MIGARO. TECHNICAL REPORT

ミガロ.テクニカルレポート

No.6
2013年秋

株式会社ミガロ.

CONTENTS 目次

ごあいさつ

いつもミガロ.製品をご愛用いただき誠にありがとうございます。

さて「ミガロ.製品をご利用中の技術者の皆様に、日々の開発に少しでもお役に立つような技術情報をご提供したい」という思いから2008年に創刊した『Migaro.Technical Report』は、このたび第6号を無事に発刊する運びとなりました。これもひとえに、ご多忙中にもかかわらず『Migaro.Technical Award（お客様論文)』にご寄稿いただいた多くのお客様、ならびに『Migaro.Technical Report』に対して貴重なご意見・ご要望をお寄せ下さった皆様のご支援の賜物と、心より感謝をしております。

本Reportの創刊もつい昨日のことのように思い出されますが、2008年はiPhoneが日本で発売開始、またAndroid搭載スマートフォンが世界で発売開始になった年でもあり、昨今のスマートデバイスの隆盛を見るにつけ、着実に変化している時代に合わせて弊社も変化していかなければならないとの決意を新たにしております。

今回も従来と同様に、第1部は「Migaro.Technical Award 2013 お客様受賞論文」、第2部は「ミガロ.SE論文」の2部構成としています。第1部の「Migaro.Technical Award」とは、日々アプリケーションの開発・保守に携わるエンジニアの方々の努力と創意工夫の成果を顕彰することを目的とし、「Delphi/400」「JC/400」「Business4Mobile」などの弊社製品をご利用中のユーザー様を対象に実践レポート（論文）を公募して厳正な審査・選考のうえ表彰する制度です。昨年に引き続き、従来のお客様論文にあたる「部門1」と「業務課題を解決した開発技術・テクニック」を簡潔にまとめていただく「部門2」の2部門構成といたしました。

今回のお客様論文は、「Delphi/400の開発効率や保守性を向上させる専用フレームワークの構築」や「Delphi/400画面とCTIの連携によりコールセンター業務を改善した事例」など、創意工夫にあふれる論文を多数ご寄稿いただきました。

第2部「ミガロ.SE論文」では、弊社SEによる技術論文を掲載しております。今回は、「Delphi/400の64bitアプリケーション開発手法」や「Delphi/400 Webコンポーネントのカスタマイズ手法」など、さまざまな応用テクニックを開発に活かしていただくための技術情報をご紹介しております。

本レポートが少しでも皆様の開発・保守のお役に立てば幸いです。

最後に『Migaro.Technical Report』第6号を発刊するにあたりまして、多くのお客様・パートナー様にご支援、ご協力をいただきましたことを、この場をお借りして、あらためて厚く御礼を申し上げます。

2013年秋

株式会社ミガロ.
代表取締役社長
上甲 將隆

Migaro. Technical Award 2013

お客様受賞論文／ミガロ. テクニカルアワード

最優秀賞

自社用開発フレームワークの構築
ーなぜフレームワークが必要か 高効率な内製実現のために

駒田 純也 様
ユサコ株式会社
システムグループ マネージャー

ユサコ株式会社
http://www.usaco.co.jp/top.html

海外の学術雑誌、書籍の輸入販売を中心に事業を展開している。特に医学、薬学等の自然科学分野に強みを持つ。近年、学術情報媒体の多様化に伴い、電子ブック、データベース、各種ソフトウェアの取り扱いを強化。学術情報を通じ、知的情報の創造、蓄積、共有による社会貢献を目指している。

Delphi/400導入の経緯

ユサコは、海外の医学系学術誌やデータベースの輸入販売などを行っている会社であり、ユーザー企業の立場で、System/38 の時代から IBM のメインフレームを使い続けている。

当時から基幹システムとして販売管理システムを自社開発しており、現行の基幹システムも開発開始から 16 年、運用開始から 14 年の間機能拡張を行いながら使い続けてきた。

とはいえ、現場で扱わなければならない情報量が格段に増え、エミュレーター画面の表示能力にも限界を強く感じ、インターフェースを再構築するためのツールを探していた。

他にも CASE ツール等も検討していたが、開発やユーザー操作の自由度が低く、Delphi/400 であればどんな要望にもフレキシブルに応えられると考えた。

なぜフレームワークが必要か

Delphi/400 での開発は自由度が高く、CASE ツールのように煩雑な事前作業は必要ないが、その反面、開発者ごとにデータベースやコーディング上の命名基準、画面構成や配色、エラー処理といったものがバラバラになってしまう恐れがある。

この問題を回避するには、モデリングやコーディング規則だけでなく、ユーザーに同じ見た目で、同じ操作基準を持った、一貫性のあるシステムを提供する「開発フレームワーク」が必要である。しかし、製品化されているものがなく、内製するに至った。

加えて、フレームワークを使用することにより仕様変更に強く、開発やテスト工数を削減し、バグも減らせるという利点も重要であった。

フレームワーク概要

フレームワークは現行システムに不足している管理用テーブルや、標準コンポーネントを拡張したオリジナルコンポーネント、キャッシュ用エンティティクラス等の共通クラス、入力値チェックや値選択機能、アプリケーション自動配布の仕組みで構成している。

開発に当たっては、コーディング量を減らし、開発効率と保守性を高めるため、オブジェクト指向でいうカプセル化(※)やロジックの汎用化 (汎化) を意識した。【図 1】

また、オブジェクト指向言語でクライアント / サーバー系システムの開発をフレームワーク部分から行うのは初めてであったため、開発支援を受け試行錯誤しながらの構築であった。

※カプセル化とは

オブジェクト内のデータを直接操作できないように、その構造を外部から隠蔽すること。オブジェクト内の仕様変更が

図1　フレームワーク概要

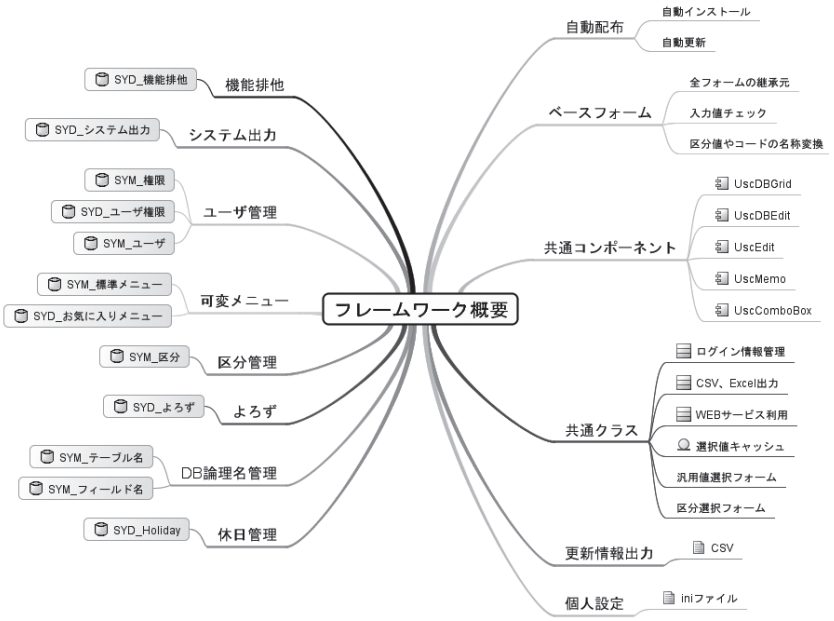
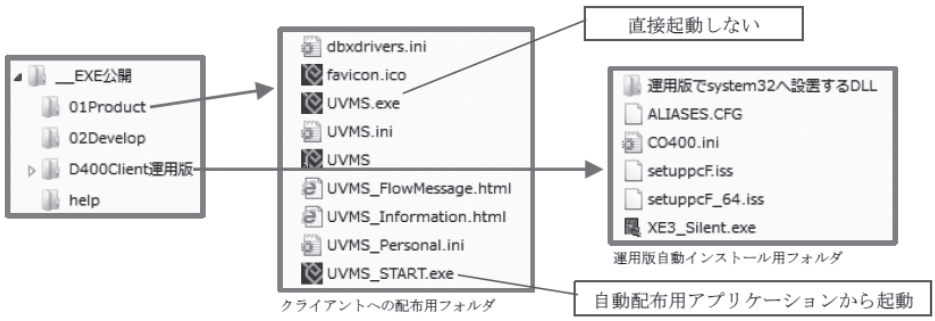


図2　アプリケーション自動配布



ソース1　サンプルコード

```
procedure xxx.Form_OnCreate(Sender: TObject);
var
  i: Integer;
begin
  //DBGrid の列タイトルを設定する
  dbgMain.fSetLogicalName(cdsLgcFld,'VCCIZREP');

  //入力項目のラベルを設定する
  for i := 0 to ComponentCount - 1 do begin
    if Components[i] is TLabel then begin
      if cdsLgcFld.Locate('TblPhyName;FldPhyName',
        VarArrayOf(['VCCIZREP',TLabel(Components[i]).Caption]),[loCaseInsensitive]) then
      begin
        TLabel(Components[i]).Caption:=cdsLgcFld.FieldByName('FldLgcName').AsString;
      end;
    end;
  end;
```

対象テーブルのフィールド論理名を
格納した ClientDataSet

```
SELECT TblPhyName, FldPhyName, FldLgcName
FROM LIB.SYMFldName
WHERE TblPhyName = 'VCCIZREP'
```

```
{* 表題：DBGrid の列タイトルを設定する
* 引数 1[i]：CDSLgcName：対象テーブルのフィールド論理名を格納した ClientDataSet
* 引数 2[i]：TableName：対象テーブル名}
procedure TUscDBGrid.fSetLogicalName(CDSLgcName: TClientDataSet; TableName: string);
var
  i: Integer;
  dbgSelf: TUscDBGrid;
begin
  dbgSelf:=self;
  for i := 0 to dbgSelf.Columns.Count - 1 do begin
    if CDSLgcName.Locate('TblPhyName;FldPhyName',
      VarArrayOf([TableName,dbgSelf.Columns[i].FieldName]),[loCaseInsensitive]) then
    begin
      dbgSelf.Columns[i].Title.Caption:=CDSLgcName.FieldByName('FldLgcName').AsString;
  end;
end;
```

他のプログラムに影響せず、他からも影響されないため、プログラムの開発効率と保守性が高まり、再利用しやすくなる。

開発効率と保守性を高める工夫

社内 SE の仕事は開発以外にも多岐にわたるため、内製を維持するには開発効率が非常に重要である。また、現場の要望に迅速に応え、競争力を上げるために、保守性も同様に重要である。

そのため、いかに同じコードを記述せずコード数を減らすか、誰が書いても同様の記述になるかを意識しており、そのいくつかを紹介する。

(1) アプリケーションの自動配布

EXE ファイルと Delphi/400 をクライアント PC へ事前にインストールせず、ユーザーの初回起動時に自動でインストールする仕組みを構築した。初回起動時以降は、EXE ファイルや Delphi/400 のバージョンアップを自動で行うようにしており、インストールと更新作業の手間をなくした。

