキー入力を制御しよう！

- Tabキーのメッセージを未処理にする。

宣言部

```
private
    { Private declarations }
  public
    { Public declarations }
  protected
    procedure CMDialogKey(var msg: TCMDialogKey);
        Message CM_DIALOGKEY;
```

protected として宣言

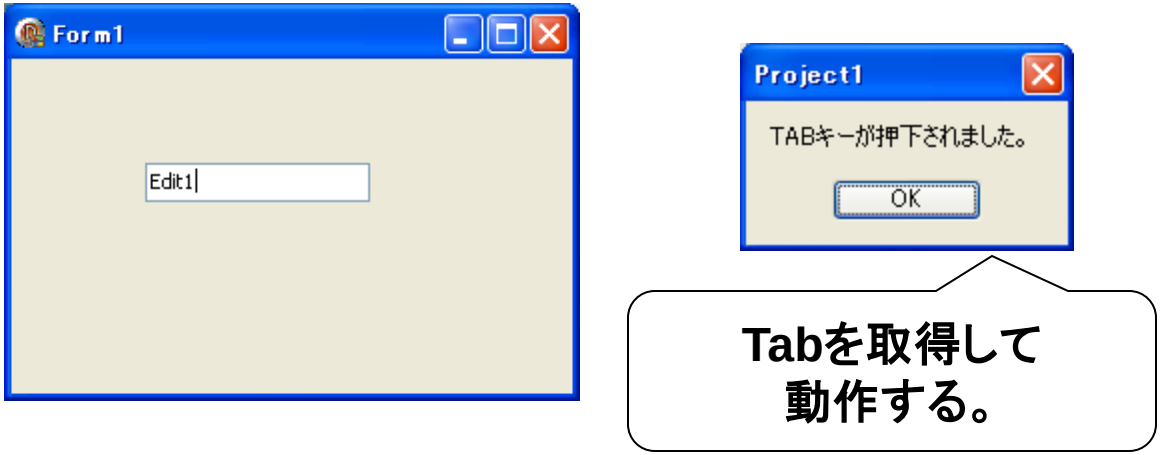
実行部

```
procedure TForm1.CMDialogKey(var msg: TCMDialogKey);
begin
  if (msg.CharCode = VK_TAB) then
  begin
    //ここに直接Tab処理を書く場合はmsg.Resultは1（処理済）に設定
    msg.Result := 0;  //メッセージを処理したか 0:未処理 1:処理済
  end else inherited; //Tab以外は通常の処理
end;
```

