

【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 9.0

株式会社ミガロ.
RAD事業部 技術支援課
吉原 泰介

【アジェンダ】



お客様より年間1,000件以上お問合せ頂いている
テクニカルサポートからの技術フィードバック！

Q1. PageControl応用テクニック

Q2. DLLモジュールの開発手法

■ Q1. PageControl応用テクニック

【質問】

PageControlコンポーネントで画面を分割したいのですが、固定の画面設計しかできないのでしょうか？

【回答】

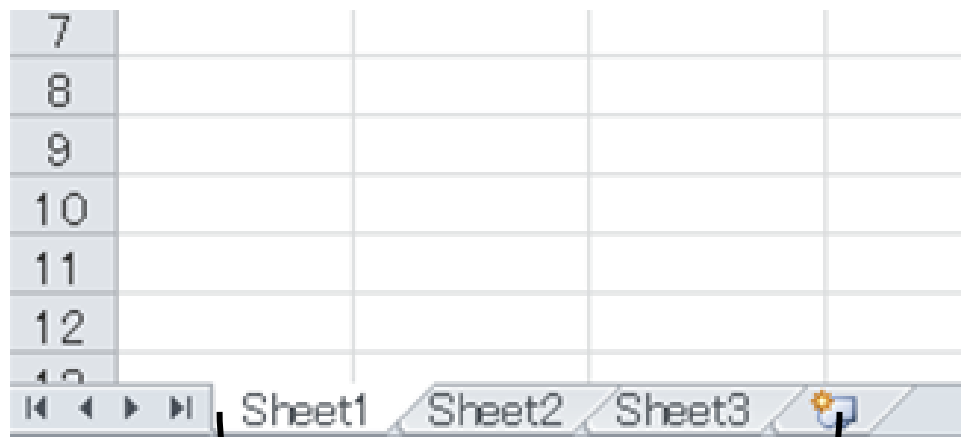
PageControlコンポーネントはExcelのシートのように複数画面を1画面内で切替できるので、情報量が多い画面設計で非常に便利です。
通常、パネルのように固定の部品として利用しますがプログラムを工夫すれば、固定でない活用もできます。

■ Q1. PageControl応用テクニック

- PageControlコンポーネントとは



Microsoft Excel



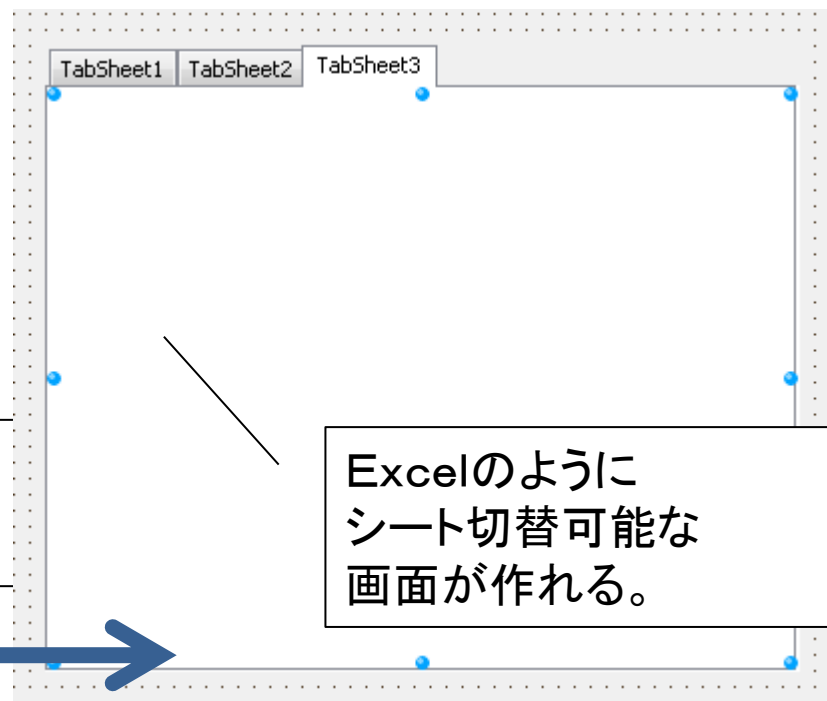
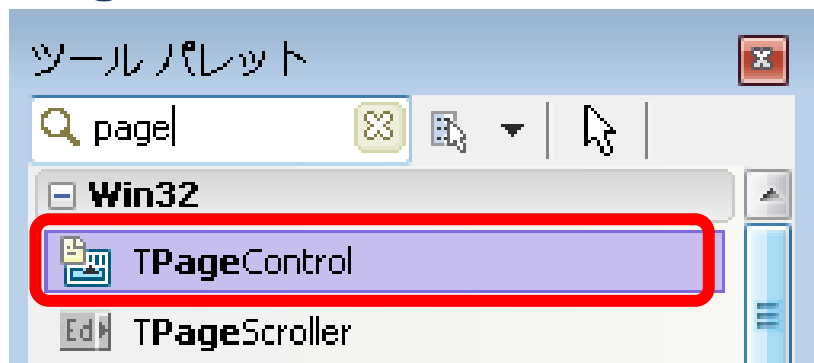
PageControlコンポーネント



1つの画面で、複数のシートを切り替えることで
たくさんの情報量を表示することができる。

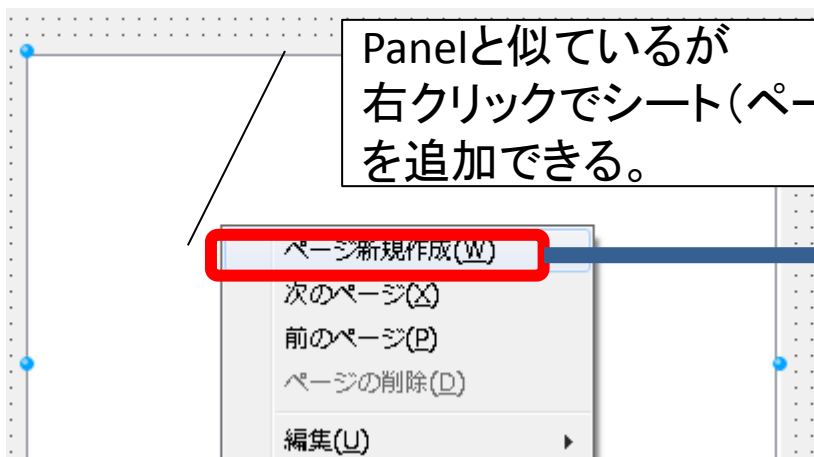
■ Q1. PageControl応用テクニック

• PageControlコンポーネントの使い方



Panelと似ているが
右クリックでシート(ページ)
を追加できる。

Excelのように
シート切替可能な
画面が作れる。



■ Q1. PageControl応用テクニック

- 関連情報を1画面に集約

The screenshots demonstrate the integration of various data points into a single view:

- Order Details:** 受注番号 (MG00264), 受注日 (2011/11/22), 受注品番 (SU02-GF), 倉庫区分 (01), 取引先 (0082), 納入先 (0082), 伝票区分 (E2), 取引区分 (直販).
- Customer Details:** 顧客番号 (00082), 会社名 (株式会社ミガロ), 〒 (556-0017), 都道府県 (大阪府), 住所 (大阪市浪速区湊町2-1-57), 電話番号 (06-6631-8601), FAX番号 (06-6631-8603).
- Employee Details:** 担当CD (0244), 担当者名 (里中 雄一), 部署CD (C03), 部署名 (第3営業部), 担当地区 (港区).
- Inventory & Charts:** Table for SU02-FG showing monthly inventory levels (4月: 4,423, 5月: 221, 6月: 2,355, 7月: 2,881) and corresponding bar and line charts.

PageControlは画面構成に便利なコンポーネントですが、
固定のレイアウトだけでなく、今回はプログラムからの応用操作
をご紹介します

■ Q1. PageControl応用テクニック

PageControlの応用テクニック

- ①シートをクリア(全て削除)する
- ②シートを作成する
- ③シートに項目(部品)を作成する
- ④フレームを利用してシート項目を一括作成する
- ⑤データから動的にシートを作成する

補足

シートをドラッグ&ドロップで入れ替える

■ Q1. PageControl応用テクニック

- ①シートをクリア(全て削除)する

受注情報 | 取引先 | 担当者 | 在庫状況

受注番号 MG00264
受注日 2011/11/22
受注品番 SU02-GF 倉庫区分 01
取引先 0082 納入先 0082
伝票区分 E2 備考 11/25までに納品
取引区分 直販

品番	品名	仕番	在庫
1221	ココナッツマリンショップ	太島町4-976	
1231	ダイブハウスタートル	東荻5-8-7	
1351	ダイビングベース新井	新井2-14-3	新井2-1
1354	アクアダイビングワールド	明太区曾根541	