これにより、ひんぱんにアプリケーションの仕様変更や Delphi/400 のバージョンアップがあったとしても、配布の手間を気にせずクライアントを最新の状態にすることができる。

各クライアントの設定を保管する ini ファイルに関しては、初回起動時にはファイルをコピーしているが、それ以降は ini ファイルの内容をチェックして追加されたセクションおよびキーのみ追加し、設定情報を消さないようにしている。64bit 版 OS のクライアントも対応済である。【図 2】

(2) フィールド論理名管理

DBGrid の列名や入力項目（DBEdit）のラベル Caption をデザイン画面で設定するのではなく、データベースに格納したフィールド論理名を画面表示時にコードから設定することにした。これにより、類似した記述の重複をなくし、変更箇所を一元化することで開発効率を上げている。

DBGrid の列名設定のコードは、DBGrid を拡張したオリジナルコンポーネントに記述している。入力項目のラベル Caption 設定箇所は、使用しないフォームもあるため汎化していない。【ソース 1】

(3) 可変メニュー

メニュー内容は、データベース化したものをツリービューを使って表示する仕組みとした。フォームの変更が不要であるためビルド回数を減らせるとともに、各メニューの有効無効を切り替えるといった対応も、データベースの変更のみで可能とした。

メニューは、開発側で用意した標準メニューと、各自がオリジナルメニューを作成できるお気に入りメニューの 2 つを用意し、各自で工夫できるようにした。【図 3】

メニュークリック時に開くフォームは、データベースに文字列として格納することになるため、他の情報と合わせ record 型として TTreeNode の Data プロパティへ格納し、メニュークリック時に読み出している。

フォームクラスは、FindClass 関数で探せるようにフォームの Create イベント等で事前に RegisterClass 手続きを使い、登録しておく必要がある。【ソース 2】

(4) 入力値チェック

オリジナルコンポーネントのプロパティに設定した値を使用し、全角 / 半角 / 半角大文字 / 数値型に合致する文字列かどうか、文字型フィールドでは EBCDIC 換算した文字数に収まっているかどうか、数値型フィールドでは数値フォーマットに合致するかどうかの妥当性チェックを行っている。

オリジナルコンポーネントへのプロパティとイベント設定は、コード上で行っている。ただし、手作業でのコーディングは行わず、Excel 化したテーブル仕様書（※）をもとに入力項目仕様書（図 4）を作成し、Excel 関数で作成したコードを貼り付けているだけである。【図 4】【ソース 3】【ソース 4】

また、妥当性チェックのコードも共通クラスに記述してあるため、ほぼノーコーディングで行っている。【ソース 5】

※ DSPFFD コマンドで FILE 化し、ODBC 経由で Excel に取り込む

(5) 値選択とローカルキャッシュ

区分やマスター値については通常、CD や ID としてフィールドに格納されているが、それを名称や他のフィールド値から検索して選択するための入力補助機能を備えている。【図 5】【図 6】

選択画面の Grid に表示される内容は事前にキャッシュしておき、高速な表示や検索を実現した。選択画面表示時にリフレッシュも可能である。

区分値選択のキャッシュは各フォームでそれぞれ行い、マスター値選択のキャッシュはキャッシュ用のクラスで行っている。【ソース 6】

(6) 値変換とエンティティクラス

各レコードの詳細を表示する欄では、DataSource の DataChange イベントを使いフォーム表示時やデータ変更時に CD や ID を名称に変換し、CD や ID とは別にわかりやすく表示している。【図 7】【ソース 7】

SQL 文に ID や CD に対応した区分やマスターテーブルを結合して表示せず、事前にエンティティクラスに読み込んだデータを使用している。そのため高速で、CD や ID を直接入力した際にもサーバーと通信することなく、表示内容を更新できる。【ソース 8】

また、手入力した CD や ID がデータベース上に存在するかどうかも ClientDataSet の Validate イベントを使い、その場でチェック可能としている。【図 8】

今後の予定・計画

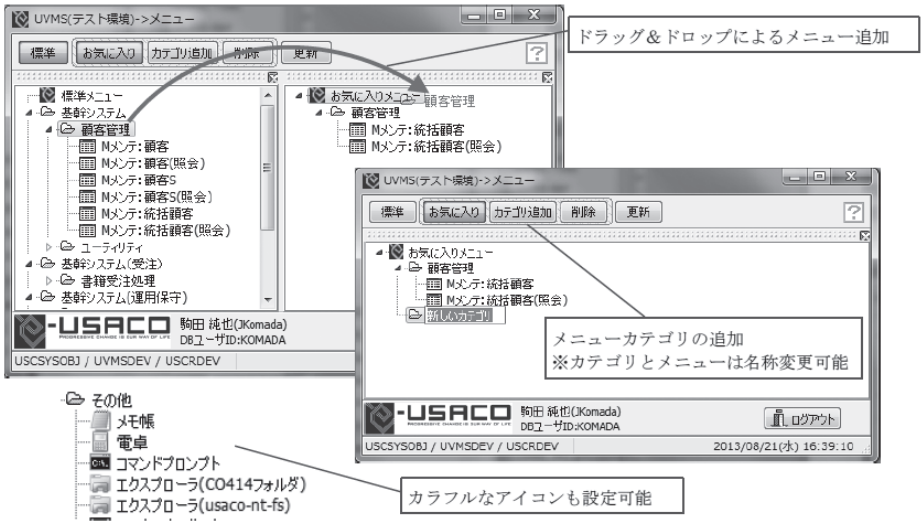
(1) 業務ポータルとしてのグループウェア

Delphi/400 とは別の統合開発環境で構築しているグループウェアの Web サービス連携機能について、これを Delphi/400 に移植し、更新情報やアラートをグループウェアに最新情報として書き込めるようにする。

(2) 印刷方法の拡張

現在の印刷方法は、帳票サーバーへ CSV データとしてデータを渡して印刷する方法しかなく、その場でプレビューすることができない。そのため、QuickReport や Excel を使った出力手段を実装したい。

図3　可変メニューの仕組み



ソース2　レコード型定義のサンプルコード

レコード型定義のサンプルコード

```
//各メニューノードの情報格納用レコード型
TMenuData = record
  FuncID:      string;  //機能 ID
  ReqAuthID: string;  //必要権限 ID
  FormName:   string;  //フォーム名
  StartPath: string;  //起動 Path
  DispMode:   string;  //表示モード
  FormMutex: string;  //フォーム排他
  ImageIdx:   Integer; //イメージ Index
end;
PMenuData = ^TMenuData; //TMenuData のポインタ型
```

フォームクラス事前登録のサンプルコード

```
//findClass で利用するための準備
RegisterClass(TfrmMntUser);
RegisterClass(TfrmMntUsrAuth);
RegisterClass(TfrmMntCust);
```

レコード型からデータを取り出すサンプルコード

```
var
  frmClass: TfrmUscBaseClass; //フォームは全て UscBase を継承している
  frm: TForm;
  exeText, paramText: PChar;
begin
  { 機能の起動処理 }
  //フォーム名が設定されているかどうかで処理を分ける
  strFormName:=TMenuData(TTreeView(Sender).Selected.Data^).FormName;
  if strFormName <> '' then begin
    strFrmDispMode:=TMenuData(TTreeView(Sender).Selected.Data^).DispMode; //Read:読取(照会)
    //フォーム起動
    frmClass:=TfrmUscBaseClass(FindClass(strFormName));
    frm:=frmClass.Create(Self,strFrmDispMode);
    frm.Show;
  end else begin
    //外部プログラムを起動
    exeText:=PChar(TMenuData(TTreeView(Sender).Selected.Data^).StartPath);
    paramText:=cfGetParamStr(exeText, pgmPath);
    ShellExecute(Handle, nil, PChar(pgmPath), paramText, nil, SW_NORMAL);
  end;
```

図4　入力項目仕様書

入力項目仕様書

英字名	表記	任意	初期値	非表示	マスタ	区分	IME開	数値	半角	半UC	JP	型	長	桁	小
CLSTFORMAT	免税顧客区分		0			○	○	○				P		1	0
CCEFFICIENT5	係数 書籍固定値A	○	0.00				○	○				P		5	2
CCEFFICIENT6	係数 備考	○									○	O	60		
YOBIDATE1	日付 予備1	○	0	○								P		8	0
YOBIDATE2	日付 予備3	○	0	○								P		8	0
YOBIDATE3	日付 予備2	○	0	○								P		8	0
YOBIFLD1	FLD予備1	○		○								A	10		
YOBIFLD2	FLD予備2	○		○								A	10		
YOBIFLD3	FLD予備3	○		○								A	10		
OTHERKBN1	その他の区分1	○		○								A	1		
OTHERKBN2	その他の区分2	○		○								A	1		
OTHERKBN3	AgentRank	○				○	○		○			A	1		
OTHERKBN4	支払日休日時	○				○	○		○			A	1		

必須	任意	IME開	数値	半角	半UC	JP	初期値
dbeCUS		dbgMain.	if fldNam				
	cdsCus	dbgMain.	if fldNam				cdsCust\
	cdsCus					if fldNa	cdsCust\
	cdsCus						cdsCust\
	cdsCus						cdsCust\
							cdsCust\
							cdsCust\
							cdsCust\
							cdsCust\
							cdsCust\
	cdsCus	dbgMain.		if fldNar			cdsCust\
	cdsCus	dbgMain.		if fldNar			cdsCust\

Excel 関数によるコード作成

(3) フィールド指定検索の拡張

対象テーブルの指定フィールドでレコードを検索する機能があるが、CD や ID でしか検索できない。これを、関連テーブルの名称（例えば、社員マスターにある社員名）を使って検索できるように拡張する。（※）【図 9】

※「含む」「=」等のオペランドリストは、フィールド型に対応した内容に自動変更。

(4) バッチ処理の実行管理

現在は IBM i 上でバッチ処理を実行しているが、Delphi/400 への置き換えも検討している。その場合は Windows サーバー上で動くことになるが、Windows のタスクスケジューラーを使うのではなく、独自の実行管理アプリケーションの構築を考えている。

フレームワーク構築で感じたこと

当社のように少人数で自社システムの開発運用を行うためには、いかに冗長コードやデータを減らし、再利用性を高めるかが重要であると考えられる。しかし、情報システム部門にはインフラ管理やユーザーサポートをはじめ、開発以外の仕事についても多くの時間を要し、開発自体も次々に相談や依頼が舞い込んでくる。

後々のことを考えれば、開発効率を上げるためのリファクタリングが重要であることはわかっている。とはいえ、ユーザーにとって目に見えるものではなく、その効果を捉えにくいものであるため、時間を割り当てにくいというジレンマを感じた。

今後も拡張やリファクタリングを続けていくことになるが、自社用にゼロから構築するのは情報量も少なくなかなか骨の折れる作業であり、テクニカルレポートのようなユーザー同士の情報共有がさらに活発に実践されていけば、IBM i と Delphi という強力な組み合わせがもっと広がるのではないかと感じた。