msg.Resultを0(未処理)にすることでOnKeyDownイベントが発生する。

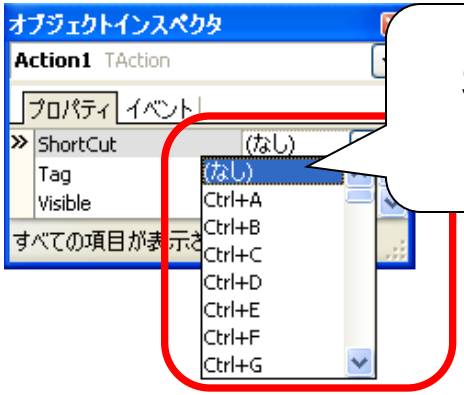
【2】Tabキー入力を制御しよう！

- OnKeyDwonイベントでTabキーが取得/処理できる。



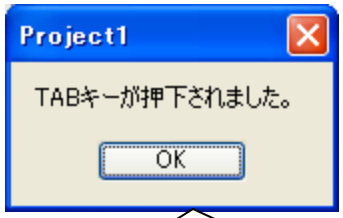
【2】Tabキー入力を制御しよう！

- 手法②ActionListコンポーネントを利用する場合



ShortCutプロパティに
Tabは存在しない

ShortCutプロパティに
直接 'Tab'と入力設定



Tabを取得して動作

【3】ClientDataSetでグループ制御しよう！

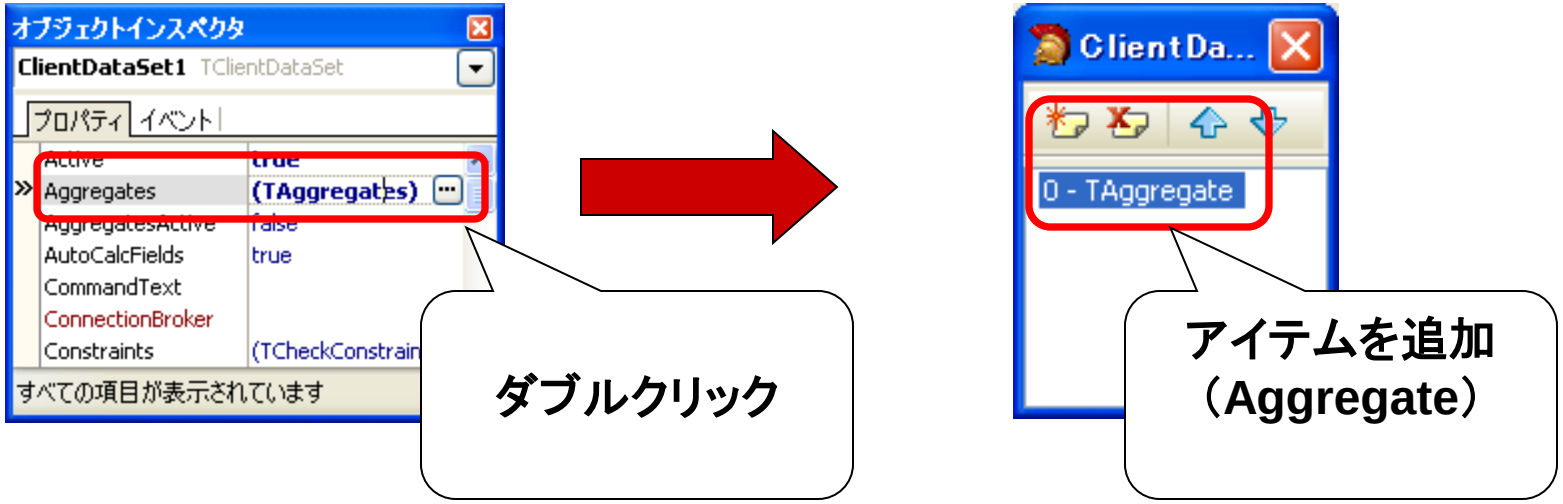
【3】ClientDataSetでグループ制御しよう！

- ClientDataSetのグループ機能を使ってみる。
  例えば値を集計して取得する。

【3】ClientDataSetでグループ制御しよう！

- ClientDataSetのグループ設定を使った算出

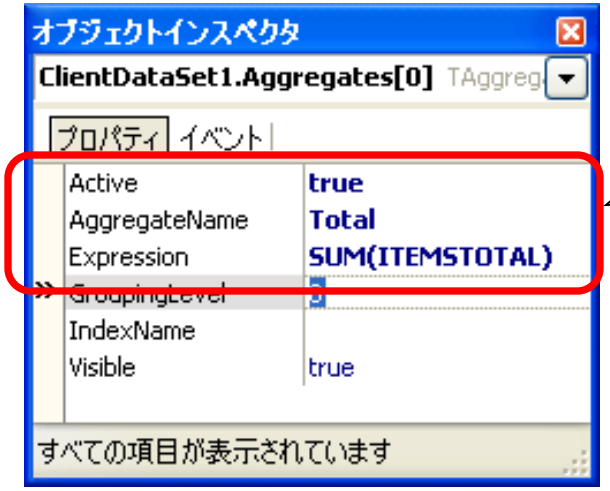
① AggregatesプロパティにTAggregateを追加する。



【3】ClientDataSetでグループ制御しよう！

- ClientDataSetのグループ設定を使った算出

②Aggregateのプロパティを設定する。



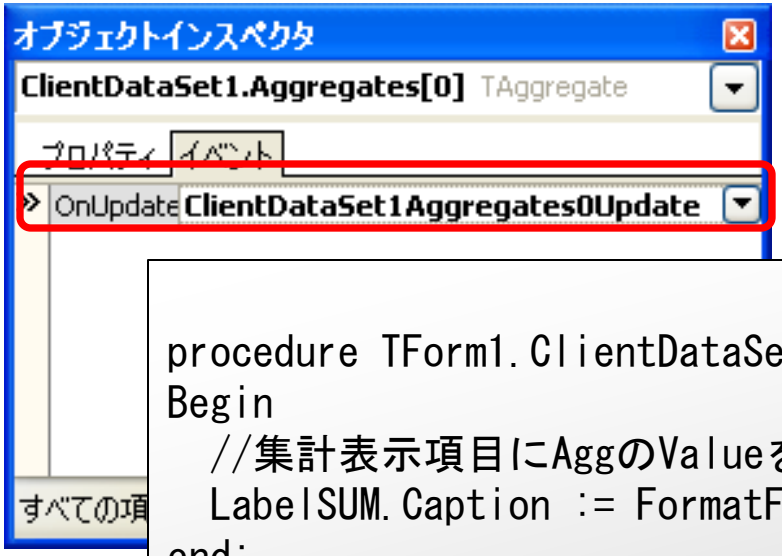
Active をTrueに設定
AggeregateNameを任意で設定
Expressionに集計公式を設定
(ここではSUMで集計)

集計公式で使えるのは
SUM：合計
MIN ：最小値
MAX：最大値
AVG：平均
COUNT：カウント

【3】ClientDataSetでグループ制御しよう！

- ClientDataSetのグループ設定を使った算出

③OnUpdateイベントで集計ロジックを記述する。




```
procedure TForm1.ClientDataSet1AggregatesOUpdate(Agg: TAggregate);
Begin
  //集計表示項目にAggのValueを転送
  LabelSUM.Caption := FormatFloat('#,#',Agg.Value);
end;
```

【3】ClientDataSetでグループ制御しよう！

- 合計、最大、最小、平均などの自動算出が可能

【3】ClientDataSetでグループ制御しよう！

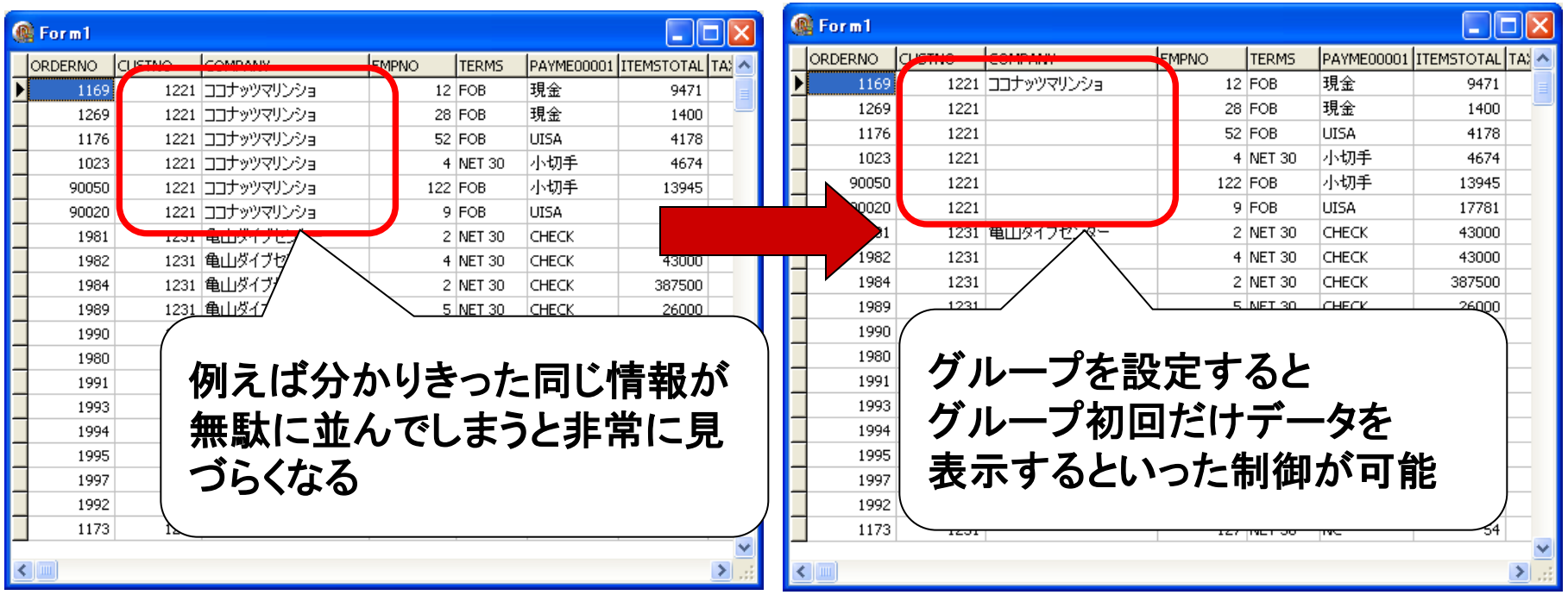
受注No	顧客No	顧客名	受注額	支払方法
1123	1221	ココナッツマリンショップ	¥13,945	小切手
1169	1221		¥9,472	現金
1023	1221		¥4,674	小切手
1076	1221		¥17,781	Uisa
1269	1221		¥1,400	現金
1176	1221		¥4,179	Uisa
1278	1231	ダイブハウスタートル	¥11,568	現金
1073	1231		¥19,414	NC
1160	1231		¥2,207	小切手
1302	1231		¥24,485	現金
1060	1231		¥15,355	小切手
1102	1231		¥2,844	現金
1202	1231		¥4,205	現金
1178	1231		¥5,512	現金
1173	1231		¥54	NC
1152	1351	ダイビングベース新井	¥97,699	現金