シートを削除

■ Q1. PageControl応用テクニック

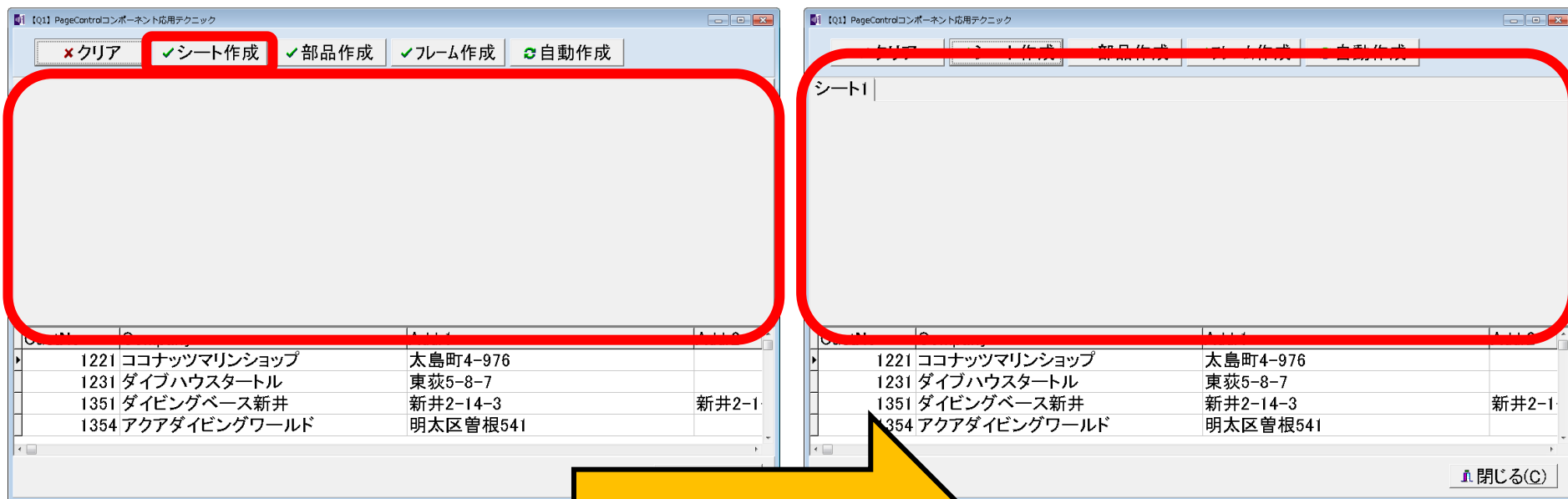
クリアボタンの処理(ソース)

```
procedure TfrmQ1.btnClearClick(Sender: TObject);
begin
  //Pageコントロールのシート数が0になるまでループ
  while PageControl1.PageCount <> 0 do
  begin
    //シートを削除
    PageControl1.ActivePage.Destroy;
  end;
end;
```

ポイント:
シート数が0になるまで
削除をする。

■ Q1. PageControl応用テクニック

- ②シートを作成する
固定レイアウトではなく、自在にシートを作成



シートを作成

■ Q1. PageControl応用テクニック

シート作成ボタンの処理(ソース)

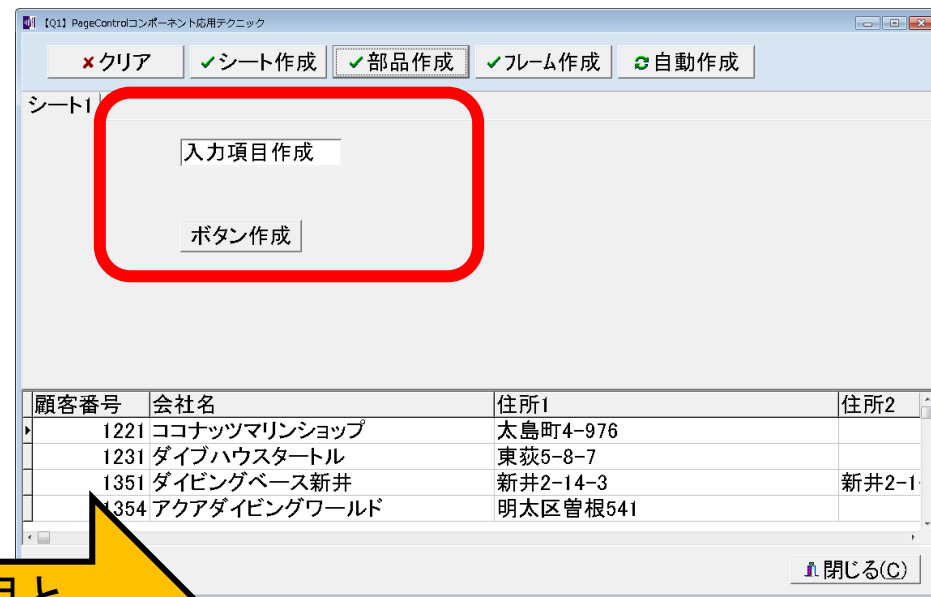
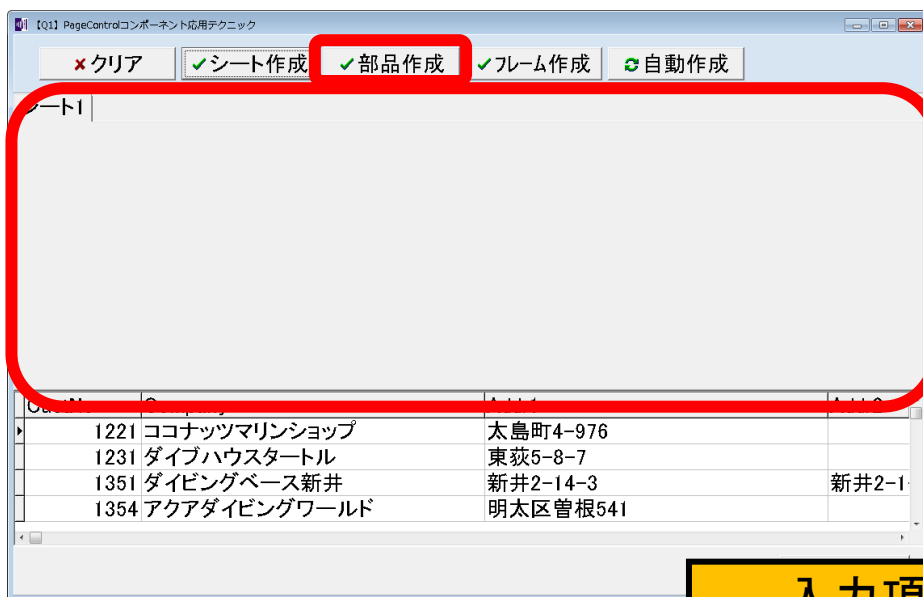
```
procedure TfrmQ1.btnManualClick(Sender: TObject);
var
  TabSheet: TTabSheet; //シート追加用
begin
  //シートを生成
  TabSheet := TTabSheet.Create(PageControl1);
  //PageControlに追加
  TabSheet.PageControl := PageControl1;
  //シート名を設定
  TabSheet.Caption := 'シート' + IntToStr(PageControl1.PageCount);
  //シート切り替え
  PageControl1.ActivePageIndex := PageControl1.PageCount - 1;
end;
```

ポイント:
シートタイトルは
見分けがつくように
シート番号で設定

応用すればデータの数にあわせてシートを作成できる(後半)

■ Q1. PageControl応用テクニック

- ③シートに項目(部品)を作成する
シートの中身を作成する



入力項目と
ボタンを作成

■ Q1. PageControl応用テクニック

部品作成ボタンの処理(ソース)

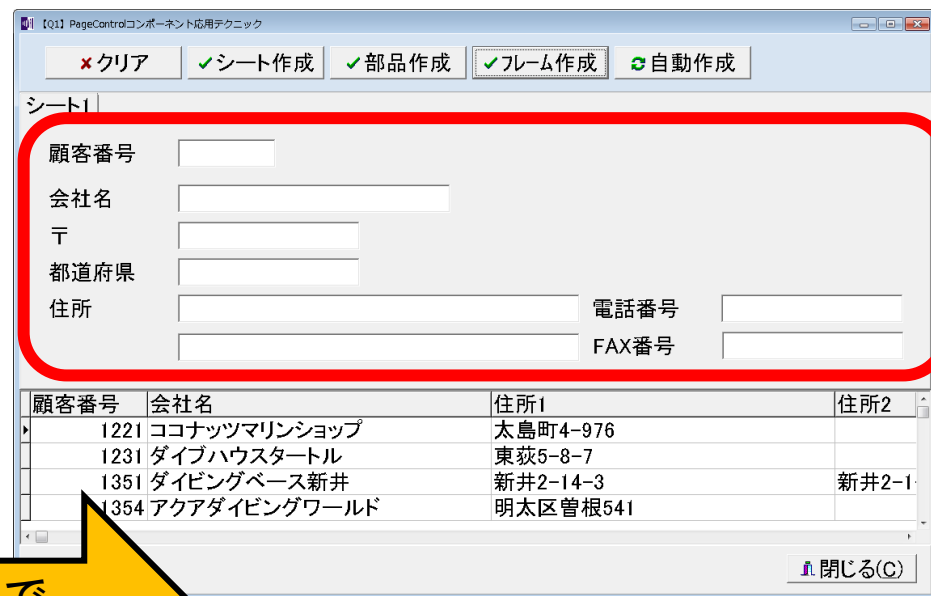
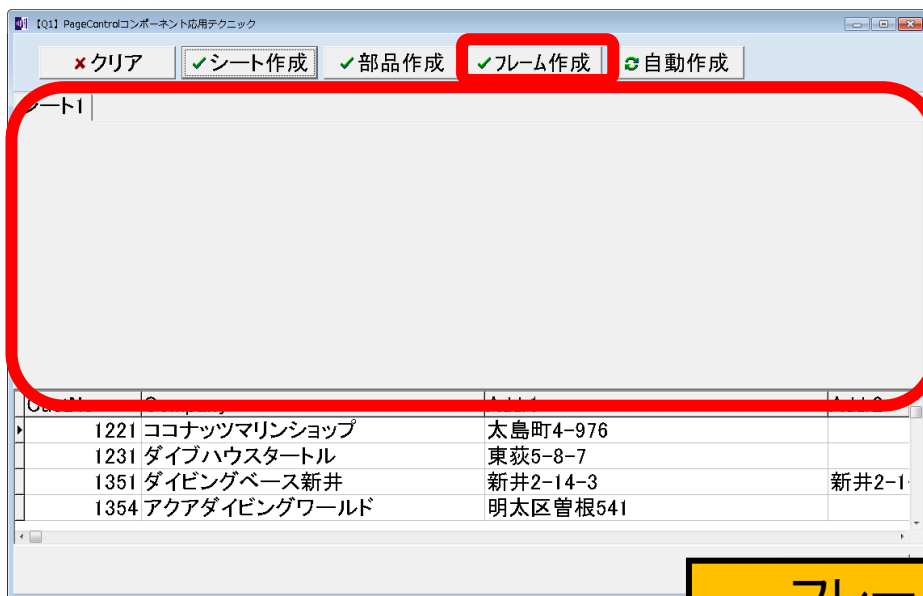
```
procedure TfrmQ1.btnCmpClick(Sender: TObject);
var
  Edit   : TEdit;      // 入力項目追加用
  Button : TButton;    // ボタン追加用
begin
  //入力項目の追加
  Edit      := TEdit.Create(PageControl1.ActivePage); //生成
  Edit.Parent := PageControl1.ActivePage;           //シートにセット
  Edit.Top   := 20;                                   //位置設定
  Edit.Left  := 192;                                  //位置設定
  Edit.Width := 200;                                  //サイズ設定
  Edit.Height := 35;                                  //サイズ設定
  Edit.Text  := '入力項目作成';                       //表示内容設定
  //ボタンの追加
  Button     := TButton.Create(PageControl1.ActivePage); //生成
  Button.Parent := PageControl1.ActivePage;           //シートにセット
  Button.Top   := 120;                                 //位置設定
  Button.Left  := 192;                                 //位置設定
  Button.Width := 150;                                 //サイズ設定
  Button.Height := 40;                                 //サイズ設定
  Button.Caption := 'ボタン作成';                     //表示内容設定
end;
```