ユーザー企業が集まり、勉強会という形でアイデアを出し合ったり相談をする機会があってもよいとも思う。

M

ソース3　Excel関数で作成したコードサンプル

```
//任意入力フィールド
cdsUnifyCust.FieldByName('UNIFYCUSTNAMEENG').Required:=False;
//'Numeric'数値フィールド制限
fSetDispObjPrpNum(dbeUNIFYCUSTNO,ekNumeric,ctNumeric,ttNone,5,2,False);
```

ソース4　オリジナルコンポーネントのプロパティ設定サンプルコード

```
//表題：画面表示用オブジェクト(TUscDBEdit)へ、プロパティをセットする ※Numeric 専用
procedure TfrmUscBase.fSetDispObjPrpNum(dbeSrc: TUscDBEdit; EditKind: TEditKind; ChkType:
TChkType; TrnsType: TTrnsType; IntSize, FracSize: Integer; AllowMinus: Boolean);
begin
  dbeSrc.prpEditKind:=EditKind; //種別
  dbeSrc.prpChkType:=ChkType; //チェックタイプ
  dbeSrc.prpTrnsType:=TrnsType; //変換タイプ
  dbeSrc.prpIntSize:=IntSize; //NUMERIC 型のフィールド長 ※NUMERIC(X,Y)の X 部分
  dbeSrc.prpFracSize:=FracSize; //NUMERIC 型の小数部長さ ※NUMERIC(X,Y)の Y 部分
  dbeSrc.prpAllowMinus:=AllowMinus; //マイナスの入力を許可するかどうか
  dbeSrc.ImeMode:=imClose;
  dbeSrc.OnKeyDown:=dbeOnKeyDownNum;
  dbeSrc.OnKeyPress:=dbeOnKeyPressNum;
end;
```

イベントもコード上で設定

ソース5　入力チェック時のサンプルコード

```
//表題：NUMERIC 型に対応した DBEdit の汎用イベント
procedure TfrmUscBase.dbeOnKeyPressNum(Sender: TObject; var Key: Char);

  //仮想文字列の作成
  dbeTemp:=TUscDBEdit(Sender);
  strTemp:=fGetVirtualStringNum(dbeTemp.Text, dbeTemp.SelStart, dbeTemp.SelLength, Key,
dbeTemp.prpAllowMinus);

  //仮想文字列が入力規則に合致しているか
  //※intSize フィールドサイズ fracSize 小数部の桁数 ※整数部＝フィールドサイズ - 小数部桁数
  if not cfChkNumberFmt(strTemp,dbeTemp.prpIntSize,dbeTemp.prpFracSize,dbeTemp.prpAllowMinus)
then Abort;
end;

//表題：対象値が NUMERIC フィールド定義に合っているかチェック
function cfChkNumberFmt(strTemp: string; intSize, fracSize: Integer; allowMns: Boolean): Boolean;

  if (Pos('-',strTemp) > 0) and (not allowMns) then Abort; //マイナスが許可されていないのに入っていないか
  if Pos('-',strTemp) > 0 then strTemp:=StringReplace(strTemp,'-','',[]);
  //小数点があるか
  if Pos('.',strTemp) > 0 then begin
    iIntPart:=Length(Copy(strTemp,0,Pos('.',strTemp)))-1; //整数部の桁数
    iFracPart:=Length(Copy(strTemp,Pos('.',strTemp)+1,MaxInt)); //小数部の桁数
  end else begin
    iIntPart:=Length(strTemp);
    iFracPart:=0;
  end;
  //桁数がフォーマット内か
  iIntMax:=intSize - fracSize; //整数部の最大桁数
  iFracMax:=fracSize; //小数部の最大桁数
  if (iIntPart > iIntMax) or (iFracPart > iFracMax) then Result:=False
  else Result:=True;
end;
```

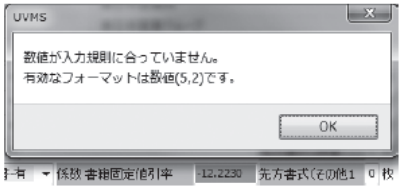


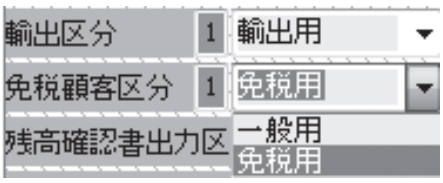
図5　マスター値選択:社員コード

図6　区分コード

ソース6　区分選択画面呼び出しのサンプルコード

```
fldName:=TUscDBEdit(Sender).Field.FieldName;
trnsType:=TUscDBEdit(Sender).prpTrnsType;
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//区分の場合
  frmKubunSelect:=TfrmKubunSelect.Create(self);
  strKubunID:=TUscComboBox(FindComponent('cbx'+fldName)).prpKubunID;
  //区分選択画面内の ClientDataSet(cdsSelect)へデータ追加
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  while not cdsKubun.Eof do begin
    if cdsKubun.FieldByName('KubunID').AsString = strKubunID then begin
      frmKubunSelect.cdsSelect.AppendRecord(['',cdsKubun.FieldByName('KubunCD').AsString,
        cdsKubun.FieldByName('KubunCDName').AsString]);
    end;
    cdsKubun.Next;
  end;
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  //選択フォーム表示
  try
    if frmKubunSelect.ShowModal = mrOk then begin
      //区分値のセット
      if not cdsLclValue.Modified then cdsLclValue.Edit;
      cdsLclValue.FieldByName(fldName).AsString:=
        frmKubunSelect.cdsSelect.FieldByName('KubunCD').AsString;
    end;
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  end;
```

図7　コードから名称に変更して表示



ソース7　DataSourceのDataChangeイベントでの呼び出し例

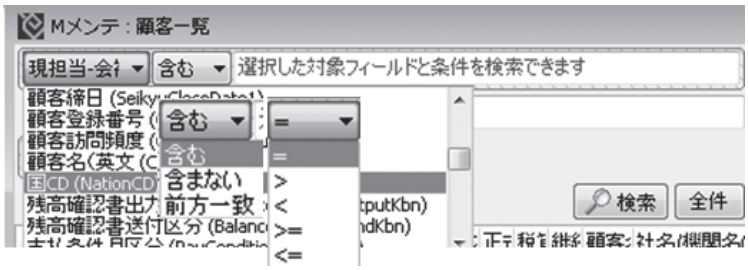
```
//レコード移動時には全項目を、フィールド値変更時には該当項目だけを処理
notAssigned:=not Assigned(Field);
if Assigned(Field) then fldName:=Field.FieldName;

//マスターデータエンティティから参照
if notAssigned or (fldName = 'CLASSIFIED') then fSetNameTrnsFields('Edit', 'CLASSIFIED',
fdmEntity.cdsClass, edtClassName);

//区分データから参照
if notAssigned or (fldName = 'EXPORT') then fSetNameTrnsFields('KubunCbx', 'EXPORT', cdsKubun,
cbxExportKbn);
```

ソース8　名称表示用オブジェクトに値をセットするサンプルコード

```
//引数 1[i]：TrnsType：変換タイプ、 引数 2[i]：FldName：対象のフィールド名
//引数 3[i]：cdsTrns：値変換用の CDS、引数 4[i]：DspObj：画面に名称を表示するコンポーネント
procedure TfrmUscBase.fSetNameTrnsFields(TrnsType, FldName: string; cdsTrns: TClientDataSet; DspObj:
TObject);

begin
  if TrnsType = 'Edit' then begin //マスターデータエンティティから参照
    edtTemp:=TUscEdit(DspObj);
    blnDetect:=cdsTrns.Locate(edtTemp.prpCDFldName,cdsLclValue.FieldByName(FldName).AsInteger,[]);
    if blnDetect then begin
      edtTemp.Text:=cdsTrns.FieldByName(edtTemp.prpTgtFldName).AsString;
    end else begin
      edtTemp.Text:='該当なし';
    end;
  end;

  if TrnsType = 'KubunCbx' then begin //区分データから参照
    cbxTemp:=TUscComboBox(DspObj);
    if cdsTrns.Locate('KubunID;KubunCD',
      VarArrayOf([cbxTemp.prpKubunID,cdsLclValue.FieldByName(FldName).AsString]),[]) then begin
      cbxTemp.ItemIndex:=cbxTemp.Items.IndexOf(cdsTrns.FieldByName('KubunCDName').AsString);
    end else begin
      cbxTemp.ItemIndex:=-1;
    end;
  end;
end;
```

図8　存在チェック結果

図9　検索画面の拡張



ゴールド賞

Delphi/400でCTI開発および関連機能組み込み ーはたしてDelphi/400で CTIは実現可能なのか?

仲井 正人 様
株式会社スマイル・ジャパン
システム部

株式会社スマイル・ジャパン
http://www.care-cure.jp/company/index2.html

健康食品の通信販売事業を行っている。健康食品の研究･開発を通して、お客様の健康づくりをサポートし、健やかでいきいきとした日々をすごしていただくことを目的として活動している。

通信販売事業と注文の流れ

当社では健康食品を研究・開発し、通信販売により商品をお客様へお届けし、お客様のお悩みを少しでも解消することを目的として運営している。基幹システム IBM i（AS/400）で「顧客管理システム」を構築しており、広告・受注・発送・入金・督促・DM の管理をしている。

お客様との関係は、例えば広告媒体については、テレビ・新聞・雑誌・ラジオ・インターネット等、多岐にわたり展開している。

お客様からの注文は、電話・FAX・はがき・インターネットで受け付けている。注文受票は、電話注文の場合は手書きで作成し、インターネット注文の場合は EC カート自動送信メールのプリントアウトを注文受票としている。

注文受票作成後からの業務がシステムの対象範囲で、顧客マスター登録および受注入力を行い、発送・入金・(督促)・DM の流れをシステム化している。

Delphi/400導入前のCTIの課題

顧客管理システムは IBM i（5250 画面）を利用しているため、そのままでは CTI の PC 連携機能が使えない。イリイの「BIG 顧客 ProCTI」の提案があり、導入することになったが、以下の利点と難点があった。

・利点

電話履歴（電話が何時何分にあったか等）が自動で残せる点と、その履歴に対応内容を追記し残せる点だ。その結果、顧客との次回の電話応対時に、スムーズに話ができるようになった。

・難点

顧客データと電話履歴データが累積されるだけなので、そのままでは受注以降の業務ができない点がまず挙げられる。また､顧客管理システム（IBM i）とデータを同期化するためには､IBM i と「BIG 顧客 ProCTI」にすべての顧客を登録、変更しなくてはならず、二重作業や二重管理に苦労する点も挙げられる。

当面の運用としては、IBM i と「BIG 顧客 ProCTI」の顧客マスターの同期化は諦め、「BIG 顧客 ProCTI」は単なる古い電話帳データを参照するだけとした。他方､「BIG 顧客 ProCTI」の電話履歴データについては IBM i に受け渡して、顧客管理システム側でも電話履歴を参照できるようにした。

このような状況と問題点により、CTI のメリットが活かしきれていなかった。そのため今回、Delphi/400 を導入し、IBM i に CTI を組み込んだシステム構築を計画した。