合計 ¥2,902,601　最小 ¥54　最大 ¥158,923　平均 ¥14,228

ClientDataSet上で値が更新されると反映も自動算出で行われる

【3】ClientDataSetでグループ制御しよう！

- ClientDataSetのグループ機能を使ってみる
  例えば重複した表示を制御

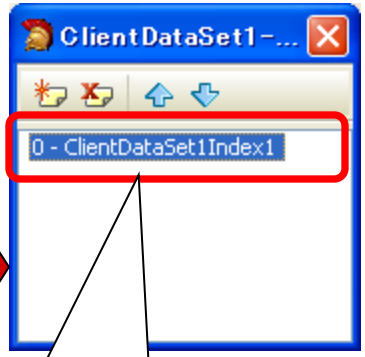


【3】ClientDataSetでグループ制御しよう！

- ClientDataSetのグループ設定を使った表示制御

①ClientDataSetのIndexDefを作成する。
グループになるキー項目をIndexDefに設定する。

ダブルクリック



アイテムを追加
（IndexDef）

・Fields：グループ化する為のキー項目を設定
・Grouplevel：1

【3】ClientDataSetでグループ制御しよう！

- ClientDataSetのグループ設定を使った表示制御

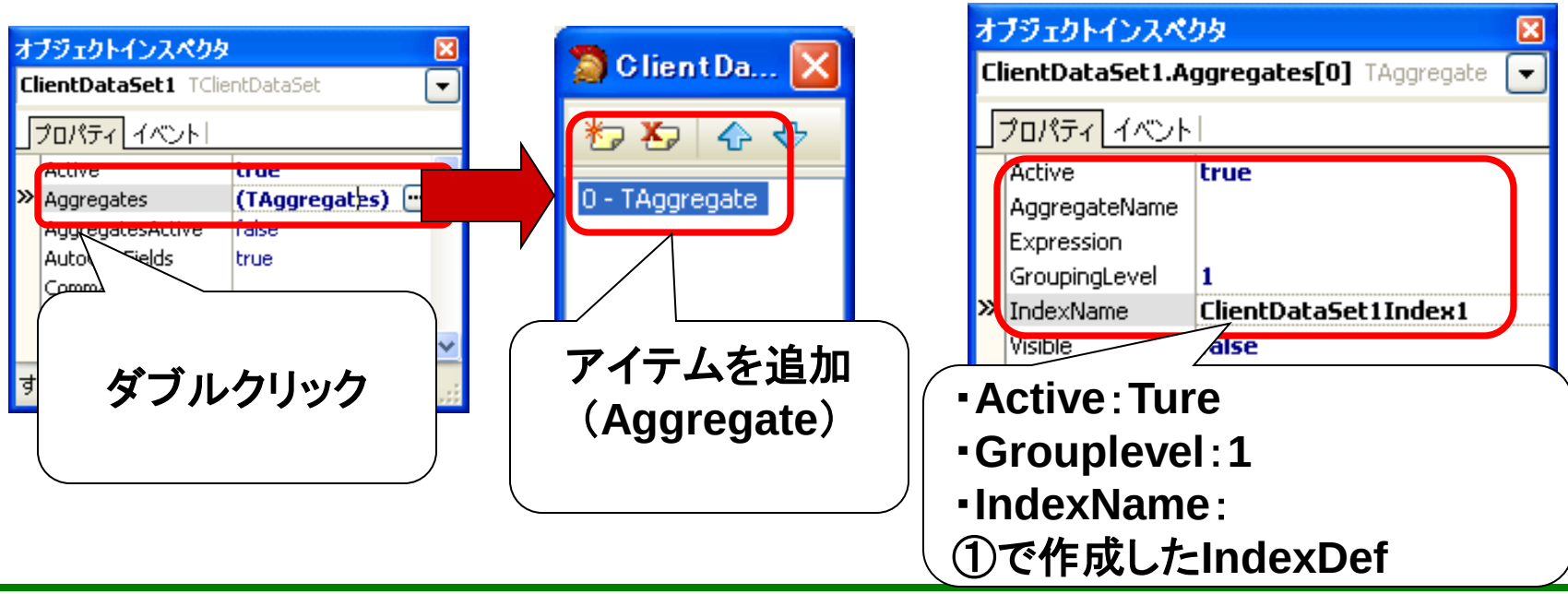
②作成したIndexDefをClientDataSetのIndexNameプロパティに設定する。

①で作成したIndexdefを設定

【3】ClientDataSetでグループ制御しよう！

- ClientDataSetのグループ設定を使った表示制御

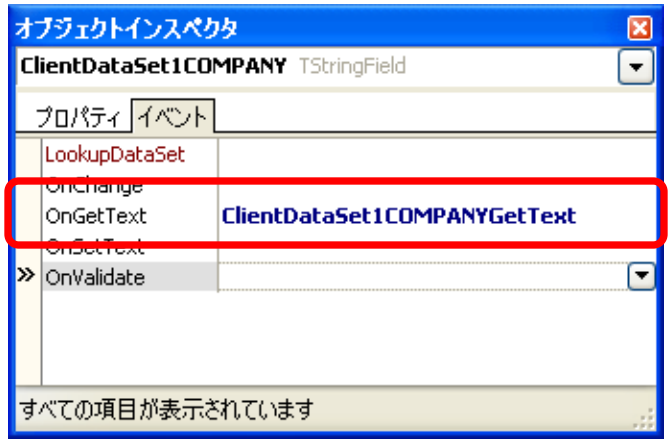
③ AggregatesプロパティでAggregateを追加し、
作成したIndexDefをIndexNameプロパティに設定する。



【3】ClientDataSetでグループ制御しよう！

- ClientDataSetのグループ設定を使った表示制御

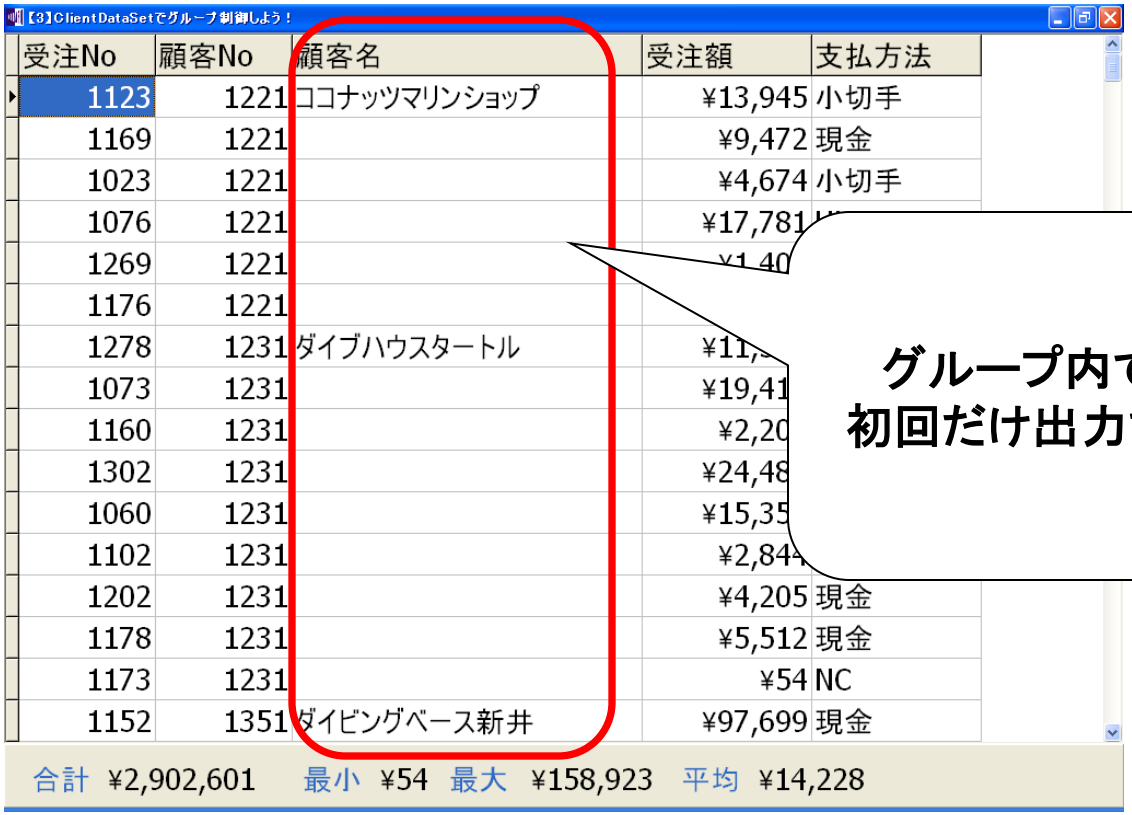
④ClientDataSetの対象のフィールドのOnGetTextイベントでロジックを記述する。




```
procedure TForm1.ClientDataSet1COMPANYGetText(Sender: TField;
 var Text: string;DisplayText: Boolean);
begin
inherited;
  //グループ初回(gbFirst)だけ値を表示
  if gbFirst in ClientDataSet1.GetGroupState(1) then
    Text := Sender.Value
  else
    Text := '';
end;
```