部品の生成やサイズ、位置を
ひとつひとつプログラムする
のは

かなり面倒！

■ Q1. PageControl応用テクニック

- ④ フレームを利用してシート項目を一括作成する。



フレームで
シート内容を作成

■ Q1. PageControl応用テクニック

フレーム作成ボタンの処理(ソース)

```
procedure TfrmQ1.btnFraClick(Sender: TObject);  
var  
    Frame:TFrame1; //フレーム追加用  
begin  
    //フレームを生成  
    Frame := TFrame1.Create(PageControl1.ActivePage);  
    //シートにフレームをセット  
    Frame.Parent := PageControl1.ActivePage;  
end;
```

ポイント:

PageControlにあらかじめ用意しているフレームをセットするだけ。

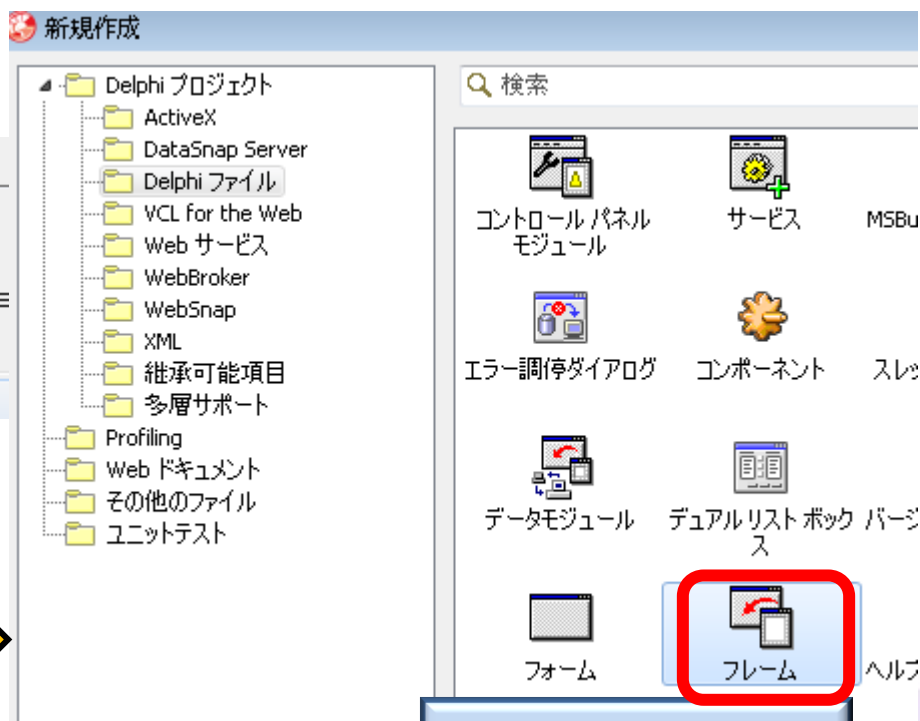
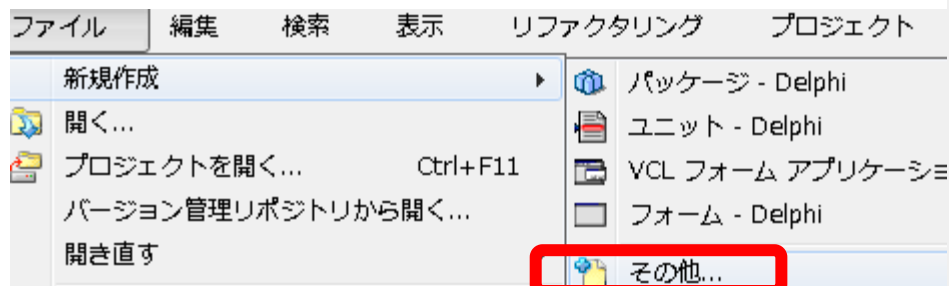
部品をひとつひとつ作るより簡単だけど
フレームとは何か？

■ Q1. PageControl応用テクニック

• フレームとは？

画面(フォーム)をコンポーネントのように部品化できる機能

ファイル>新規作成>その他



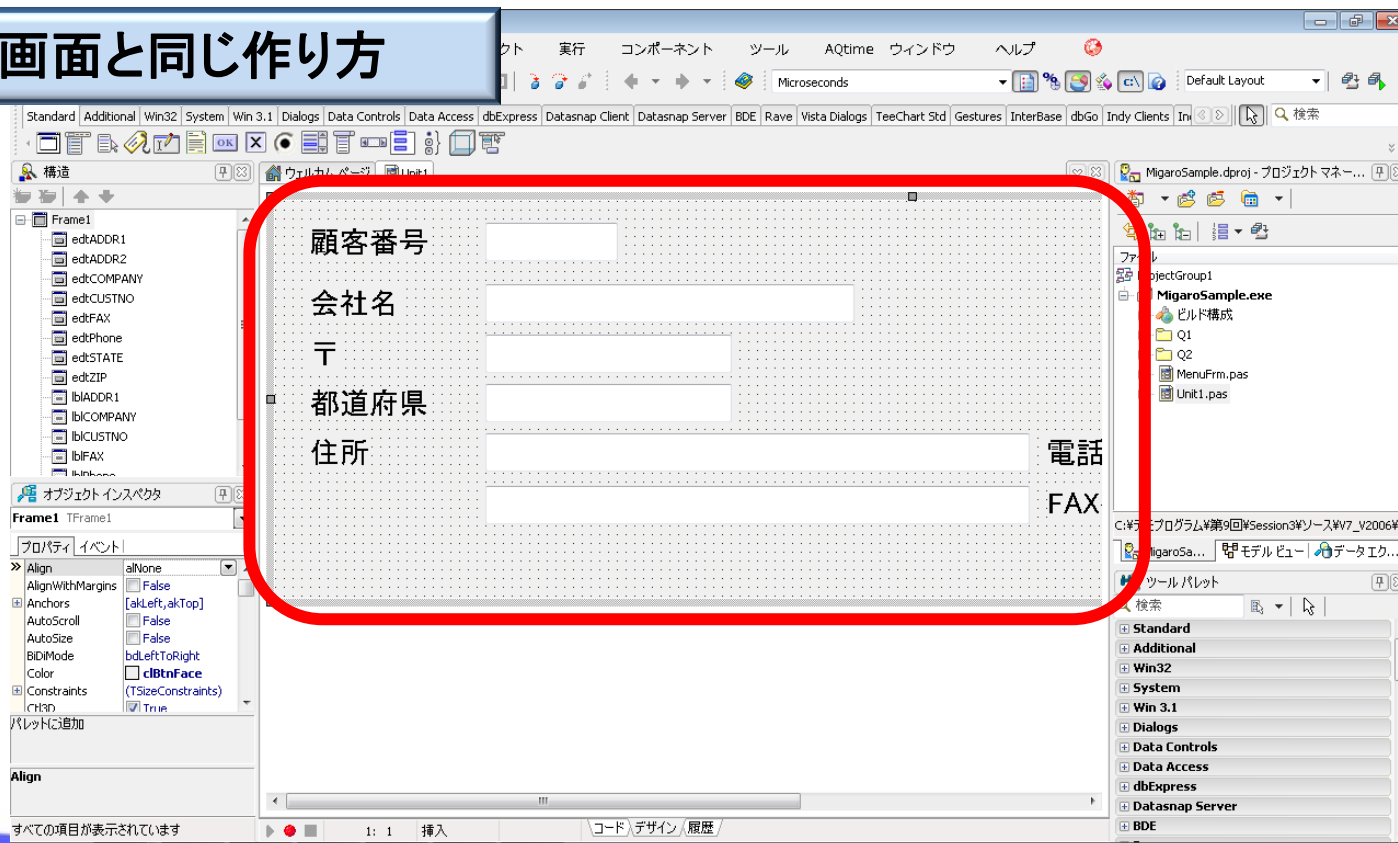
フレーム

■ Q1. PageControl応用テクニック

- フレームとは？

画面(フレーム)にコンポーネントを貼りつけて開発

通常の画面と同じ作り方



■ Q1. PageControl応用テクニック

・ フレームとは？

作ったフレームというコンポーネントとして貼り付けられる

The screenshot shows a software development environment with a tool palette on the left and a form editor on the right. The tool palette has a search bar and a list of components. The 'Frames' component is highlighted with a red box. A yellow arrow points from this box to a form in the editor. The form contains several input fields for customer information. Two callouts with red borders and white backgrounds point to the form, stating '共通化した部品として貼り付け' (Paste as a shared component). A third callout with a black border and white background at the bottom left states '継承と同じような再利用ができる' (Can be reused in a way similar to inheritance).