Delphi/400でのCTI開発について

(1) CTI 動作環境の準備

はたして Delphi/400 で CTI は実現可能なのか、わからないところからのスタートだった。

図1　CTI ActiveXのテスト取込手順

【問合せに対するミガロ. からの回答(抜粋)】

取り込んだActiveXコントロールをコンポーネントとして利用するには、新規または
既存のパレットページを指定し、新規または既存パッケージへ作成したコンポーネ
ントを追加後、パッケージの登録処理を行っていただけば、利用できると思います。

ActiveXコントロールの詳細仕様は未確認ですが、「OnOffering」、
「OnChangeCallState」、「OnChangeAgentNumber」、「OnDisconnectedHeld」
などのイベントにより、通話の開始・終了が制御できるのではないかと思います。

「OnConnected」と「OnDisconnected」は、このコンポーネントが電話機の親機との
通信開始・終了を表すイベントか、または、こちらが通話関係の可能性もありますので、
詳しくはActiveXコントロールのマニュアル等をご確認ください。

取り込んだコンポーネントの動作については、テストプログラムを使って確認して
いただく方が良いかと思います。
先ずは、ActiveXコントロールの詳細仕様を確認し、実際にコンポーネントとして
登録し、テストプログラムにより動作を確認していく、という手順になると思います。

図2　ツールパレット追加&設計画面

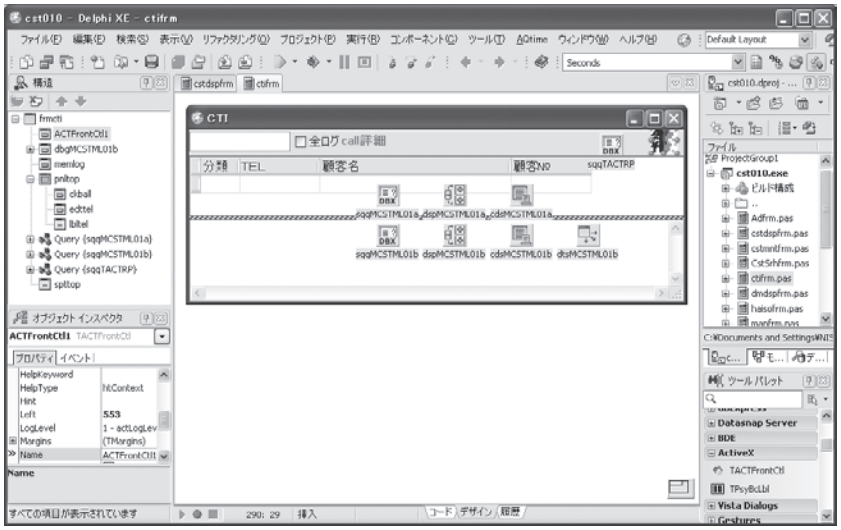
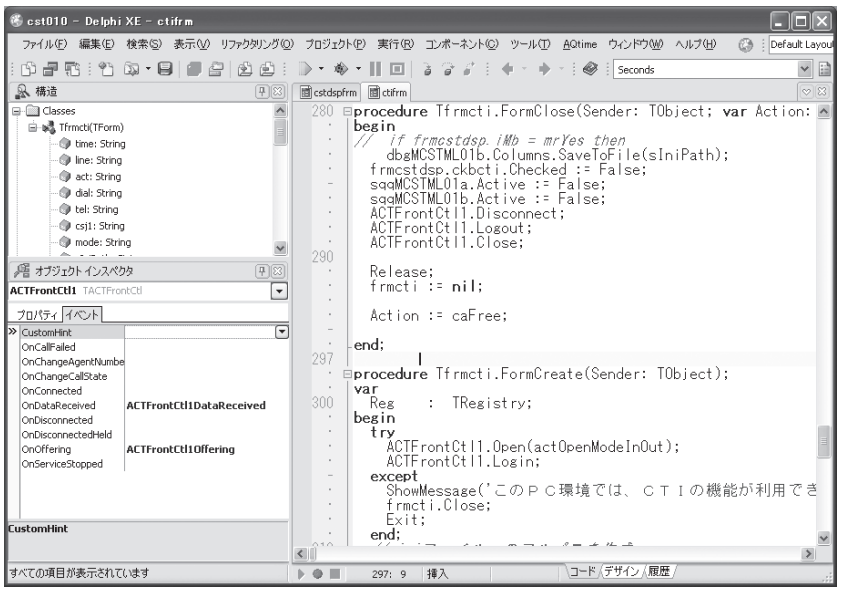


図3　ロジック&イベント



CTIを動作させるには、まずCTIに対応可能な電話の主装置が必要である。主装置は、「BIG顧客ProCTI」の導入時に、NTTのα NXに切り替えた。そして、PCにNTTより納品されるNX－CTIミドルウェアをインストールする。このインストールにより、電話機(内線番号）と電話の主装置とPCの紐付けが行われる。また、NTT製ソフトの中にはα CTベーシックシステムがあり、そのソフト起動により電話機とPCの紐付けができているかを確認できる。

(2) Delphi/400とCTI連携方法の確認

α CTベーシックシステムは、ActiveX CT協議会のACTコントロールに対応とのことなので、ミガロ．導入支援担当に確認してみた。すると、Delphi/400のコンポーネントとしてインポートできるとの回答を得た。

早速インポートを行うと、ActiveXコントロールからプログラムソースが展開され、そのソースをpas（Delphiソースファイル）で保存した。

プログラムソースの展開や保存後の方法について、再度ミガロ．へ問い合わせをし、回答を得た。【図1】

コンポーネントのインストールを行うと、ツールパレットにActiveXのページができて、その中にACTFrontCtlのコンポーネントができたので、そのコンポーネントを画面に貼り付けた。【図2】

そこから先がまたわからなかったが、ActiveXから展開したソースを見ていると、参照元がC:¥WINDOWS¥system32¥ACTFront.ocxであることがわかった。同じフォルダーのACTFront.ocxを見ていると、並びにACTFRONT.HLPファイルを発見し、これを開いてみるとアプリケーション開発、リファレンス、定数一覧があり、参照しながら組み立てることができた。

(3) CTI連動システム開発のポイント

Delphi/400による「新システム」では、顧客マスター保守がメインで、CTIはあくまでも1機能とした。電話を利用しない部署もあるため、顧客マスタ保守は必要最低限の機能を共通機能とし、利用したい機能を追加で選択できる形にした。他方、電話を受ける部門では、CTIで自動表示できる情報を最大限に発揮できる内容を目指した。

電話番号に関する実務面の工夫として、顧客情報にはメイン番号（通常は固定電話）・携帯電話番号・別送先電話番号の3つが存在しているケースがあり、それぞれをCTIで連動させることとした。また、同じメイン番号に対して1家族内から数人が登録されることもあり、電話番号+’A’、’B’……と複数登録されているケースにも連動させることとした。

CTIの仕様としては、電話の着信時は「顧客マスター保守」画面上部に複数候補から1行だけを自動的に表示し、通話時には、複数候補の1番目を自動で展開させ、一覧選択で切替可能とした。

(4) Delphi/400でのプログラミング内容

CTI機能フォームを作成し、ACTFrontCtlのコンポーネントを貼り付ける。他の必要部品もフォームへ貼り付けて、画面を完成させる。【図2】

ロジックとしては、FormCreateの時にOpenとLogin、FormCloseの時にDisconnectとLogoutとCloseをする。ACTFrontCtlのイベントOfferingには、電話が鳴った時の処理を記述した。ACTFrontCtlのイベントDataReceivedには、通話した時の処理を記述した。【図3】

また、メイン・携帯・別送電話の連動については、3つのSELECT文をUNIONでつなぐことで候補一覧を表示し、選択で切り替え可能とした。電話番号+’A’、’B’……と複数登録されている対応には、WHERE中でLIKEを使用し、前方一致条件とした。

システム開発についての評価と感想

・Delphi/400を知らなかったSEが1～2か月で実現

CTIの動作環境は揃っていた。Delphi/400で構築するための調査はミガロ．の支援により実現できた。開発のポイントはテクニカルセミナーなどでヒントを得た結果、プログラミングは非常にシンプルであった。

最初の「はたしてDelphi/400で、CTIは実現可能なのか？」は、やってみると「簡単にできた」の一言に尽きる。

現に、2012年3月にDelphi/400導入決定、4月インストール、5月トレーニングコース（開発入門、開発基礎）受講、そして6月にはCTIをDelphi/400で動かすところまでできた。その後、微調整等はあったが、Delphi/400を知らなかったSEがわずか1～2か月で構築できた。

ここで得た結論としては、Delphi/400にActiveX等を組み込むことにより、わずかな時間で意図していることが実現可能ということだ。

・お客様とCSの関係をスムーズに

ここまでが現状のCTI開発であるが、今後は、CTIで顧客情報に関して芋づる式にどれだけ必要な内容を表示できるかがカギであり、お客様とCS（カスタマーサービス）部門の関係をいかにスムーズにさせるかに焦点をあてたい。

現在表示できる内容は、電話対応履歴・購入履歴・購入履歴1番目の荷物追跡(通常：佐川Web、メール便：ヤマトWeb)・定期登録内容・DM履歴・Google住所マップ・メール履歴となっており、これらをさらに充実させていきたい。

今後の予定はまだ構想中ではあるが、メソッドとしてCALLがあり、PC上で発番（電話発信）可能なため、CS担当者ごとに本日電話する一覧を表示し、選択すると発番する仕組みを構築できたらと検討中である。

エンドユーザーの評価・評判・反応

CTIの恩恵を一番受けるのがCS部門であるが、平均年齢が高めでもあり一度IBM i（5250画面）で覚えているので、一気に切り替えが難しい状況にあり、現在のところ併用している。

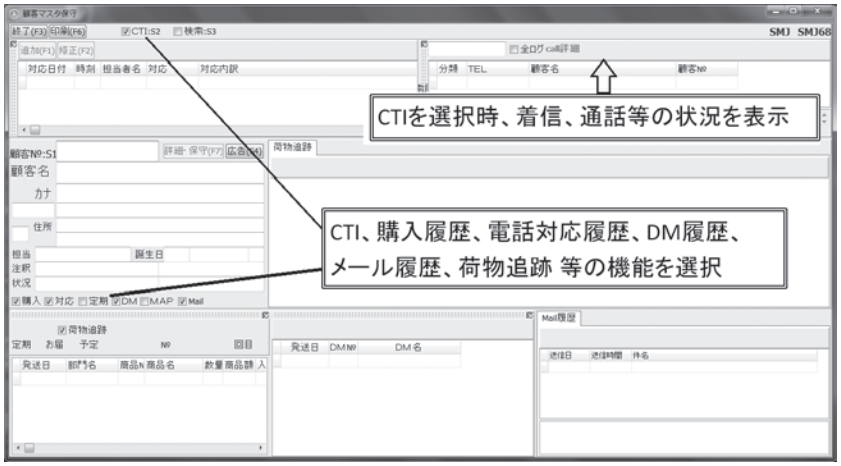
新システムの使用感想としては、「以前は画面遷移が多いため、一度に複数の情報が見られなかったが、新システムでは一度で情報が見え画面遷移せず処理ができるため、作業効率が上がった」や「一度に見られる情報が増え、自動で表示されること」との評価があった。この評価は、5250画面からDelphi/400に切り換えたこと、およびCTI機能も含めてのものである。【図4】～【図9】