【3】ClientDataSetでグループ制御しよう！

- ClientDataSetのグループ設定を使った表示制御



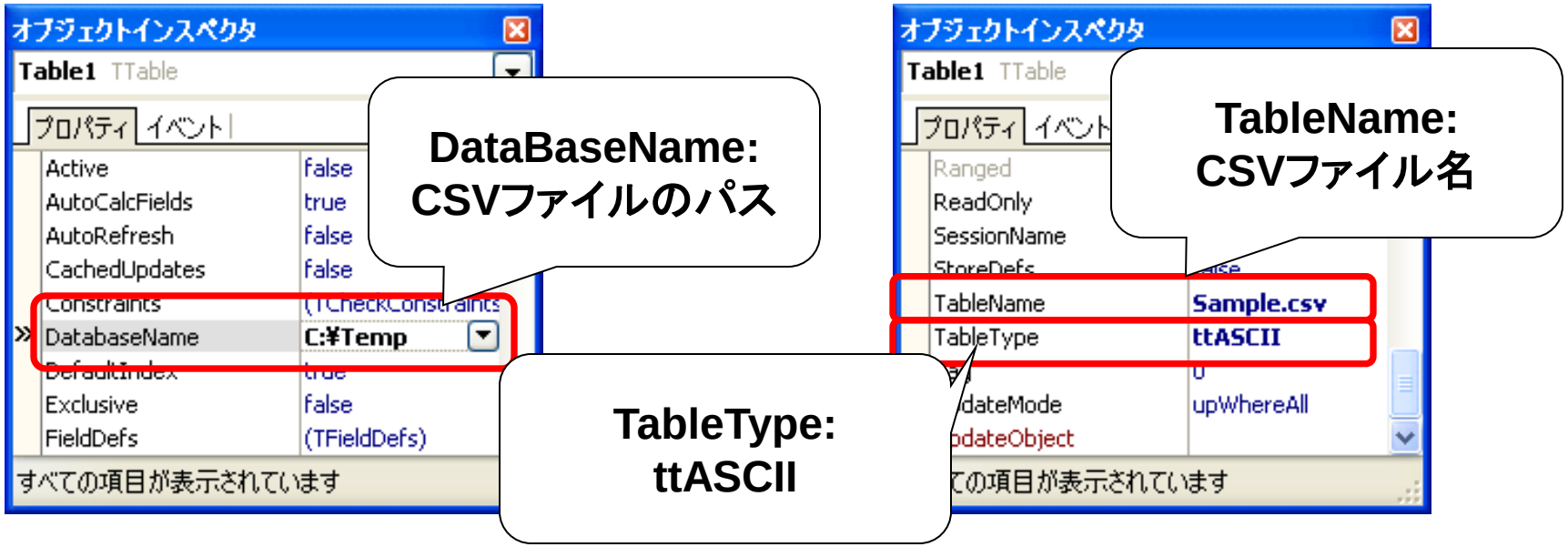
グループ内で重複した表示は
初回だけ出力するよう制御できる

【4】CSVファイルをTableで読み込もう！

【4】CSVファイルをTableで読み込もう！

- CSVファイルをTableで読み込むには

CSVファイルをTableコンポーネントで読み込むには
以下のようにプロパティを設定する。

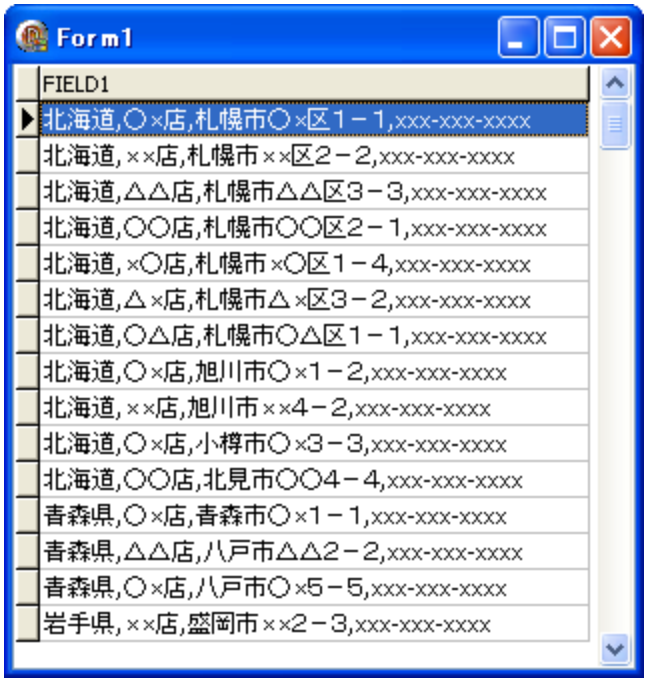


【4】CSVファイルをTableで読み込もう！

- 例えば[地域、店舗名、住所、電話番号]のデータを持つCSVファイルをそのまま読み込んでみる

Sample.csv

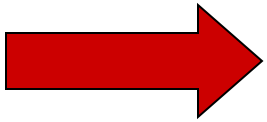
```
北海道,〇×店,札幌市〇×区１－１,xxx-xxx-xxxx
北海道,××店,札幌市××区２－２,xxx-xxx-xxxx
北海道,△△店,札幌市△△区３－３,xxx-xxx-xxxx
北海道,〇〇店,札幌市〇〇区２－１,xxx-xxx-xxxx
北海道,×〇店,札幌市×〇区１－４,xxx-xxx-xxxx
北海道,△×店,札幌市△×区３－２,xxx-xxx-xxxx
北海道,〇△店,札幌市〇△区１－１,xxx-xxx-xxxx
北海道,〇×店,旭川市〇×１－２,xxx-xxx-xxxx
北海道,××店,旭川市××４－２,xxx-xxx-xxxx
北海道,〇×店,小樽市〇×３－３,xxx-xxx-xxxx
北海道,〇〇店,北見市〇〇４－４,xxx-xxx-xxxx

....
```



【4】CSVファイルをTableで読み込もう！

- 問題点

CSVファイルにはDDSのようなフィールド定義がないので項目を正しく区分けして読み込めない。



CSVファイル内容のフィールドを定義したスキーマファイルを用意する必要がある。

スキーマファイルとは，ASCII テーブルとDBとの間でデータをやり取りする場合に使用し，テーブルの構造とその項目のデータ型に関する情報が入ったテキストファイル。拡張子SCH 。

【4】CSVファイルをTableで読み込もう！

- スキーマファイルの構造例

```
[SAMPLE]                          // ファイル名（拡張子なし）
FILETYPE = Delimited              // 形式：Delimited または Fixed
CHARSET = ascii                   // 言語ドライバ名
DELIMITER = “                     // 文字項目の囲み文字(省略可)
SEPARATOR = ,                     // 項目間 の区切り文字
Field1 = REGION,CHAR,12,0,0       // 以下項目（フィールド）情報
Field2 = STORE,CHAR,10,0,13
Field3 = ADDR,CHAR,40,0,24
Field4 = TELNO,CHAR,12,0,65
```