ツールパレット

検索

Standard

Frames

TMainMenu

TPopupMenu

TLabel

TEdit

TMemo

TButton

顧客番号

会社名

〒

都道府県

住所

顧客番号

会社名

〒

共通化した部品として貼り付け

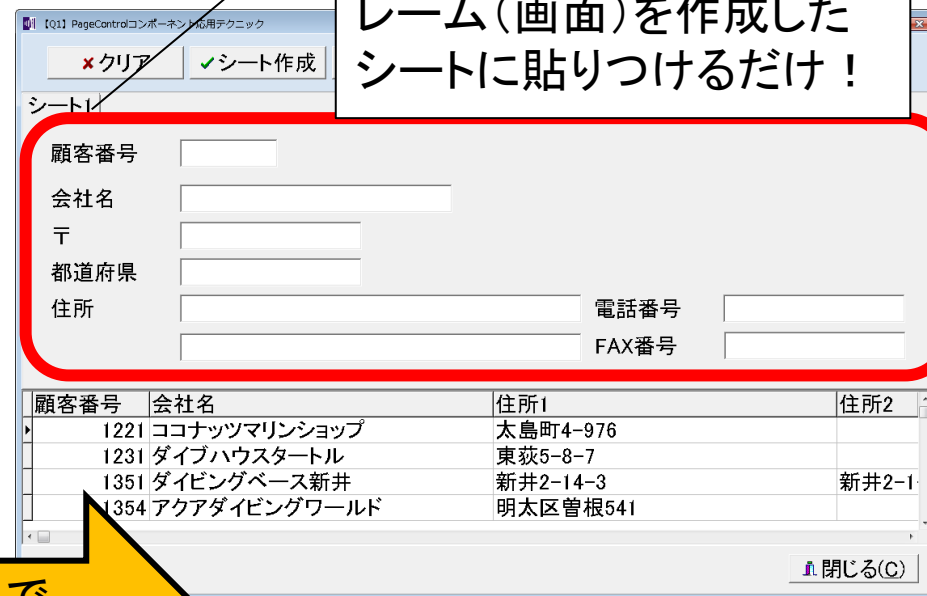
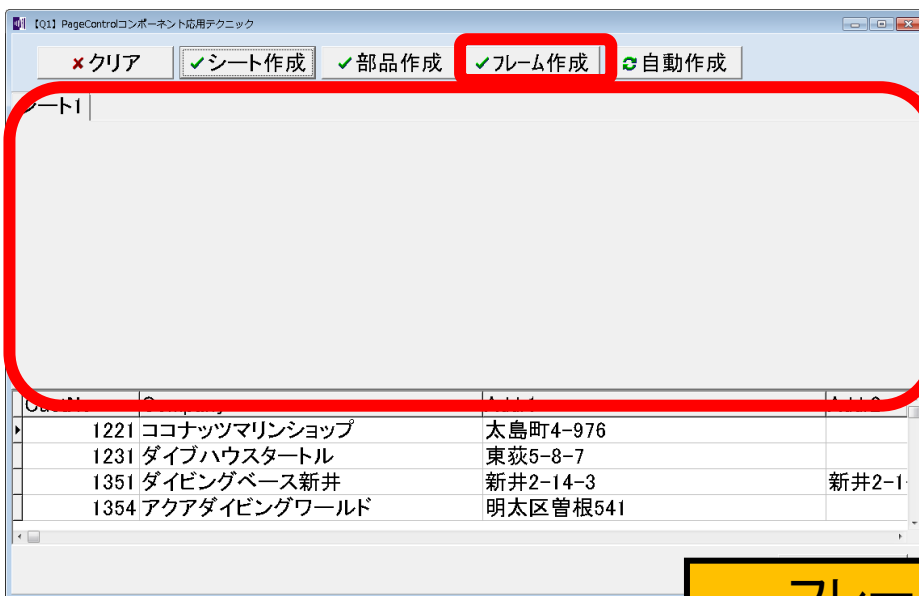
共通化した部品として貼り付け

継承と同じような再利用ができる

■ Q1. PageControl応用テクニック

● ④フレームを利用してシート項目を一括作成する

あらかじめ作成しているフレーム(画面)を作成したシートに貼りつけるだけ!



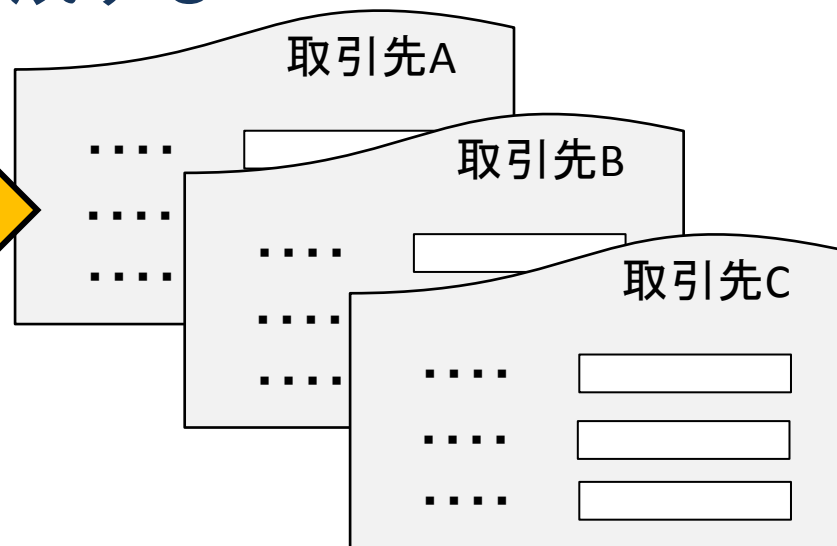
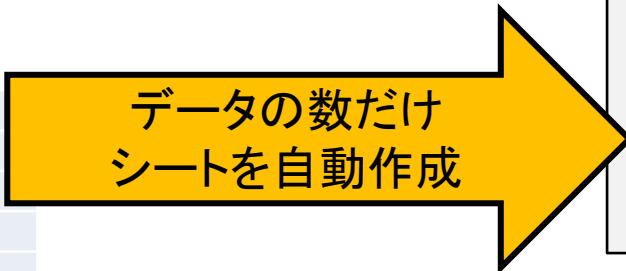
フレームで
シート内容を作成

■ Q1. PageControl応用テクニック

- ⑤ データから動的にシートを作成する
たとえば..

取引先情報

取引先 A	...
取引先 B	...
取引先 C	...
取引先 D	...
...	



いろいろな用途

支店情報

支店1	...
支店2	...
支店3	...
支店4	...
...	

社員情報

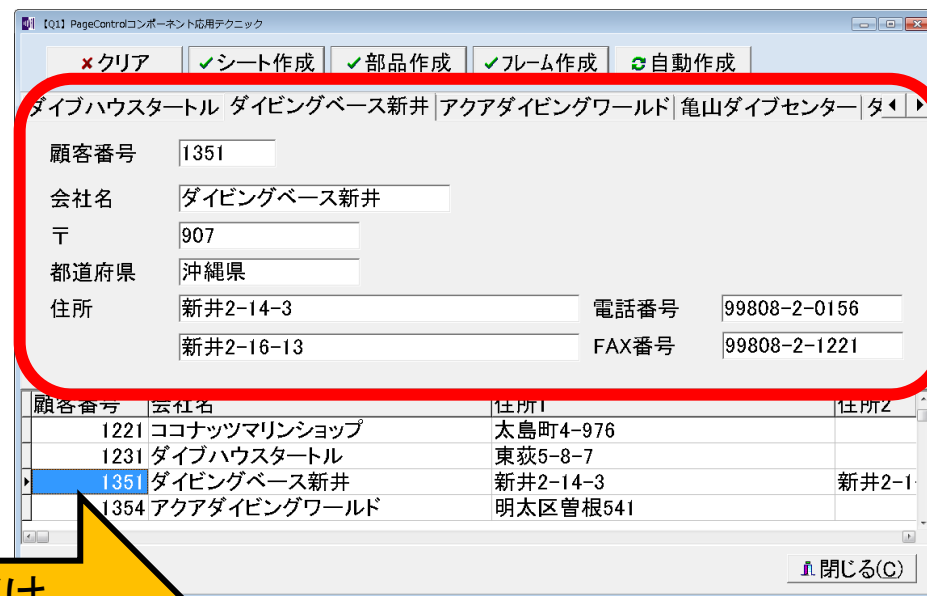
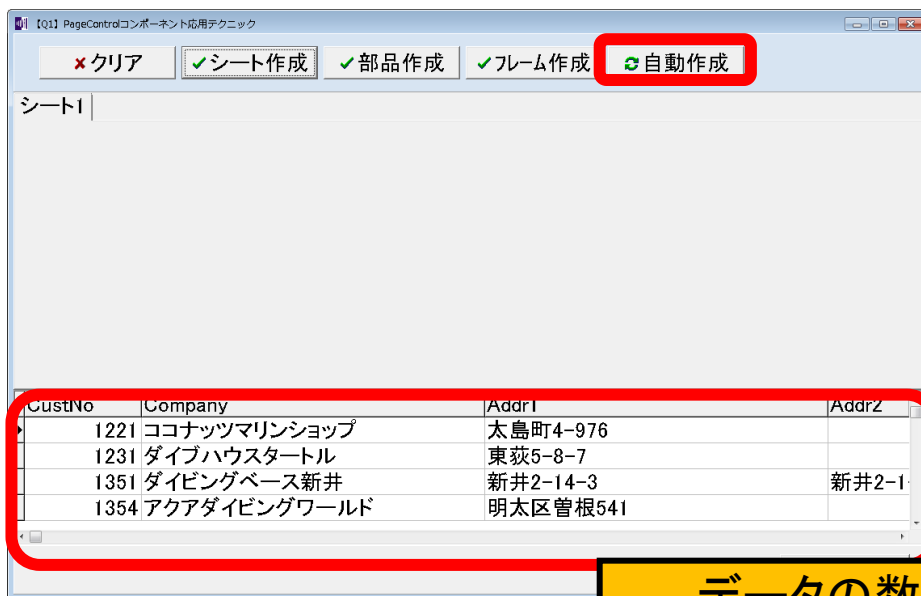
社員 A	...
社員 B	...
社員 C	...
社員 D	...
...	