CTI機能の利用は電話対応において

図4　顧客マスタ保守　初期画面(機能未選択)

図5　顧客マスタ保守　顧客マスタ詳細・保守(機能未選択)

図6　顧客マスタ保守　初期画面(機能選択)



柱になるところであり、CTI へのさらなる機能追加によりお客様対応にも大きな効果を発揮する。今後も CTI はもちろん、システムに必要な機能を追加していきたい。【図 10】【図 11】

M

図7　顧客マスタ保守　着信時(機能選択)

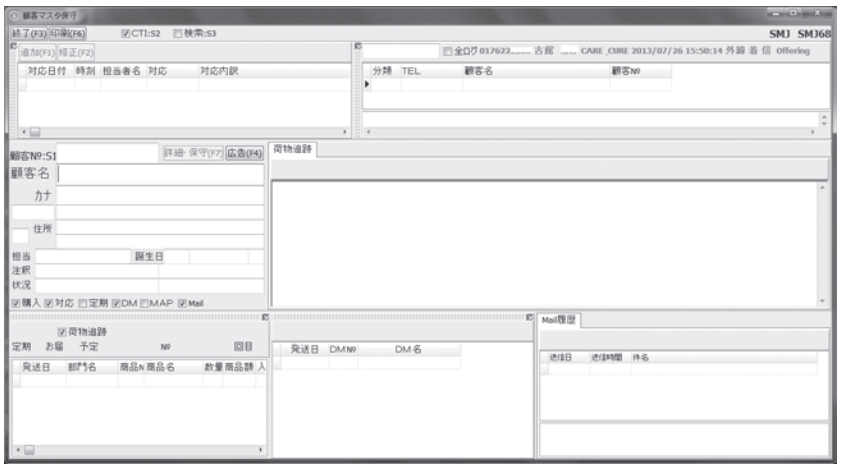


図8　顧客マスタ保守　通話時(機能選択)

図9　顧客マスタ保守　取引履歴確認(機能選択)

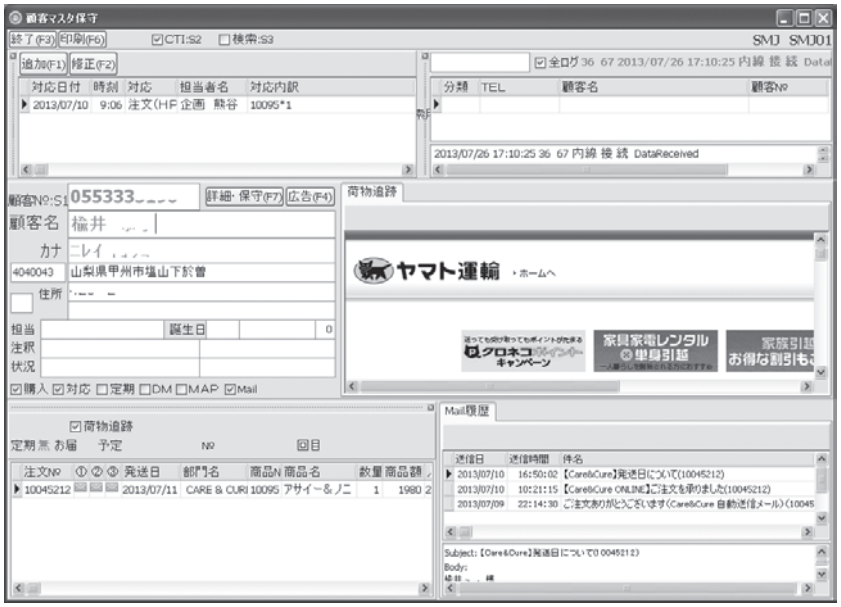
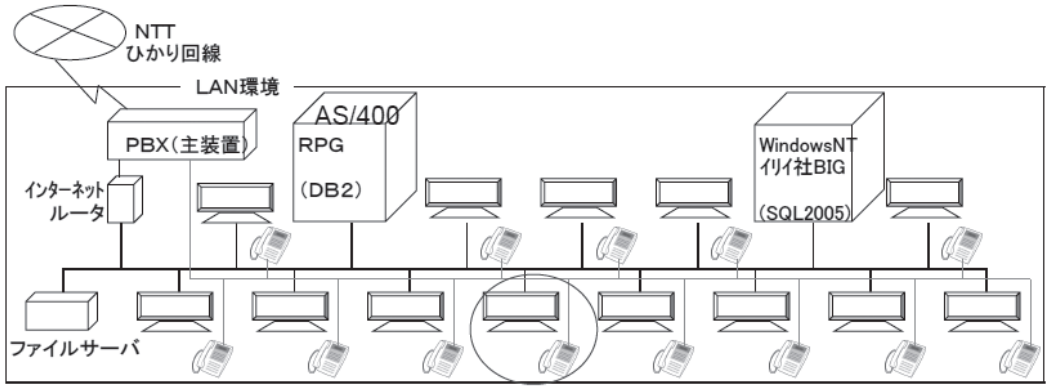
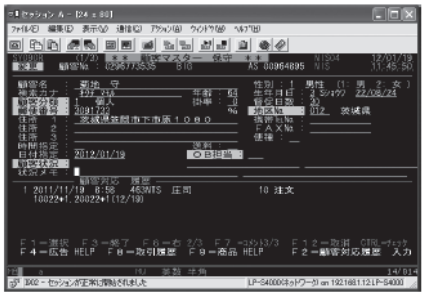


図10　システム構成(対応前)

■Before



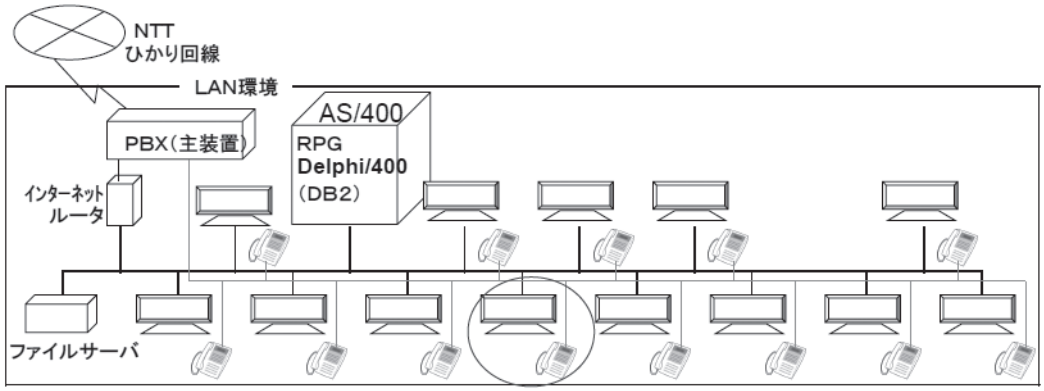


(5250画面)　　・顧客マスタ保守(15画面)
・定期休止　処理　　・定期購入マスター保守
・定期休止解除処理・定期中止(解約)　処理

イリイ社　BIG顧客proCTI

図11　システム構成(対応後)

■After



■2つのシステムを統合
- ・5250画面
- ・イリイ社　BIG顧客proCTI(CTI機能)

■5つの処理を統合
- ・顧客マスタ保守(15画面)
- ・定期購入マスター保守
- ・定期休止　処理
- ・定期休止解除　処理
- ・定期中止(解約)　処理

■機能追加
- ・住所GoogleMAP表示
- ・荷物追跡表示(通常:佐川、メール便:ヤマト)
- ・メール履歴
- ・住所検索機能(ｸﾞﾙｰﾌﾟﾃｸﾉﾛｼﾞｰ社郵便番号検索API)

シルバー賞

IBM WebFacingからJC/400への移行・リニューアル手法
ーJC/400へのスムーズな移行とWebの操作性向上で顧客満足度UP

八木 秀樹 様

極東産機株式会社
管理本部 社長室
システム開発課 主任

極東産機株式会社
http://www.kyokuto-sanki.co.jp/

昭和 23 年に畳の製造機器メーカーとして創業以来、職人さんの快適な職場環境作りと消費者の豊かな生活空間作りを 2 本柱として、伝統技術と先端技術の融合により、ユニークなオリジナル商品を開発。畳製造機器はもとよりインテリア施行省力機器、カーテン縫製機器等、幅広く事業を拡大している。

1.短期間でのシステム移行が求められた経緯

極東産機では、Web による「畳ネットワーク工事管理システム」を 2000 年に本稼働させ、2006 年には基幹システムとの連動を図るために、WebFacing により刷新し、機能拡張を行った。

当システムは当社と当社の取引先である畳店､材料店､工事依頼先とを結ぶネットワークで、IBM i + WebFacing によってすべて社内開発し稼働させた。WebFacing は IBM i 上で RPG 言語で開発し、IBM i のデータベースに直接アクセスしている。

今回のシステム移行は、システムを稼働させているサーバー（WebFacing 管理サーバーおよび IBM i ）の老朽化によりハードのリプレースは決定していたが、更改後の新サーバー上で WebFacing が稼働しないのと、WebFacing について問い合わせをする窓口もすでにないことから、迅速かつ確実に別システムへの移行が求められる状況になっていた。

そこで、2011 年 12 月に JC/400 で作成した「得意先との Web-EDI」（※）が好評で、さらに「仕入先との Web-EDI」（2012 年 5 月から本稼働）を稼働させて実績作りができたことで、WebFacing による「畳ネットワーク工事管理システム」を JC/400 へ完全移行することが決定した。

※『ミガロ. テクニカルレポート 2012 年』掲載の「JC/400 による取引先との Web-EDI システム構築」（Migaro. Technical Award 2012 最優秀賞受賞）を参照。

2.WebFacingからの移行についての工夫

(1) IBM i のデータベースは変更しない

JC/400 も WebFacing と同様に、IBM i のデータベースに直接アクセスする。現在稼働中の「畳ネットワーク工事管理システム」で使われているデータベース設計はまったく変更せず、そのまま利用するようにした。

(2) プログラムパターンの分類

プログラム作成の効率を上げるため、WebFacing でシステム作成した当時のプログラムを機能別に分類し、数種類のパターン（入力、照会など）に分けた。パターン化することにより開発効率を上げた。