※スキーマファイル内の情報は，大文字と小文字を区別する。

【4】CSVファイルをTableで読み込もう！

- スキーマファイルの設定詳細

FILETYPE

テキストファイルの形式として，Fixed（固定長テキスト）または Delimited（カンマ付きテキスト）のどちらかを指定。

CHARSET

使用する言語ドライバの名前を指定。

DELIMITER

各文字フィールド（英数字または文字）を囲む半角一文字を指定。

【4】CSVファイルをTableで読み込もう！

- スキーマファイルの設定詳細

SEPARATOR

各項目を区切る半角の文字を一文字指定

項目情報

Field
(= 項目名, データ型, 文字数, 小数点以下桁数, 開始位置)
テーブルの項目（列）の属性を指定。
各行は「FieldX = 」で開始。
ここで X は項目番号（Field1, Field2 など）

【4】CSVファイルをTableで読み込もう！

- スキーマファイルで使用できるデータ型

CHAR	文字
NUMBER	64 ビット浮動小数点数
BOOL	論理値 (T または F)
LONGINT	32 ビット倍長整数
DATE	日付型項目
TIME	時刻型項目
TIMESTAMP	日付＋時刻型項目

【4】CSVファイルをTableで読み込もう！

- スキーマファイルをCSVファイルと同じパスに配置すると正しくフィールドを区切って読み込める

Sample.csv

```
北海道,〇×店,札幌市〇×区１－１,xxx-xxx-xxxx
北海道,××店,札幌市××区２－２,xxx-xxx-xxxx
北海道,△△店,札幌市△△区３－３,xxx-xxx-xxxx
北海道,〇〇店,札幌市〇〇区２－１,xxx-xxx-xxxx
```

Sample.SCH

```
[Sample]
FILETYPE = Delimited
CHARSET = ascii
SEPARATOR = ,
Field1 = REGION,CHAR,12,0,0
Field2 = STORE,CHAR,10,0,13
Field3 = ADDR,CHAR,40,0,24
Field4 = TELNO,CHAR,12,0,65
```

CSVファイルを
DBファイルとして読み込み

【5】ExcelファイルをADOTableで読み込もう！

【5】ExcelファイルをADOTableで読み込もう！

- 同じようにExcelをTableで読み込むことはできないか

ExcelをTableで読み込むためには、クライアントごとにPC上のODBCで設定が必要になるので、ADO接続を利用すると簡単に実現できる。

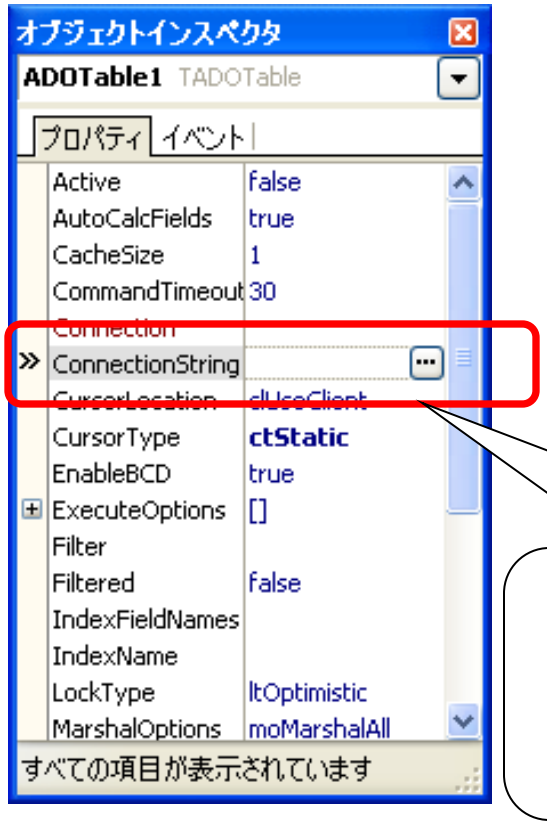
【5】ExcelファイルをADOTableで読み込もう！

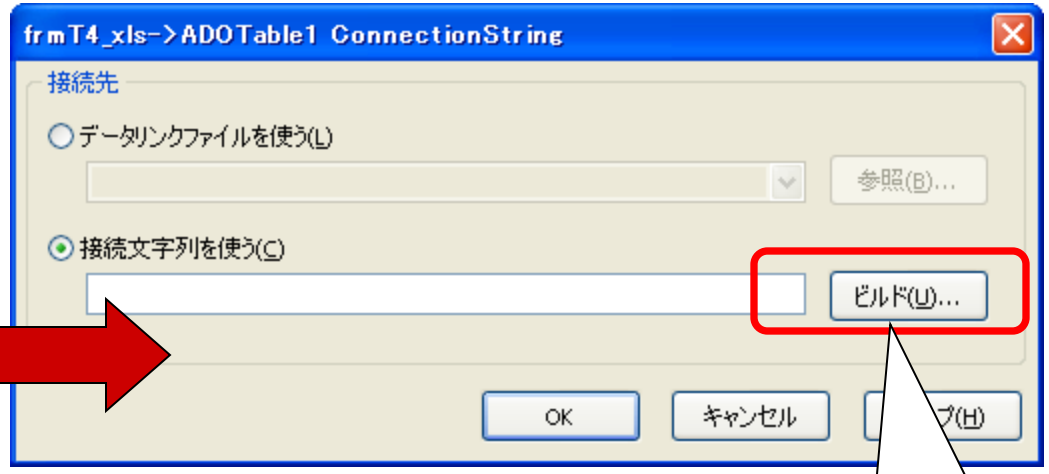
【前提】
ADOでExcelを読み込む場合、1行目が列名として扱われる

	A	B	C	D
1	地域	店舗	住所	電話番号
2	北海道	○×店	札幌市○×区1－1	xxx-xxx-xxxx
3	北海道	××店	札幌市××区2－2	xxx-xxx-xxxx
4	北海道	△△店	札幌市△△区3－3	xxx-xxx-xxxx
5	北海道	○○店	札幌市○○区2－1	xxx-xxx-xxxx
6	北海道	×○店	札幌市×○区1－4	xxx-xxx-xxxx
7	北海道	△×店	札幌市△×区3－2	xxx-xxx-xxxx
8	北海道	○△店	札幌市○△区1－1	xxx-xxx-xxxx
9	北海道	○×店	旭川市○×1－2	xxx-xxx-xxxx
10	北海道	××店	旭川市××4－2	xxx-xxx-xxxx
11	北海道	○×店	小樽市○×3－3	xxx-xxx-xxxx
12	北海道	○○店	北見市○○4－4	xxx-xxx-xxxx
13	青森県	○×店	青森市○×1－1	xxx-xxx-xxxx
14	青森県	△△店	八戸市△△2－2	xxx-xxx-xxxx
15	青森県	○×店	八戸市○×5－5	xxx-xxx-xxxx
16	岩手県	××店	盛岡市××2－3	xxx-xxx-xxxx
17	宮城県	○×店	仙台市○×区3－3	xxx-xxx-xxxx
18	宮城県	×△店	仙台市×△区4－2	xxx-xxx-xxxx
19	宮城県	○○店	古川市○○1－3	xxx-xxx-xxxx
20	秋田県	△×店	秋田市△×3－3	xxx-xxx-xxxx