部品構成情報

部品1	...
部品2	...
部品3	...
部品4	...
...	

■ Q1. PageControl応用テクニック

- ⑤データから動的にシートを作成する
ここまでのプログラムを応用



データの数だけ
シートを自動作成

■ Q1. PageControl応用テクニック

自動作成ボタンの処理(ソース)

```
procedure TfrmQ1.btnAutoClick(Sender: TObject);
var
  TabSheet:TTabSheet; //シート追加用
  Frame:TFrame1;      //フレーム追加用
begin
  with SQLQuery1 do
  begin
    //データの最初のレコードへ移動
    First;
    //データがなくなるまでシートを追加
    while not(EOF) do
    begin
      //シートの生成
      TabSheet := TTabSheet.Create(PageControl1);
      //PageControlに追加
      TabSheet.PageControl := PageControl1;
      //シート名に会社名を設定
      TabSheet.Caption := FieldByName('Company').AsString;
```

データの数だけループして
①シートの作成
②フレームの作成
③データのセット
を行う。

①シートの作成

■ Q1. PageControl応用テクニック

自動作成ボタンの処理(ソース)

```
//フレームの生成
Frame := TFrame1.Create(TabSheet);
//フレームをシートにセット
Frame.Parent := TabSheet;
//データをフレームにセット
Frame.edtCUSTNO.Text := FieldByName('CustNo').AsString; //顧客番号
Frame.edtCOMPANY.Text := FieldByName('Company').AsString; //会社名
Frame.edtZIP.Text := FieldByName('Zip').AsString; //郵便番号
Frame.edtSTATE.Text := FieldByName('State').AsString; //都道府県
Frame.edtADDR1.Text := FieldByName('Addr1').AsString; //住所1
Frame.edtADDR2.Text := FieldByName('Addr2').AsString; //住所2
Frame.edtPHONE.Text := FieldByName('Phone').AsString; //電話番号
Frame.edtFAX.Text := FieldByName('FAX').AsString; //FAX
//次のデータへ
Next;
end;
end;
end;
```

②フレームの作成

③データのセット

ポイント:
フレームの部品は
「フレーム.部品」で処理する

フレームを複数用意しておけば、
データ毎のシート画面を作り分けることも可能

■ Q1. PageControl応用テクニック

補足

シートをドラッグ&ドロップで入れ替える

受注情報 | 取引先 | 担当者 | 在庫状況

受注番号 MG0020

受注日 2011/11/22

受注品番 SU02-FG

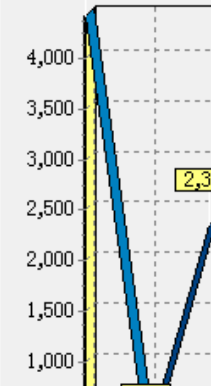
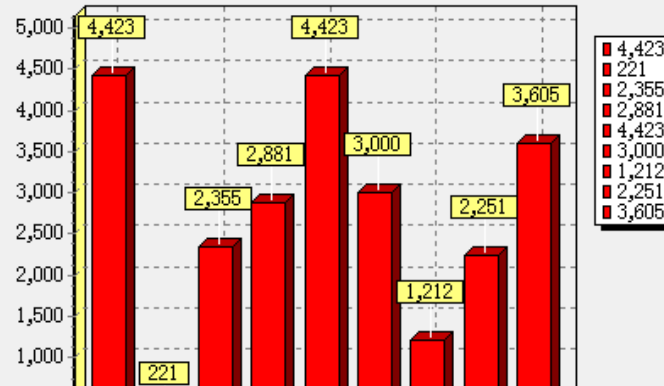
取引先 0

シートのタイトルを
ドラッグ&ドロップ

在庫状況 | 受注情報 | 取引先 | 担当者

SU02-FG

月	在庫
4月	4,423
5月	221
6月	2,355
7月	2,881



■ Q1. PageControl応用テクニック

MouseDown イベント処理(ソース)

```
procedure TfrmQ1.PageControl1MouseDown(Sender: TObject;  
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);  
begin  
  //ドラッグ処理を開始する  
  PageControl1.BeginDrag(false);  
end;
```

DragOver イベント処理(ソース)

```
procedure TfrmQ1.PageControl1DragOver(Sender, Source: TObject; X,  
  Y: Integer; State: TDragState; var Accept: Boolean);  
begin  
  //ドラッグ元がPageControlの時だけ処理する  
  if Sender is TPageControl then Accept := true;  
end;
```

■ Q1. PageControl応用テクニック

DragDropイベント処理(ソース)

```
procedure TfrmQ1.PageControl1DragDrop(Sender, Source: TObject; X,  
  Y: Integer);  
var  
  i: Integer;  
  r: TRect;  
begin  
  if not (Sender is TPageControl) then Exit; //ドラッグ元がPageControlの時だけ処理する  
  with PageControl1 do  
  begin  
    for i := 0 to PageCount - 1 do //ドロップ先のシート番号を特定して移動する  
    begin  
      Perform(TCM_GETITEMRECT, i, LPARAM(@r));  
      if PtInRect(r, Point(X, Y)) then  
      begin  
        if i <> ActivePage.PageIndex then  
        begin  
          ActivePage.PageIndex := i;  
        end;  
        Exit;  
      end;  
    end;  
  end;  
end;  
end;  
end;
```

■ Q2.DLLモジュールの開発手法

❗【質問】

Delphi/400ではEXEアプリケーション以外に、DLLも開発できると聞きます。どんな点で便利なのですか？

✉【回答】

DLLは、他から呼び出される(利用される)ことを前提とした汎用的な共通モジュールになります。
EXEと違い、呼出されたら起動するだけでなく、結果も呼び出し元に自由に返せる点、他言語からも利用できる点で非常に便利な使い方できます。ただしEXEと違い、DLL単体では実行できません。

■ Q2.DLLモジュールの開発手法

• DLLとは？



Dynamic Link Library(ダイナミックリンクライブラリ)

DLLとは、動的なリンクによって利用されるライブラリのことである。
Windowsでは、DLLファイルの拡張子として「.dll」が付く。

様々なプログラムから呼び出されるための、汎用的な機能が
モジュール化されており、EXEなどの実行ファイルがリンクを
読み込むことによって共通して利用できるようになっている。

IT用語辞典抜粋

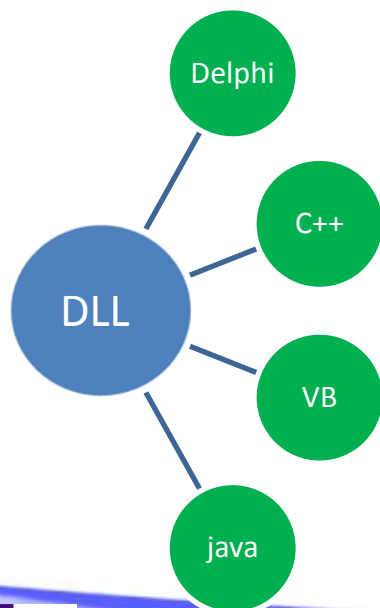
■ Q2.DLLモジュールの開発手法

• DLLとは？



つまり..

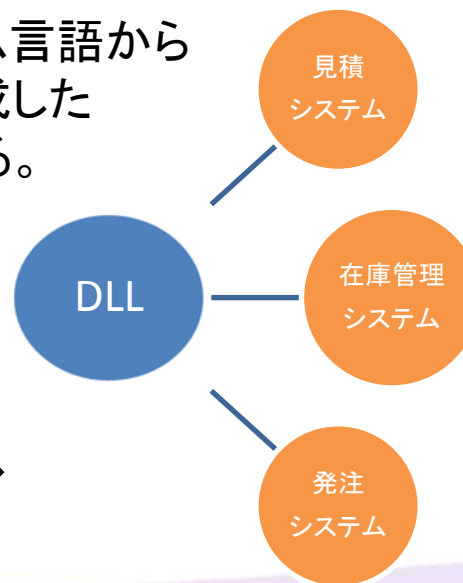
いろいろなプログラム言語、アプリケーションで
利用できる便利な共通プログラム



様々なプログラム言語から
Delphi/400で作成した
機能を利用できる。

ex.)
ログオン認証機能、
名称取得機能...

複数システム
の共通化



様々なアプリケーションから
共通機能として利用できる。
機能変更時もDLLを入替える
だけなので、運用も◎

ex.)
社員検索画面、取引先検索画面..