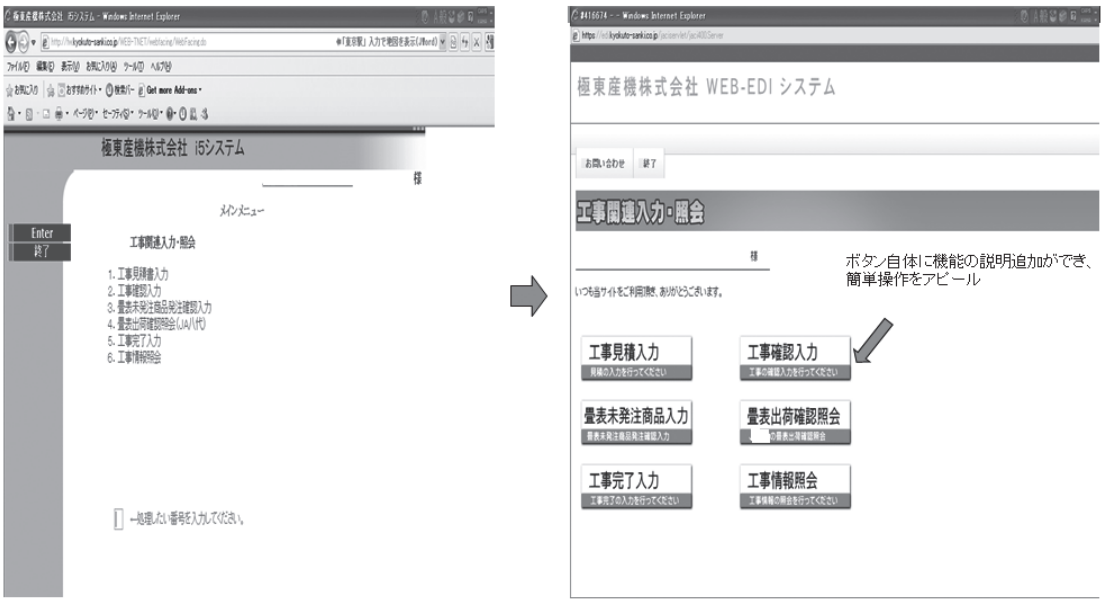
(3) 主要ロジックは WebFacing から移植

更新やエラーチェック、レコード読み込み条件などの主要なロジックは、WebFacing で作成したロジックをそのまま移植した。ほとんどのロジックはそのまま使えるため、デバックもスムーズに行えた。

(4) 画面デザインは慣れるまで変更しない

WebFacing は 24 x 80 の画面制約がある。JC/400 で開発するのだから、画面デザインを変えたいと思ったが、すべての移植が完了し稼働確認するまでは、画面デザインの改良には着手しなかっ

図1　今回の開発前後の画面比較



WebFacingサーバー(刷新前)　　WEB-EDIサーバー(刷新後)

図2　今回の開発前後の画面比較

↑ 日付の選択はカレンダーの日付クリックするだけ

←(左の)ダウンロード(csv)ボタンをクリックし、

↓(下の)ダウンロードで　csvデータ　としてパソコンへ取込OK

た。

すべての構築が完了し、工数に余力が出てから、画面デザインを変更していった。画面デザインは、ネット上や書籍などのテンプレートを活用し、カスタマイズして進めた。また、リーフレットやカタログを製作している社内デザイナーに依頼して、タイトルロゴを作成してもらったので見栄えが一変し良くなった。

(5) 問題点の解消

開発テキストには、よく使用する機能の解説や事例があるため、テキストを読むだけで問題点解決できることが大半だった。

エラーの原因が推測できない場合は、ミガロ．テクニカルサポートを積極的に活用させてもらった。電話やメールでの迅速丁寧なサポートで安心感があり、このサポートがプログラムの開発効率を上げてくれた要因だ。

(6) ホームページビルダーの利用

WebFacing は画面デザインを数種類のパターンから選択するしかなかったが、JC/400 は HTML 画面で自由に設計できるため、ホームページビルダーを活用し画面をデザイン開発することができた。見栄えも良くなり、JC/400 で画面の操作性も上がっており、社内外の関係者から好評を得ることができた。

(7) ノーツや AutoFax サーバーとも連携

リニューアルした際に、どのようにしたらもっと便利になるのかを利用者に調査した。

「入力された情報が自動的に取引先へ連絡できるようになればもっと便利になる」ということで、ノーツメールや FAX 送信サーバーに連動させ、「工事依頼書」などを自動送信できる仕組みを新たに追加した。

(8) 旧 IBM i で開発し、新 IBM i で稼働

新システムは、ハード（IBM i）のリプレースよりも前に仕上げる必要があったために、旧 IBM i 上で開発を進めた。想定していた予定より前倒しで移植が完了したので、旧 IBM i でリニューアル稼働をさせた。

2013 年 5 月、新しいサーバー（IBM i）へ切り替えを行ったが、OS のバージョンが変わっても、作成したシステムにも何の変更もなく順調に稼働した。

(9) 新・旧システムで並行稼働

新システムを作成した際に、数社だけでテスト稼働を実施した。テスト稼働といっても、テスト環境や新旧システム環境での重複入力のテスト運用ではなく、実際の環境で利用してもらい、問題点がないかをチェックした。これは、データベース設計を変更していないからできたことだ。

3.JC/400へ移行・リニューアルの所感

(1) 見栄えの向上と操作性の向上

当システムは、当社の取引のある畳店や工事依頼される取引先によって運用されるシステムである。そのため、運用開始前に説明会を行ったところ、当社の営業担当の第一声が「画面が見やすくなった。操作性も良くなっているので、これなら話が進めやすい。興味を持ってもらえる」であった。

実際、本稼働後も営業担当経由で「簡単にネットで確認できそうなので、ログインする方法を教えてほしい」など、開発者にとってはうれしい問い合わせがあった。【図 1】【図 2】【図 3】【図 4】

(2) 機能の充実

多少の開発が必要であるが、ボタン 1 つでデータの CSV 形式でのダウンロードや Word 形式での伝票印刷機能が装備できたため、非常に便利だ。これらは WebFacing ではできなかったことだ。

データダウンロードの開発では、セキュリティ面を重視し、取引先ごとにプログラムや作成ファイルを変える手法を取った。ミガロ．のテクニカルサポートのアドバイスもあり、難なくスムーズに開発対応ができた。

利用者にはマニュアルで説明するまでもなく、簡単な操作で処理が行えるため、好評を得ている。

(3) 開発の手軽さ・効率のよさ

ホームページビルダー等で画面デザインを行った後、IBM i へプロジェクトを配布し、Web サーバーへ配布する。その後、ユーザーがロジックを組む込むことができるスケルトンプログラム（必要なロジックがコーディングされているプログラム）が自動作成される。

このスケルトンプログラムには、「使用ファイル定義部分」「配列テーブルの定義部分」「パラメーター定義部分」「キーリスト定義部分」「ボタン押下時処理部分」「データの抽出と画面の転送部分」「画面データの取得とファイルへの更新部分」が明確にわかるようになっている。これらの部分に必要なロジックを組み込んで、コンパイルして完了すれば、プログラム制作が完成する。

今回、以前のシステムで稼働していたロジック部分を再考する必要がないため、非常に効率が良く、また難しそうに思えるロジック部分もプログラムに必要な部品をはめ込む感覚で行えるため、慣れてきたころから開発ペースが飛躍的に上がった。

4.今後の展望

iPad 等のスマートパッドでの開発に適している SmartPad4i（SP4i）や、Query を作成する感覚で開発できる Business4Mobile（B4M）を活用して、より便利なシステム構築を推進していく。特に、SP4i の開発手法は JC/400 と同じであるため、開発の効率アップが望める。

近々、iPad を利用した生産管理システムを新たに稼働させる。この開発には SP4i と B4M を利用する。RPG でタブレット開発できるので、私にとっては新たな発見が多く、ワクワクする開発となっている。

さらに、利用者が便利さを実感し、喜んで使っていただける仕組みづくりを推進していきたい。

M

図3　今回の開発前後の画面比較

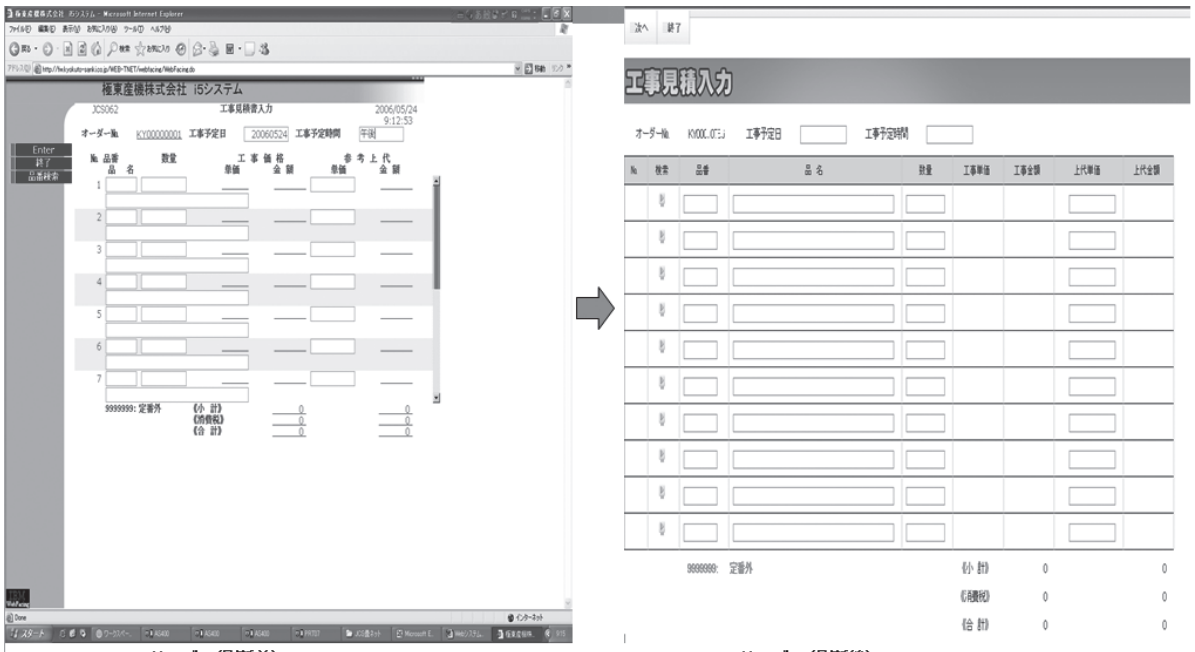
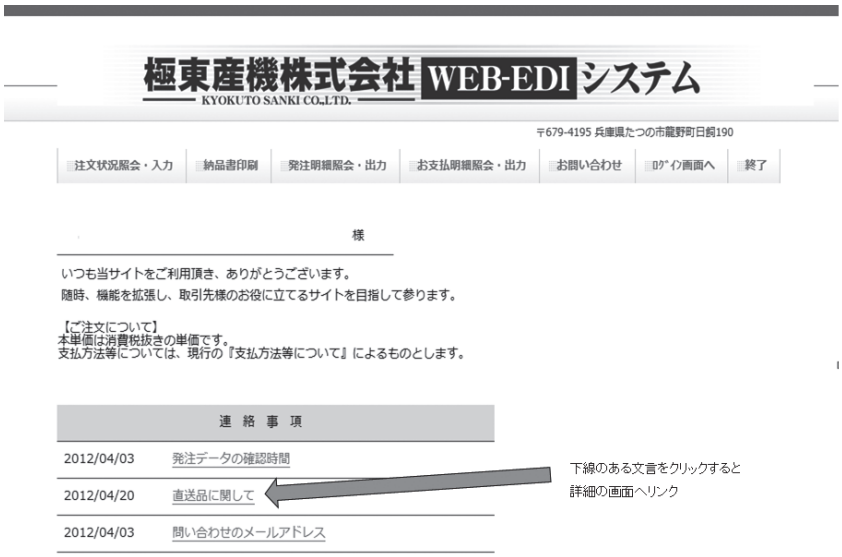