1行目が項目名

【5】ExcelファイルをADOTableで読み込もう！

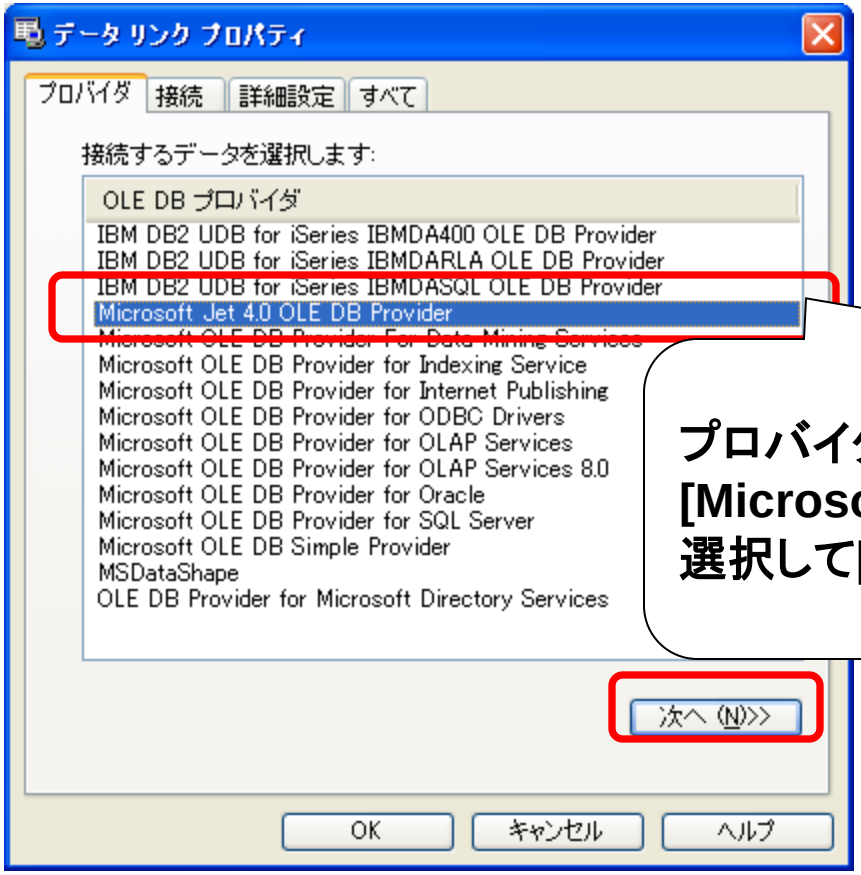




ADOConectionや
ADOTableの
ConnectionStringプロパティ
で接続設定

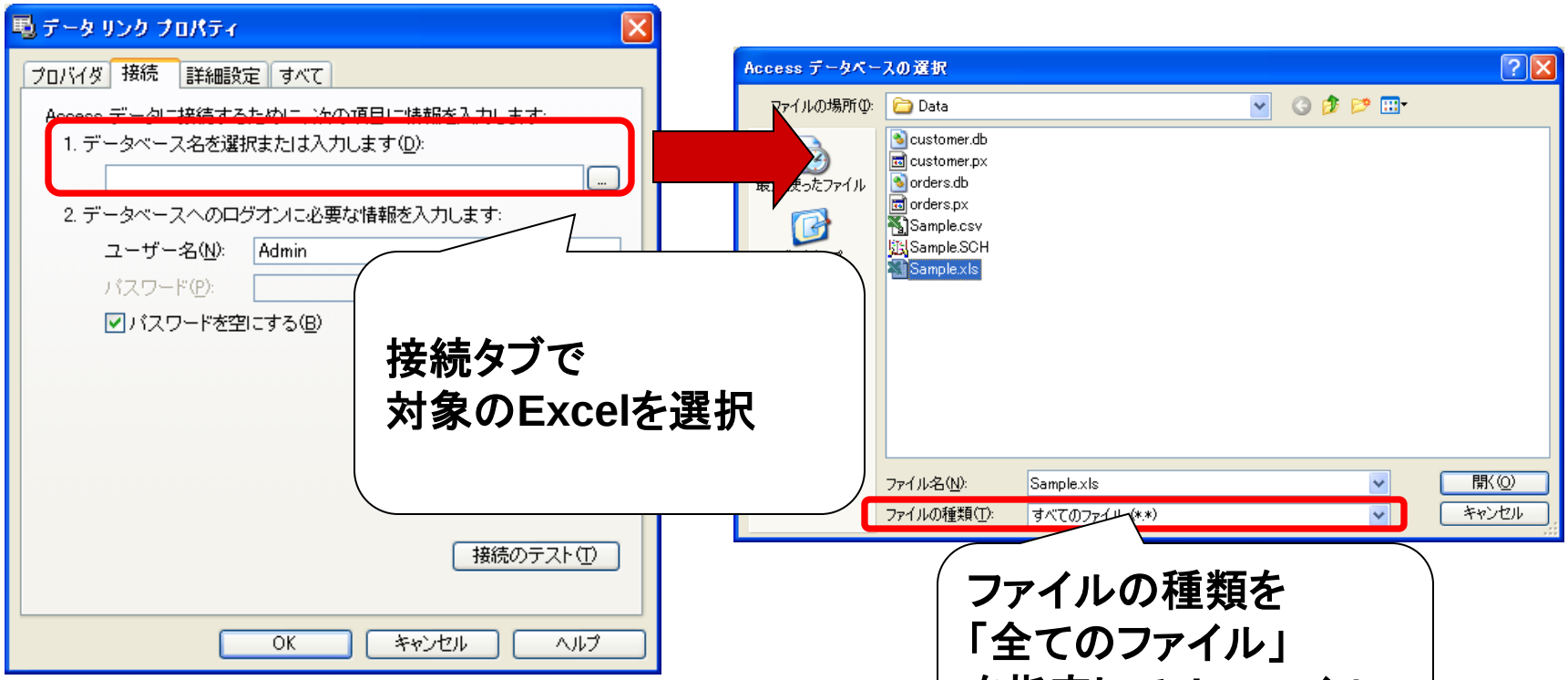
[ビルド]をクリック

【5】ExcelファイルをADOTableで読み込もう！

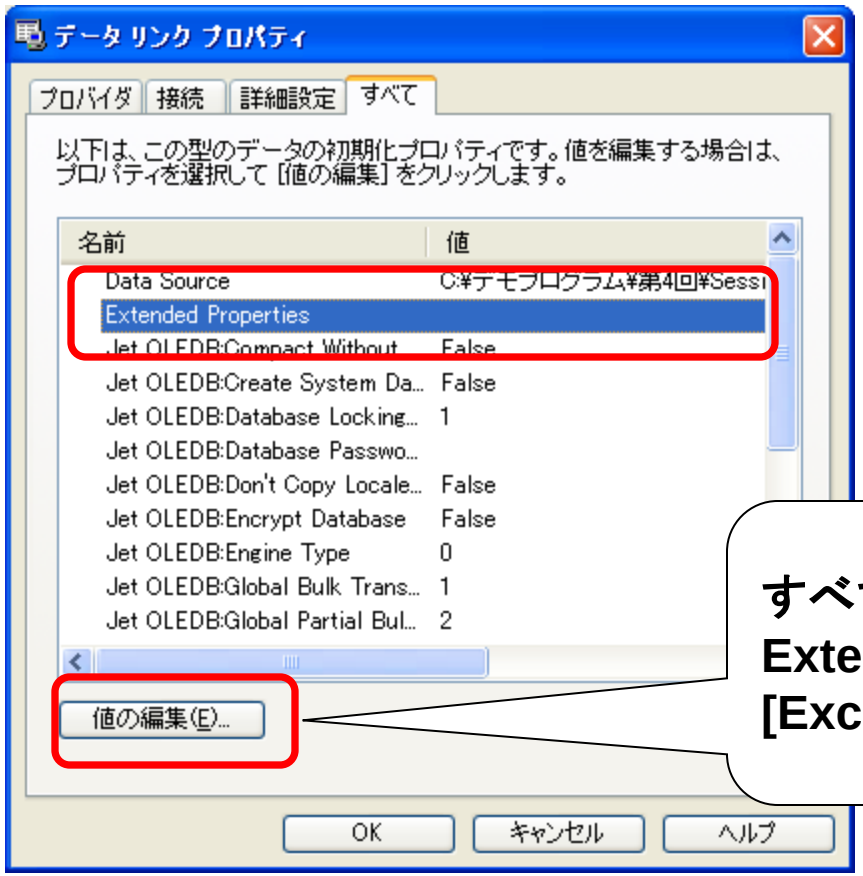


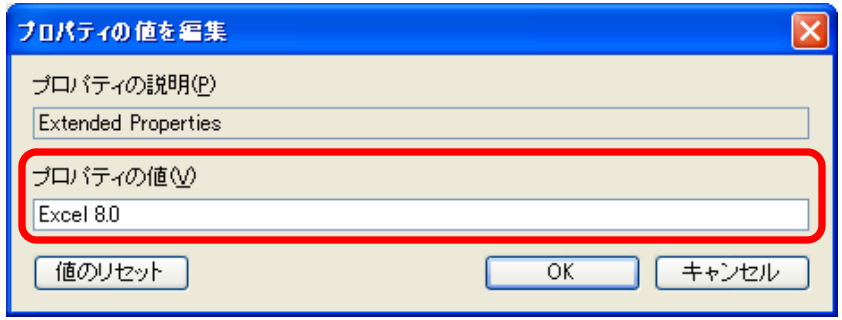