重複開発が不要
EXEのスリム化

■ Q2.DLLモジュールの開発手法

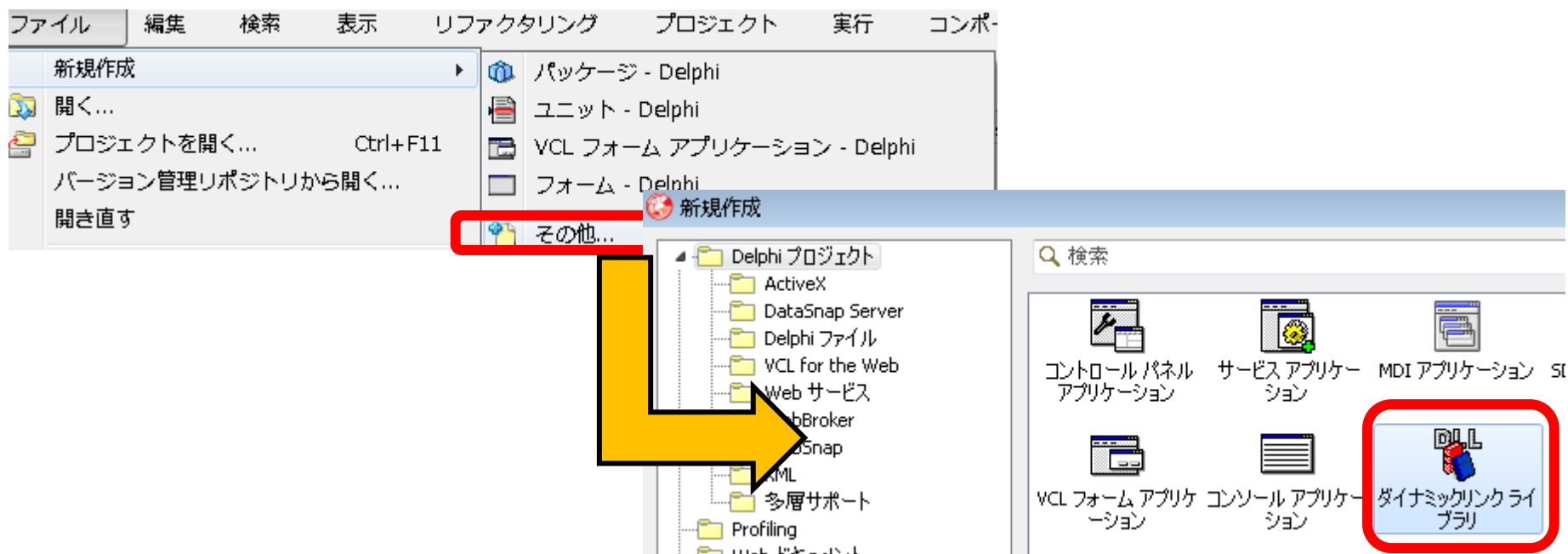
DLLモジュールの開発手法

- ①DLLの基本開発手順
- ②Delphi/400コンポーネントを組み込んだDLLの開発
- ③画面を含むDLLの開発

■ Q2.DLLモジュールの開発手法

• ① DLLの基本開発手順

ファイル>新規作成>その他



■ Q2.DLLモジュールの開発手法

• ① DLLの基本開発手順

DLL用のプロジェクトが作成される

```
Project1
├── library Project1;
│   ├── { DLL のメモリ管理に関する重要なメモ: パラメータまたは関数結果として文字列を渡す
│   │   手続きまたは関数を、DLL がエクスポートする場合は、ShareMem をライブラリの
│   │   uses 句およびプロジェクトの uses 句 ([プロジェクト|ソース表示] を選択) の
│   │   最初に記載する必要があります。これは、
│   │   DLL との間で渡されるすべての文字列に当てはまります。レコードやクラスに
│   │   ネストされているものも同様です。ShareMem は共有メモリ マネージャ BORLNDMM.DLL に対する
│   │   ユニットです。この DLL は作成対象の DLL と一緒に配置する必要があります。
│   │   BORLNDMM.DLL を使用しないようにするには、PChar 型または ShortString 型の
│   │   パラメータを使って文字列情報を渡します。}
│   ├── uses
│   │   SysUtils,
│   │   Classes;
│   ├── {$R *.res}
│   └── begin
│       Lend.
└──
```

Project1.dproj - プロジェクト

ファイル

- Project1group1
- Project1.dll**
- ビルド構成

DLLとして作成される

■ Q2.DLLモジュールの開発手法

• ① DLLの基本開発手順

DLLプログラム例(ソース)

```
Library Project1;
```

```
uses  
  SysUtils,  
  Classes;
```

```
//計算関数 (パラメータを足し算して結果を返却)  
function Keisan(a, b: Integer): Integer; stdcall;  
begin  
  Result := a + b;  
end;
```

```
{$R * res}  
exports  
  Keisan;
```

```
begin  
end.
```

宣言
ポイント:
外部から呼び出したい関数、
手続きはexportsの下に全て
宣言する。

関数、手続き(プログラム)
ポイント:

- ①stdcallをつける。
- ②StringなどのDelphi独自の型を使用しない。
※他言語でも使われる一般的なPCharやIntegerなどの汎用的な型を使用する。

■ Q2.DLLモジュールの開発手法

• ① DLLの基本開発手順

コンパイル・・・完成

Win32

名前	更新日時	サイズ	種類
 Project1.dll	2011/11/03 19:27	392 KB	アプリケーション拡張

■ Q2.DLLモジュールの開発手法

• 作成したDLLの呼出 (Delphi/400)

EXE側 呼び出しプログラム例(ソース)

```
function Keisan(a, b: Integer): Integer; stdcall; external 'project1.dll';
```

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
  SpinEdit3.Value := Keisan(SpinEdit1.Value, SpinEdit2.Value);  
end;
```

宣言
ポイント:
external 'DLL名(パス含む)'
を宣言する。



実行(呼び出し)
ポイント:
通常の関数と同じように使えます。

 Project1.dll

■ Q2.DLLモジュールの開発手法

• 作成したDLLの呼出(VBA)

EXE側 呼び出しプログラム例(ソース)

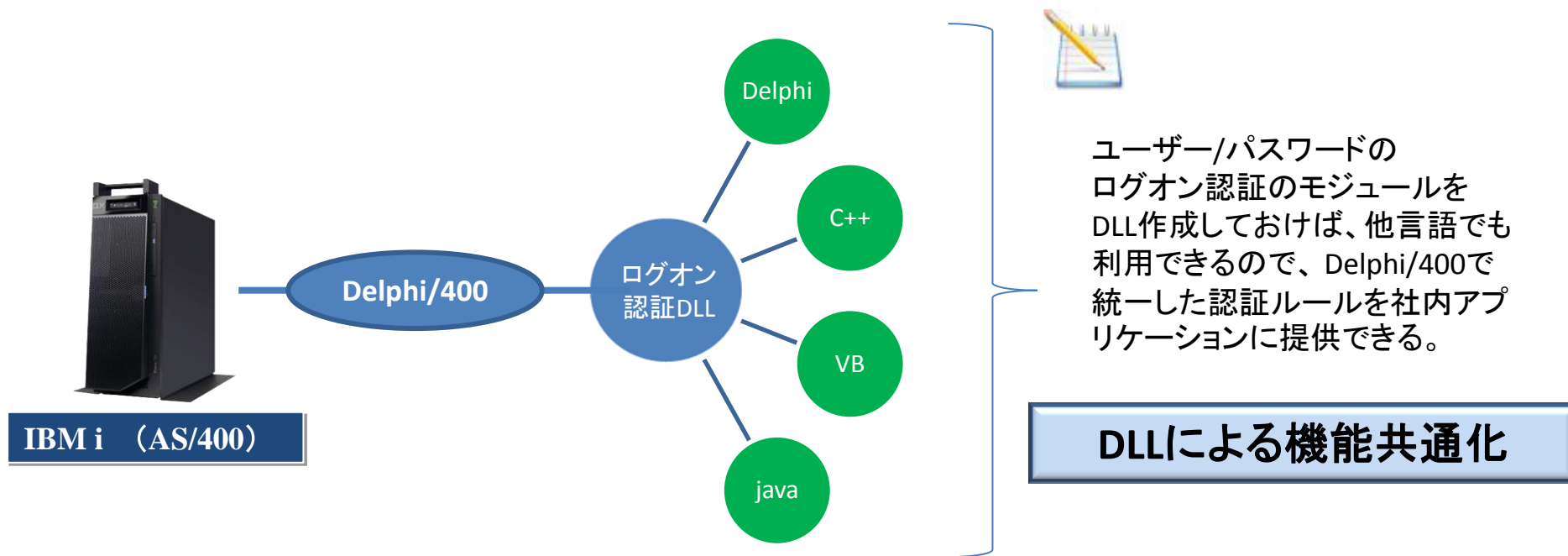
```
Declare Function Keisan Lib "Project1.dll" (ByVal a As Long, ByVal b As Long) As Long  
  
Sub ボタン1_Click()  
    Range("H7") = Keisan(Range("D7"), Range("F7"))  
End Sub
```

入力項目1	入力項目2	計算結果
1,000	2,500	3,500

Project1.dll

■ Q2.DLLモジュールの開発手法

- ② Delphi/400コンポーネントを組み込んだDLLの開発
作成するDLL: IBMi(AS/400)ログオン認証モジュール



■ Q2.DLLモジュールの開発手法

ログオン認証DLLプログラム例(ソース)

```
library Project2;
```

```
uses
```

```
  SysUtils,  
  Classes,  
  Scdconn;
```

```
{$R *.res}
```

```
//ログオン認証 (パラメータの認証結果を返却)
```

```
function Login(usrid, password: Pchar): Boolean; stdcall;
```

```
var
```

```
  AS400: TAS400;
```

```
begin
```

```
  AS400 := TAS400.Create(nil); //画面はないので、コンポーネントは生成する。
```

```
  try
```

```
    AS400.Userid := usrid; //パラメータをセット
```

```
    AS400.PWD := password; //パラメータをセット
```

宣言

ポイント:

AS400コンポーネントを使う場合はUsesにScdconnを追加する。

※CALL400を使う場合はScdcallも追加。

ポイント:

画面がない場合はコンポーネントはプログラムで生成する。

■ Q2.DLLモジュールの開発手法

ログオン認証DLLプログラム例(ソース)