図4　お客様向けメッセージ画面



シルバー賞

Delphi/400とDelphiを利用したIBM i 資源の有効活用
ーIBM i で作成した帳票配布、Delphi + OfficeエンジンによるExcel→PDFバッチ変換

小山 祐二 様
澁谷工業株式会社
経営情報システム部 主任

澁谷工業株式会社
http://www.shibuya.co.jp/

パッケージプラントを主力製品とする東証名証１部上場の機械メーカー。特に、国内外の大手飲料メーカーに採用されているボトリングシステム製造では、世界トップの地位を確立している。また、半導体製造装置、医療機器などの製造も行っている。

はじめに

澁谷工業は1931年創立、1949年設立の会社で、今日まで多くのお客様に支えられ、2011年に創立80周年を迎えることができた。当社では「カスタマーファースト」を貫き、お客様のニーズにあわせたパッケージングプラントを“ターンキー（直ぐに稼働できる状態）”で提供するビジネスを主体としている。

当社の基幹システムにおける既存帳票作成方法や帳票配布方法は、主にIBM iネイティブ環境による帳票（以下「SPOOLF」）とクイックレポート（以下「ツール」）を利用している。しかし、このSPOOLFとツールは利用するうえで強みもあるが、弱みも多い。

そこで本稿では、「Delphi/400とDelphiを利用したIBM i 資源の有効活用」と題して、既存方法とは別の帳票作成や配布方法を紹介する。

具体的には、IBM i でExcelをベースとした帳票を作成し、それをDelphi/400を利用してエンドユーザーに配布する方法、さらには「Delphi + Office エンジンによる Excel → PDF バッチ変換」方法を解説する。

開発経緯

最初にツールにおける帳票作成について簡単に説明する。このツールはヘッダー、明細、フッターごとにセグメントを分け、それらにデータをセットし、帳票を作成するものである。

このツールはレイアウトを作成するうえで、視覚的に非常に理解しやすいものである。しかし、このツールを利用するには、細かな罫線制御やセル内のフォント等の制御をすべてアプリケーション側で行う必要があり、この点が考慮点となる。

当時、帳票作成方法の選択肢に関しては、(1) ツール (2) SPOOLF (3) 新たな帳票作成ツールの導入、があった。

しかし (2) を利用すれば、逆に工数が膨らみ、帳票作成における柔軟度も下がることになる。また (3) を導入するコストもかけられなかったため、このツールを使い続けていたのが現状である。

そこで、他のさまざまな方法を模索する中で、やはりクライアント側ではなく、サーバー側（IBM i ）で処理できないかと考えた。

検証の結果、IBM i でJavaを利用してIFS（IBM iにおけるファイル格納領域）にExcelを作成できることがわかった。ここではその作成方法の詳細には触れないが、Excel自体の既存機能により（条件付き書式や各種セルの書式設定等）、前述の考慮点で挙げた制御が可能となった。すなわち、アプリケーション側で細かな制御の必要なく、帳票を作成できる。

苦労した点／その解決方法

ここでいくつかの問題点が出てきた。それは、IBM i で作成した帳票を、いかにエンドユーザーに配布するかという

図1

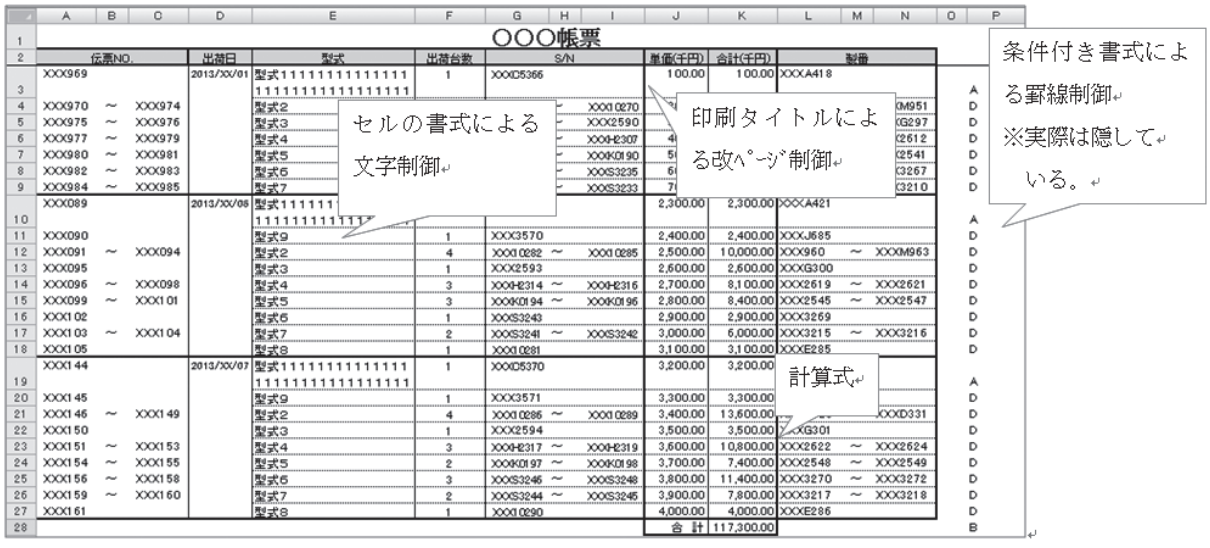
【FTPでIFSのファイルをクライアントPCにGETする方法】

```
open 「IBM i接続先」
user 「IBM iユーザー」 「IBM iパスワード」
binary
quote site namefmt 1
get  「IFSのパス」¥○○○.xls  C:¥□□□.xls
quit
```

ソース1

```
var
  FromName     : string;
  ToName       : string;
  err          : string;
  strFileName  : string;
  strFilePath  : string;
begin

  strFileName :=  Application.Exename;
  strFilePath :=  ExtractFilePath(strFileName);

  FromName    :=  '/「IFS上パス」/○○○.xls' ;
  ToName      :=  strFilePath  + '□□□.xls' ;

  //IFS上Excel→指定PC保管場所
  AS4001.Connect ;
  CoIFS1.Connect ;

  CoIFS1.CopyFileFromAS(FromName,ToName,true,true,false,0) ;

  CoIFS1.DisConnect ;
  as4001.Disconnect ;

end;
```

図2



ことである。

・解決その１

実際にその方法を考えると、最初に、IBM i の NetServer 機能（クライアント側から、IFS をネットワークドライブとして認識できる機能）を利用し、IBM i をファイルサーバーとして利用することが考えられた。しかし、IBM i のコストから考えると、この利用方法は非常にコストパフォーマンスの悪い解決方法になってしまう。

・解決その２

また、IBM i で作成した Excel をメールに添付し、エンドユーザーに配布する方法も考えられる。この方法は、個人別ログイン等でエンドユーザーと送信先メールアドレスを関連付ける必要はあるが、IBM i で Java を利用することにより簡単に実現可能である。

ただし、Excel をメールに添付して送信する方法はセキュリティに問題が生じるため、その帳票の性質を十分に考慮する必要がある。

・解決その３

そこで、次に挙げられる方法として、FTP で IFS の Excel を GET する方法が考えられる。確かにこの方法は、NetServer 機能と抱き合わせて考えることにより可能である。

とはいえ、IFS の Excel を GET する方法を知っている人は少ないのではと思う。参考までに以下にその方法を記しておく。

該当 IFS フォルダーに適切な制御を与え、ネットワークドライブ接続を確立することにより、Delphi をトリガーとして IFS の該当ファイルの取得は可能である。

なお、私の知る限りこのような処理を行っている人は知らないし、一般的にもこのような処理を行っている人は少ないと予想される。【図 1】

・解決その４

そこで私が行った内容を紹介する。当社では Delphi/400 を導入しており、IBM i とやり取りを行っている。一般的な使い方でもかなり有効に活用できるが、この Delphi/400 のコンポーネントの中に、IFS を操作できる CoIFS コンポーネントがある。

当然 Delphi では、IFS をネットワークドライブとして認識させることは可能である。そして信じられないかもしれないが、Delphi と FTP の連携では多少面倒な処理も、Delphi/400 では簡単に処理可能となる。【ソース 1】

しかし、ここで新たな問題点が発生した。検証当初は、拡張子を「xlsx」タイプで Excel を作成していた。このタイプで作成した場合、かなり処理時間がかかることがわかり、実運用には不向きだという結論に至った。

そこでゼロベースで再考した結果、拡張子を「xls」タイプで Excel を作成し、処理時間を比較してみた。その結果、処理時間としては約 1/7 ～ 1/9 くらい短いことが確認できた。幸いなことに、マイクロソフトの発表によれば、拡張子 xls タイプの Excel は Office2013 も対応していることが確認できた。そこで私は、拡張子 xls タイプで Excel を作成することとした。【図 2】

Delphi ＋ Officeエンジンによる Excel→PDFバッチ処理

ある意味、裏技的な処理を紹介する。多くのユーザーが利用している Office というソフトがある。ご存知の通り、Office2007 以降では PDF 変換機能が備わっており、一度該当ファイルを開き、保存する時に PDF を選択すれば簡単に PDF を作成することができる。

しかし、その該当ファイルを開くことなく、バッチ的に PDF を作成する方法はないだろうか？ うれしいことに、マイクロソフトはこれらの情報を公開している。（※）

この情報を参考に、Delphi で該当 Excel を PDF にバッチ変換できることを検証できた。参考までに、そのコーディングおよび実行結果を記しておく。【ソース 2】【図 3】

この検証結果をベースに、Java を利用して IFS に作成した Excel を、Delphi/400 を利用してクライアントに配布、そして IFS に作成した Excel をベースに「Delphi ＋ Office エンジンを利用して Excel → PDF バッチ処理」変換し、それをクライアントに配布することを可能とした。

※ Saving Workbooks to PDF and XPS Formats in Excel2007
http://msdn.microsoft.com/en-us/library/bb407651.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-5

DelphiでExcelとPDFを作成しなかった理由

今回は、IBM i で Excel を作成する方法、およびその Excel から Office エンジンを利用してバッチ的に PDF を作成する方法を紹介した。しかし、Delphi でも Excel や PDF の作成は可能である。

では、なぜ Delphi で Excel や PDF を作成しなかったか？ その理由として、Delphi で Excel を作成する場合は OLE で作成するため、多少の処理時間がかかることが挙げられる。またクライアント側で処理するのではなく、他のプロシージャーとともに IBM i で処理することがよいのではと思う。