プロバイダタブで
[Microsoft Jet 4.0 OLE DB Provider]を
選択して[次へ]

【5】ExcelファイルをADOTableで読み込もう！



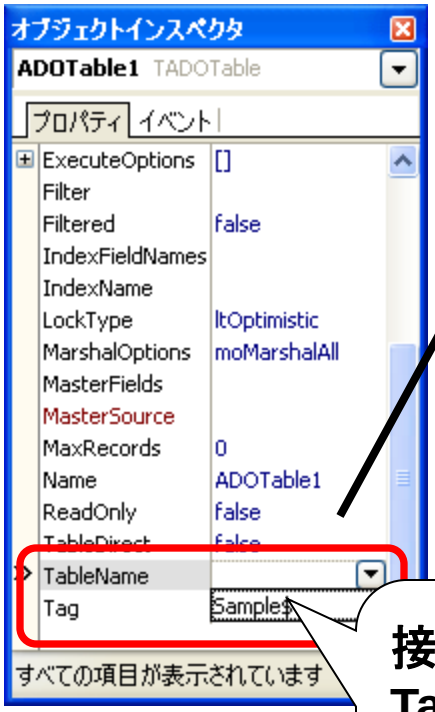
【5】ExcelファイルをADOTableで読み込もう！

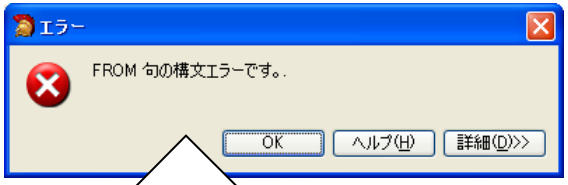




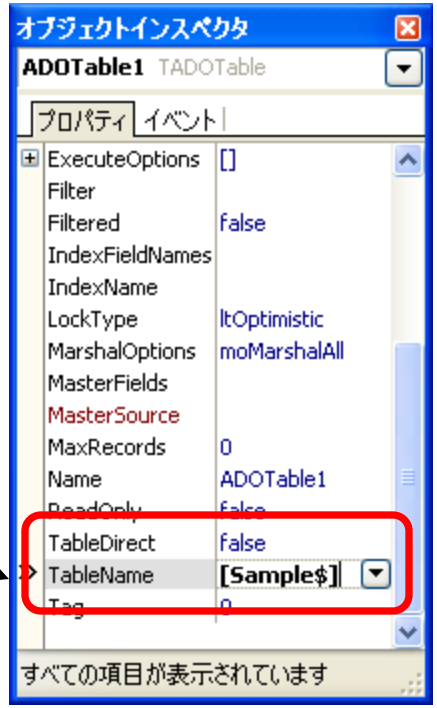
すべてタブで
Extended Propertiesを選択して値の編集。
[Excel 8.0]で設定してOK