```
try
  AS400.Connect;      //接続
except
  Result := False;   //失敗(例外)の場合はFalseを返却
  Exit;
end;
AS400.Disconnect;   //成功したら接続を切ってTrueを返却
Result := True;
finally
  FreeAndNil(AS400); //生成したコンポーネントは破棄
end;
end;

exports
Login;

begin
end.
```

認証

ポイント:

ここでは、ログオン認証を接続だけで行っているが、接続した上で、社員マスタなどで独自に認証する仕組みを構築するのも◎

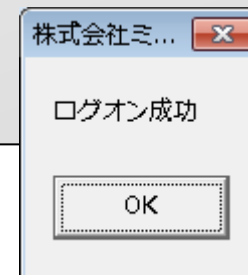
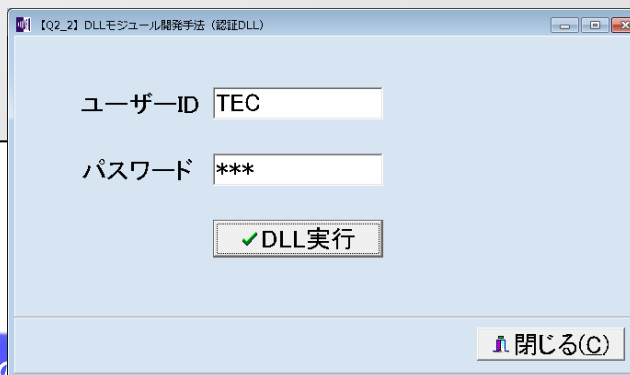
■ Q2.DLLモジュールの開発手法

• 作成したDLLの呼出 (Delphi/400)

EXE側 呼び出しプログラム例(ソース)

```
function Login(usrid, password: Pchar): Boolean; stdcall; external 'project2.dll';  
  
procedure TfrmQ2_2.BitBtn1Click(Sender: TObject);  
begin  
  if (Login(PChar(Edit1.Text), PChar(Edit2.Text))) then  
  begin  
    ShowMessage('ログオン成功');  
  end  
  else  
  begin  
    ShowMessage('ログオン失敗');  
  end;  
end;  
end;
```

実行(呼び出し)
ポイント:
DLLの型に合わせてパラメータを渡す。



■ Q2.DLLモジュールの開発手法

• 作成したDLLの呼出(VBA)

EXE側 呼び出しプログラム例(ソース)

```
Declare Function Login Lib "Project2.dll" (ByVal UID As String, ByVal PWD As String) As Boolean
```

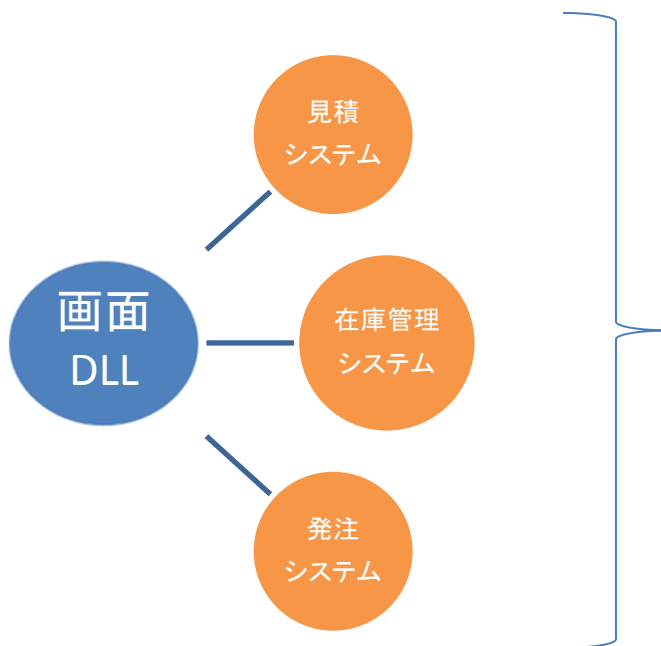
```
Sub ボタン2_Click()  
  If Login(Range("D12"), Range("F12")) Then  
    Range("H12") = "ログオン成功"  
  Else  
    Range("H12") = "ログオン失敗"  
  End If  
End Sub
```

A	B	C	D	E	F	G	H
Q2.DLLモジュール開発手法と活用							
②Delphi/400コンポーネントを組み込んだ開発							
			ユーザーID		パスワード		認証結果
	認証DLL実行		TEC		***		ログオン成功

■ Q2.DLLモジュールの開発手法

● ③画面を含むDLLの開発

作成するDLL: 取引先選択画面を起動して選択データを返す



取引先検索画面や社員検索画面など複数のアプリケーションで同じ機能が必要する場合は多い。

それぞれのアプリケーションに同じ画面(機能)を組み込むよりも共通モジュールとしてDLLを作成しておけば、社内アプリケーションを共通化およびスリム化することができる。

また共通機能変更時にDLLの置換えだけで済むので、運用も楽になる。

同じ画面を各EXEに含まないで済む

機能変更時にDLL置換えだけで済む

EXEの分割より、PGM連携が簡単

■ Q2.DLLモジュールの開発手法

● ③画面を含むDLLの開発

作成するDLL: 取引先選択画面を起動して選択データを返す

顧客番号 ...

会社名

〒

都道府県

住所

電話番号

FAX番号

閉じる(C)

DLLモジュール

取引先選択

顧客番号	会社名	郵便番号	都道府県	住所1
1221	ココナツマリンショップ	100-01	東京都	太島町4-976
1231	ダイブハウスタートル	166	東京都	東荻5-8-7
1351	ダイビングベース新井	907	沖縄県	新井2-14-3
1354	アクアダイビングワールド	808	福岡県	明太区菅根541
1356	亀山ダイブセンター	263	千葉県	稲毛区亀山町632-1
1380	ダイブショップブルーリーフ	105	東京都	鯖松町23-738
1384	MHMダイバーズクラブ	271	千葉県	埴輪町32
1510	オーシャンパラダイスサービス	100-04	東京都	兄島村745
1513	Fantastique Aquatica	96950		Avenue F Garapan
1551	クアトロスポーツクラブ		北海道	北区北5条東666丁目
1560	いるか村	430	静岡県	鰻林521-33
1563	パブルスポーツ	900	沖縄県	昆布西203
1624	上牛ダイビングクラブ	154	東京都	上牛8534

✓ 選択 ✕ キャンセル

■ Q2.DLLモジュールの開発手法

- ③画面を含むDLLの開発
DLLで画面を作成する手順

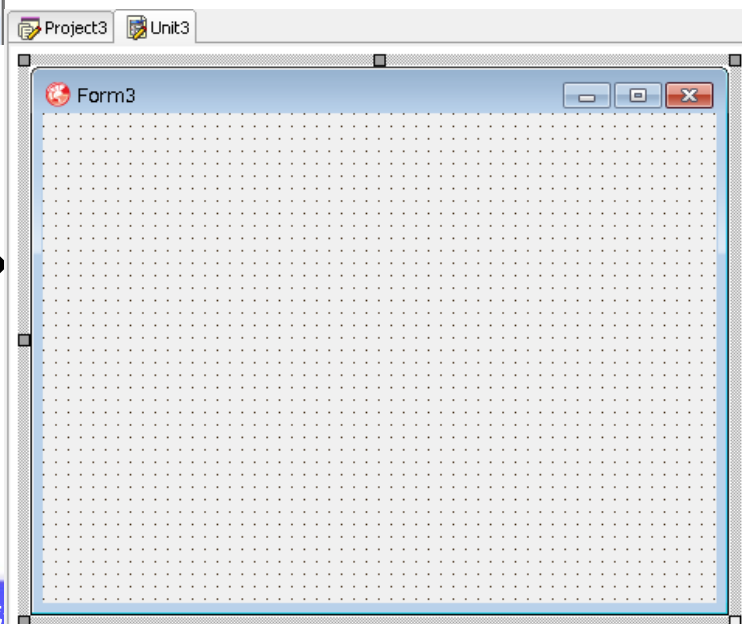
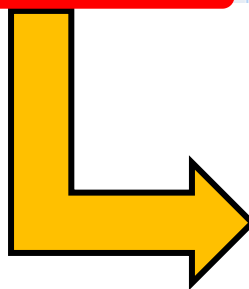
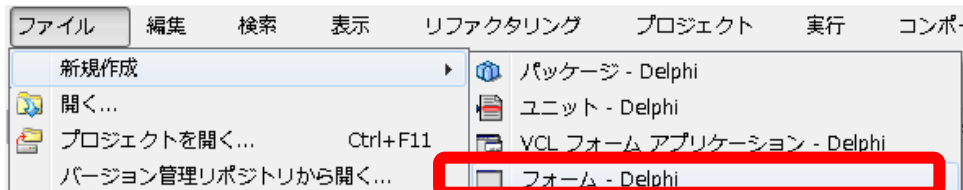
DLL用プロジェクトを作成

```
1 library Project3;
2
3 { DLL のメモリ管理に関する重要なメモ: パラメータまたは関数結果として
4   手続きまたは関数を DLL がエクスポートする場合は、ShareMem をライブ
5   uses 句およびプロジェクトの uses 句 ([プロジェクト|ソース表示] を
6   最初に記載する必要があります。これは、
7   DLL との間で渡されるすべての文字列に当てはまります。レコードやクラ
8   ネストされているものも同様です。ShareMem は共有メモリ マネージャ B
9   ユニットです。この DLL は作成対象の DLL と一緒に配置する必要があります。
10  BORLNDMM.DLL を使用しないようにするには、PChar 型または
11  パラメータを使って文字列情報を渡します。}
12
13 uses
14   SysUtils,
15   Classes;
16
17 {$R *.res}
18
19 begin
20 end.
```

■ Q2.DLLモジュールの開発手法

- ③画面を含むDLLの開発
DLLで画面を作成する手順

ファイル>新規作成>フォーム - Delphi



■ Q2.DLLモジュールの開発手法

● ③画面を含むDLLの開発

DLLで画面を作成する手順 画面設計とプログラム(通常フォームと同様)

取引先選択画面DLLプログラム例(ソース)・・・詳細部分は割愛

```
public
{ Public 宣言 }
pCUSTNO : Integer; //顧客番号 (受渡用)
pCOMPANY : String; //会社名 (受渡用)
end;
. . .
//OKボタンクリック
procedure TfrmTRCD.btnOKClick(Sender: TObject);
begin
//DBGridで選択された値を
with SQLQuery1 do
begin
pCUSTNO := FieldByName('CUSTNO').AsInteger;
pCOMPANY := FieldByName('COMPANY').AsString;
end;
end;
```

顧客番号	会社名	郵便番号	都道府県	住所
------	-----	------	------	----

■ Q2.DLLモジュールの開発手法

DLLで画面を作成する手順

画面設計とプログラム(通常フォームと同様)

DLL本体で画面を呼び出すプログラム例(ソース)

```
library Project3;
```

```
uses  
  SysUtils,  
  Classes,  
  TRCDfrm in 'TRCDfrm.pas' {frmTRCD};
```

```
function ShowForm(var CustNo:Integer; var Company:PChar): Integer; stdcall; export;
```

```
var  
  Form : TfrmTRCD; //取引先選択画面用
```

```
begin
```

```
  Form := TfrmTRCD.Create(nil); //取引先選択画面生成  
  Result := Form.ShowModal; //画面起動(処理待ち)  
  CustNo := Form.pCUSTNO; //顧客番号返却  
  Company := PChar(Form.pCOMPANY); //会社名返却  
  Form.Release; //破棄
```

```
end;
```

宣言

ポイント:

画面と値をやりとりする場合はパラメータを用意しておく。

画面の起動

ポイント:

DLLのExports関数の中で対象の画面(フォーム)を生成起動する。

■ Q2.DLLモジュールの開発手法

DLLで画面を作成する手順

画面設計とプログラム(通常フォームと同様)

取引先選択画面DLLプログラム例(ソース) DLL本体

```
{ $R *.res }  
exports  
  ShowForm;  
begin  
end.
```

DLLの関数(ShowForm)
を呼出すと起動する!

顧客番号	会社名	郵便番号	都道府県	住所1
1221	ココナッツマリンショップ	100-01	東京都	太島町4-976
1231	ダイブハウスタートル	166	東京都	東荻5-8-7
1351	ダイビングベース新井	907	沖縄県	新井2-14-3
1354	アクアダイビングワールド	808	福岡県	明太区曾根541
1356	亀山ダイブセンター	263	千葉県	稲毛区亀山町632-1
1380	ダイブショップブルーリーフ	105	東京都	鯖松町23-738
1384	MHMダイバーズクラブ	271	千葉県	埴輪町32
1510	オーシャンパラダイスサービス	100-04	東京都	兄島村745
1513	Fantastique Aquatica	96950		Avenue F Garapan
1551	クアトロスポーツクラブ		北海道	北区北5条東666丁目
1560	いるか村	430	静岡県	鰐林521-33
1563	バブルスポーツ	900	沖縄県	昆布西203
1624	上牛ダイビングクラブ	154	東京都	上牛8534

✓ 選択 ✕ キャンセル

■ Q2.DLLモジュールの開発手法

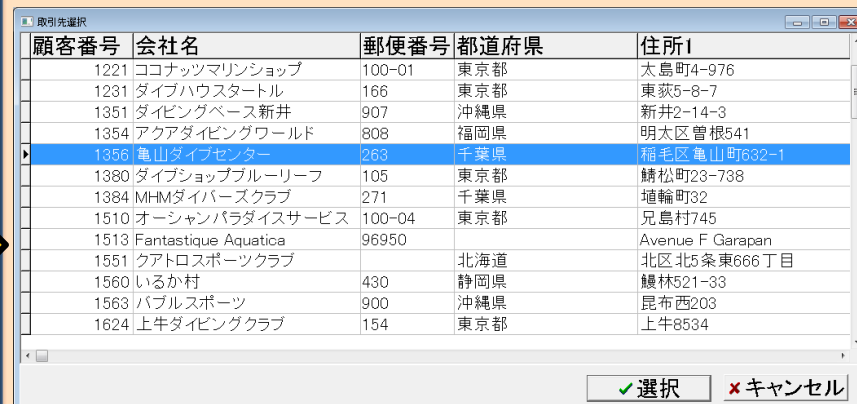
- 作成したDLLの呼出 (Delphi/400)

EXE側 呼び出しプログラム例(ソース)

```
function ShowForm(var CustNo:Integer; var Company:PChar): Integer; stdcall; external 'project3.dll';  
  
//顧客番号の補助ボタンクリック  
procedure TfrmQ2_3.btnDLLClick(Sender: TObject);  
var  
    iCustNo : Integer; //顧客番号 (DLL呼び出し用)  
    pcCompany: PChar; //会社名 (DLL呼び出し用)  
begin  
    //DLLで取引先選択画面を起動  
    ShowForm(iCustNo, pcCompany);  
    //DLL返却の顧客番号をセット  
    Frame1.edtCUSTNO.Text := IntToStr(iCustNo);  
    //DLL返却の会社名をセット  
    Frame1.edtCOMPANY.Text := pcCompany;  
end;
```

値の連携

DLLモジュール(画面)



顧客番号	会社名	郵便番号	都道府県	住所1
1221	ココナッツマリンショップ	100-01	東京都	太島町4-976
1231	ダイブハウススタートル	166	東京都	東荻5-8-7
1351	ダイビングベース新井	907	沖縄県	新井2-14-3
1354	アクアダイビングワールド	808	福岡県	明太区曾根541
1356	亀山ダイブセンター	263	千葉県	稲毛区亀山町632-1
1380	ダイブショップブルーリーフ	105	東京都	鯖松町23-738
1384	MHMダイバーズクラブ	271	千葉県	埴輪町32
1510	オーシャンパラダイスサービス	100-04	東京都	兄島村745
1513	Fantastique Aquatica	96950		Avenue F Garapan
1551	クアトロスポーツクラブ		北海道	北区北5条東666丁目
1560	いるか村	430	静岡県	鯉林521-33
1563	パプルススポーツ	900	沖縄県	昆布西203
1624	上牛ダイビングクラブ	154	東京都	上牛8534

ポイント:
EXE連携と違って、
簡単に選択値を受け取れる。

■ Q2.DLLモジュールの開発手法

- 作成したDLLの呼出(VBA)

EXE側 呼び出しプログラム例(ソース)

```
Declare Function ShowForm Lib "Project3.dll" (ByRef CustNo As Long, ByRef Company As String) As Long
```

```
Sub ボタン3_Click()  
Dim CustNo As Long  
Dim Company As String
```

```
If ShowForm(CustNo, Company) = 1 Then
```

```
Range("D17") = CustNo  
Range("F17") = Company
```

```
End If
```

```
End Sub
```



DLLモジュール(画面)

顧客番号	会社名	郵便番号	都道府県	住所1
1221	ココナッツマリンショップ	100-01	東京都	太島町4-976
1231	ダイブハウススタートル	166	東京都	東荻5-8-7
1351	ダイビングベース新井	907	沖縄県	新井2-14-3
1354	アクアダイビングワールド	808	福岡県	明太区曾根541
1356	亀山ダイブセンター	263	千葉県	稲毛区亀山町632-1
1380	ダイブショップブルーリーフ	105	東京都	鯖松町23-738
1384	MHMダイバースクラブ	271	千葉県	埴輪町32
1510	オーシャンパラダイスサービス	100-04	東京都	兄島村745
1513	Fantastique Aquatica	96950		Avenue F Garapan
1551	クアトロスポーツクラブ		北海道	北区北5条東666丁目
1560	いるか村	430	静岡県	鯉林521-33
1563	バブルスポーツ	900	沖縄県	昆布西203
1624	上牛ダイビングクラブ	154	東京都	上牛8534

✓ 選択 ✗ キャンセル

■ Q2.DLLモジュールの開発手法

● 作成したDLLの呼出 (VBA)

Microsoft Excel - DLLTest.xls

ファイル(F) 編集(E) 表示(V) 挿入(I) 書式(O) ツール(T) データ(D) ウィンドウ(W) ヘルプ(H)

MS Pゴシック 11 B I U

F14 =

	A	B	C	D	E	F	G	H
1		Q2.DLLモジュール開発手法と活用						
2		①DLLの基本開発手順						
3								
4				入力項目1		入力項目2		計算結果
5		計算DLL実行		1,200	+	2,400	=	3,600
6								
7								
8								
9		②Delphi/400コンポーネントを組み込んだ開発						
10								
11				ユーザーID		パスワード		認証結果
12		認証DLL実行		TEC		***		ログオン成功
13								
14		③画面を含むDLLの開発						
15								
16				顧客番号		会社名		
17		画面DLL実行		1351		ダイビングベース新井		
18								
19								

顧客番号	会社名	郵便番号	都道府県	住所1
1221	ココナツマリンショップ	100-01	東京都	太島町4-976
1231	ダイブハウススタート	166	東京都	東荻5-8-7
1351	ダイビングベース新井	907	沖縄県	新井2-14-3
1354	アクアダイビングワールド	808	福岡県	明太区曽根541
1356	奥山ダイビングセンター	263	千葉県	稲毛区奥山町632-1
1380	ダイブショップブルーリーフ	105	東京都	鶴松町23-738
1384	MHMダイバースクラブ	271	千葉県	稲輪町32
1510	オーシャンパラダイスサービス	100-04	東京都	兄島村745
1513	Fantastique Aquatica	96950		Avenue F Garapan
1551	クアトロスポーツクラブ		北海道	北区北森東666丁目
1560	いるか村	430	静岡県	駿林521-33
1563	パブルスポーツ	900	沖縄県	昆布西203
1624	上半ダイビングクラブ	154	東京都	上午8534

選択 キャンセル

Delphi/400以外のアプリケーションにもDelphi/400が活用できる!

ご清聴ありがとうございました。