【5】ExcelファイルをADOTableで読み込もう！



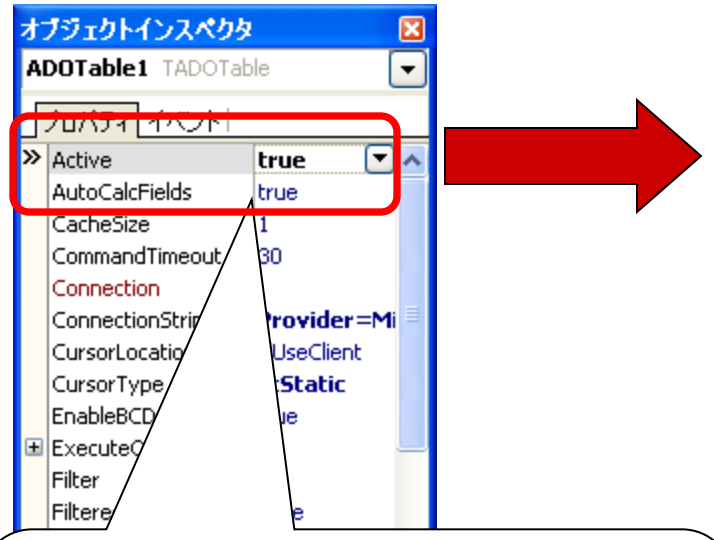


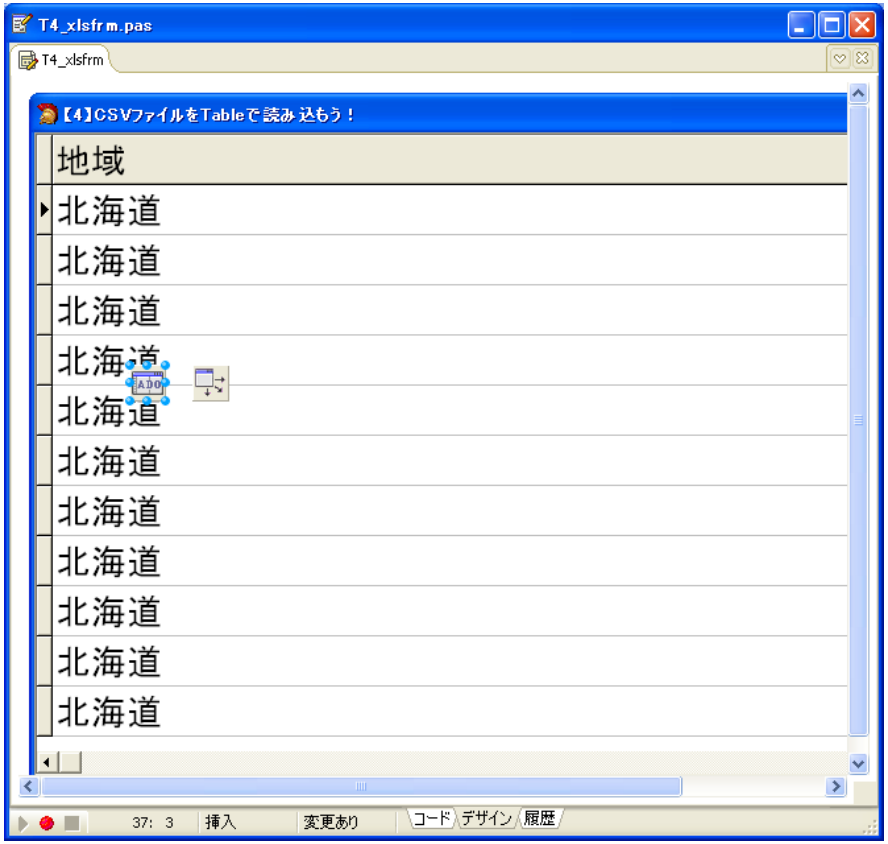
Sample$のままアクセスすると
エラーになるので
Sheet名を[]で囲って指定する
例）[Sample$]



接続設定が完成すると
TableNameプロパティにxlsのシート
がリスト表示されるようになる。
例）Sample$

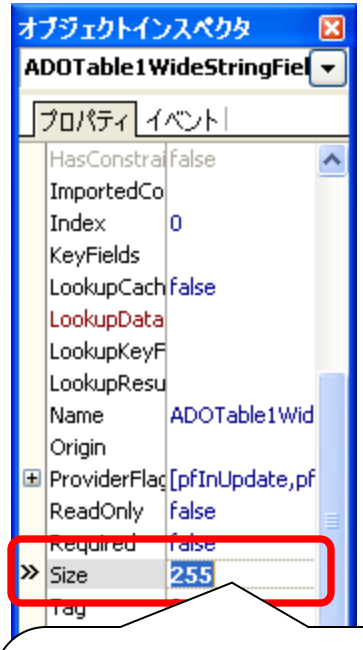
【5】ExcelファイルをADOTableで読み込もう！





ActiveをTrueにすると Excelがファイルとして読み込めるようになる

【5】ExcelファイルをADOTableで読み込もう！



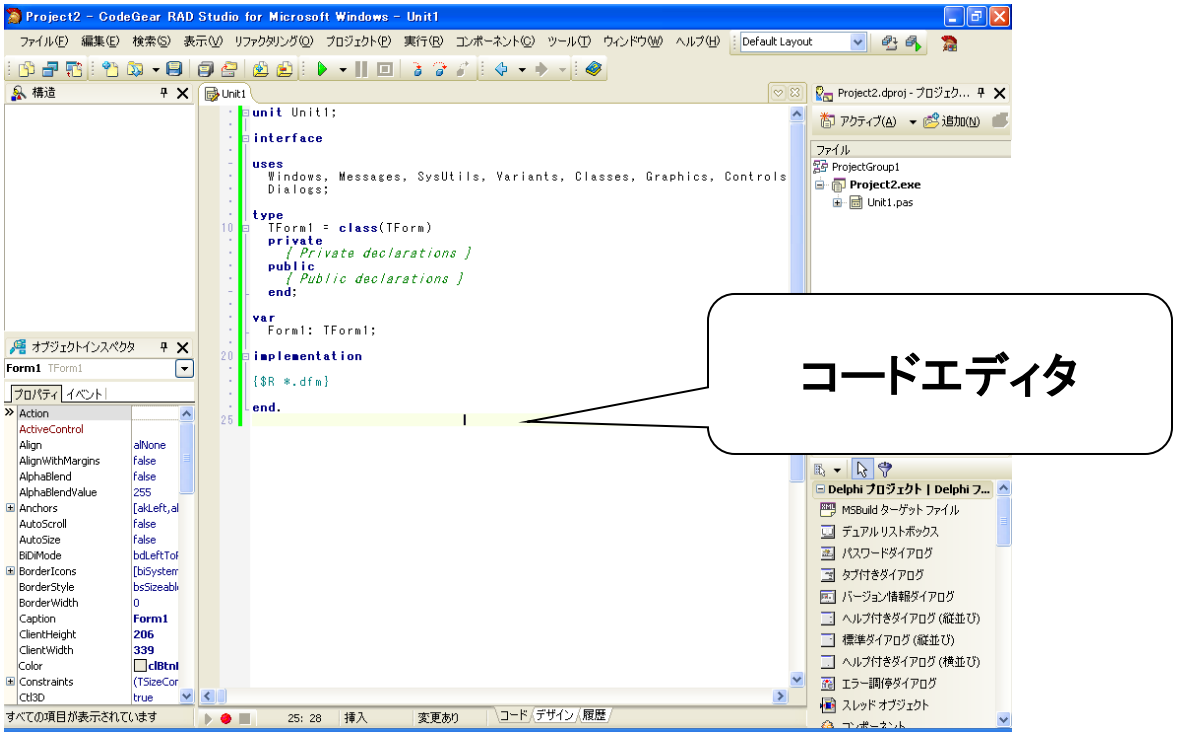
ディフォルトで項目のサイズが255になっているので、項目のSizeプロパティを適切な長さに設定する

CSVと同じように読込みを実現

【お持ち帰りトピック】開発で役立つショートカット機能

【お持ち帰りトピック】開発で役立つショートカット機能

- 知っているとコーディング効率が上がる
  ショートカットをピックアップ



【お持ち帰りトピック】開発で役立つショートカット機能

- 編集系

ショートカット	動作
Ctrl+/	選択行に//(コメント)追加、解除。(Delphi2005以降で有効)
Ctrl+I	タブ文字を挿入。
Ctrl+N	改行を挿入。
Ctrl+Y	選択行を削除。
Ctrl+Shift+I	選択されたブロックをインデント。
Ctrl+Shift+U	選択されたブロックのインデント解除。
[Alt]+[Backspace]	元に戻す。
[Alt]+[Shift]+[Backspace]	やり直し。

【お持ち帰りトピック】開発で役立つショートカット機能

移動系

ショートカット	動作
Ctrl+クリック	宣言部に移動。
Ctrl+Shift+ ↑	宣言⇔実装に相互移動。
Ctrl+Shift+ ↓	宣言⇔実装に相互移動。
Ctrl+数字キー(0～9指定)	ブックマーク位置に移動。。
Ctrl+Shift+数字キー(0～9指定)	カーソル位置にブックマークを設定。
Ctrl+PgDn	画面の最下部に移動。
Ctrl+PgUp	画面の最上部に移動。

【お持ち帰りトピック】開発で役立つショートカット機能

- その他

ショートカット	動作
F1	選択箇所のヘルプ検索。
F12	フォームとその関連するユニットを切り替える。
Ctrl+Shift+Enter	選択内容の使用箇所を検索。
Ctrl+Enter	カーソル位置のファイルを開く。
Ctrl+J	テンプレートメニューを開く。
Ctrl+F12	ユニットのリストを表示。
Ctrl+Shift+C	クラス宣言に対応するクラス補完。
Ctrl+Space	コード補完を開く。