また、PDF 作成も無償サードパーティー製品で作成可能である。ただし、罫線の制御やフォントの制御を加味して帳票を作成するには、少し労力が必要である。他方、有償サードパーティー製品を利用するのは１つの手段であり、当然それなりのコストも発生する。

なお、今回の組み合わせによる帳票作成は、当社は Office2007 導入済みということもあり、追加投資を必要としなかった。

メッセージと課題

以上を踏まえたうえで、私は IBM i で Excel を作成することを決断した。そして、その Excel から PDF を作成することとした。

そこで危惧されることは「Java は敷居の高い言語？」と感じる人が多いはずである。しかし、私はそんなことはないと断言する。なぜなら、今回の Java はそんなに難しい考え方を行っていないこともあるが、RPG 畑で育ってきた私自身が実装している。本稿を読んだ人はぜ

ソース2

```
var

  F_ExcelApp  : Variant  ;
  F_ExcelVer  : Currency ;
  F_ExcelBook : Variant  ;

  F_ErrMessage: String   ;

const
  PDF_TYPE    = $00000000;
  QUALITY_TYPE= $00000000;

begin

  //ファイル存在チェック
  if not FileExists(F_FromExcel)  then
  begin
    MessageBox(HWND(Nil),'ターゲットファイルが存在しません。処理を中止します。',
               Nil, MB_OK) ;
    Exit ;
  end;

  //Ｅｘｃｅｌ操作開始
  F_ExcelApp  := CreateOleObject( 'Excel.Application' );

  //Ｅｘｃｅｌバージョン取得
  F_ExcelVer  := StrToFloatdef(F_ExcelApp.Application.Version,0.0);

  //Ｅｘｃｅｌ⇒Ｐｄｆ変換処理
  if F_ExcelVer >=12.0 then

    //Office2007以上
  begin

    F_ExcelBook :=F_ExcelApp.Application.WorkBooks;
    F_ExcelBook :=F_ExcelApp.Application.WorkBooks.Open(F_FromExcel);

    //変換処理
    try

      F_ExcelBook.ExportAsFixedFormat(
          PDF_TYPE,                 //  Type_                    : XlFixedFormatType;
          F_ToPdf ,                 //  Filename                 : OleVariant;
          QUALITY_TYPE,             //  Quality                  : OleVariant;
          True,                     //  IncludeDocProperties     : OleVariant;
          True ,                    //  IgnorePrintAreas         : OleVariant;
          EmptyParam,               //  From                     : OleVariant;
          EmptyParam,               //  To_                      : OleVariant;
          False,                    //  OpenAfterPublish         : OleVariant;
          EmptyParam                //  FixedFormatExtClassPtr: OleVariant)
          );

    except

      on E: Exception do
      begin
        F_ErrMessage:=e.Message;
        MessageBox(HWND(Nil),PChar(F_ErrMessage), Nil ,MB_OK) ;

      end;

   end;

  end

  else
  begin
    //Office2007以下
    MessageBox(HWND(Nil),
    'Office2007以降のバージョンがインストールされていません。処理を中止します。'
     Nil, MB_OK) ;

  end;

  //Ｅｘｃｅｌ操作終了
  F_ExcelApp.Application.quit;

end;
```

ひチャレンジしてほしい。

今後の課題としては、Excel → PDF変換に関してはクライアント側で処理する形となってしまったが、このプロセスをサーバー側で処理できないかを検討したいと思う。

エンドユーザーの評価と今後の展望

今回紹介した方法は、まだ部分的な業務の使用にとどまっている。当社では、今でもSPOOLFで帳票出力しているものが多々あることもあり、新しい手法については、利用しているエンドユーザーの評価としては上々である。

そして今回の導入により、各プロセスのテンプレートを作成することができたため、今後の業務に展開していく予定である。

最後に

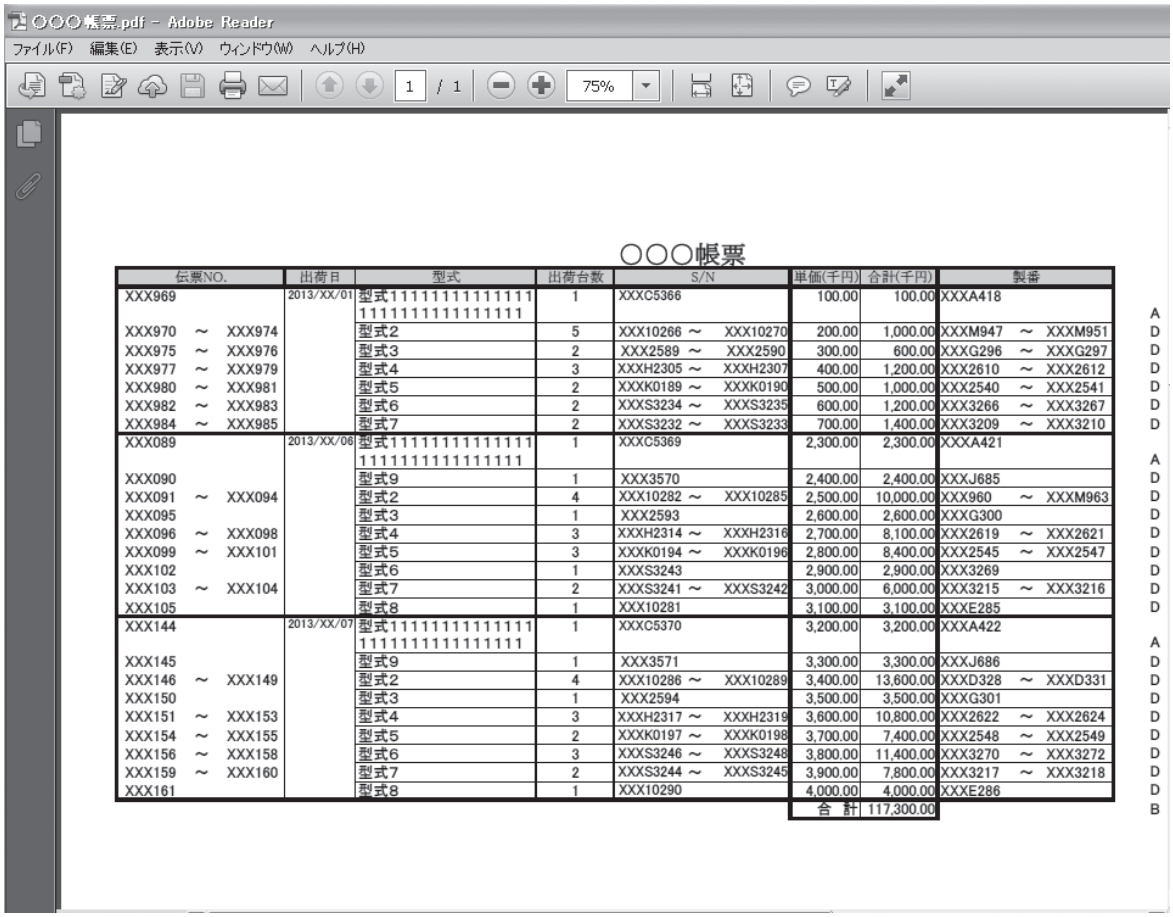
本稿を読んだ人は、今回紹介した方法により、IBM i ネイティブやネイティブ以外の資源の融合によってIBM i の資源をさらに有効に使えるという、1つの利用例があることが理解できたと思う。加えて、まだIBM i の資源を有効に使える可能性が潜んでいると考えられるため、今後もその可能性を模索していきたい。

ただし、問題は残っている。それはIBM i の情報は他のサーバーとは異なり、非常に入手しにくい点である。この状況を打開する1つの方法として、企業間を越えたナレッジの共有が考えられる。自企業から同地区の企業へ、そして全国の企業へとその輪を広めていくことが、今後の有効な手段の1つだと思われる。

ぜひ、本稿執筆をトリガーとして、全国の企業の方々とナレッジ共有を実現させて、Delphi/400およびDelphiとともに、今まで以上にIBM i を有効かつ効率的に活用していきたい。

M

図3



○○○帳票

伝票NO.	出荷日	型式	出荷台数	S/N	単価(千円)	合計(千円)	製番	
XXX969	2013/XX/01	型式1111111111111111111111111111111	1	XXXC5366	100.00	100.00	XXXA418	A
XXX970 ～ XXX974		型式2	5	XXX10266 ～ XXX10270	200.00	1,000.00	XXXM947 ～ XXXM951	D
XXX975 ～ XXX976		型式3	2	XXX2589 ～ XXX2590	300.00	600.00	XXXG296 ～ XXXG297	D
XXX977 ～ XXX979		型式4	3	XXXH2305 ～ XXXH2307	400.00	1,200.00	XXX2610 ～ XXX2612	D
XXX980 ～ XXX981		型式5	2	XXXK0189 ～ XXXK0190	500.00	1,000.00	XXX2540 ～ XXX2541	D
XXX982 ～ XXX983		型式6	2	XXXS3234 ～ XXXS3235	600.00	1,200.00	XXX3266 ～ XXX3267	D
XXX984 ～ XXX985		型式7	2	XXXS3232 ～ XXXS3233	700.00	1,400.00	XXX3209 ～ XXX3210	D
XXX089	2013/XX/06	型式1111111111111111111111111111111	1	XXXC5369	2,300.00	2,300.00	XXXA421	A
XXX090		型式9	1	XXX3570	2,400.00	2,400.00	XXXJ685	D
XXX091 ～ XXX094		型式2	4	XXX10282 ～ XXX10285	2,500.00	10,000.00	XXX960 ～ XXXM963	D
XXX095		型式3	1	XXX2593	2,600.00	2,600.00	XXXG300	D
XXX096 ～ XXX098		型式4	3	XXXH2314 ～ XXXH2316	2,700.00	8,100.00	XXX2619 ～ XXX2621	D
XXX099 ～ XXX101		型式5	3	XXXK0194 ～ XXXK0196	2,800.00	8,400.00	XXX2545 ～ XXX2547	D
XXX102		型式6	1	XXXS3243	2,900.00	2,900.00	XXX3269	D
XXX103 ～ XXX104		型式7	2	XXXS3241 ～ XXXS3242	3,000.00	6,000.00	XXX3215 ～ XXX3216	D
XXX105		型式8	1	XXX10281	3,100.00	3,100.00	XXXE285	D
XXX144	2013/XX/07	型式1111111111111111111111111111111	1	XXXC5370	3,200.00	3,200.00	XXXA422	A
XXX145		型式9	1	XXX3571	3,300.00	3,300.00	XXXJ686	D
XXX146 ～ XXX149		型式2	4	XXX10286 ～ XXX10289	3,400.00	13,600.00	XXXD328 ～ XXXD331	D
XXX150		型式3	1	XXX2594	3,500.00	3,500.00	XXXG301	D
XXX151 ～ XXX153		型式4	3	XXXH2317 ～ XXXH2319	3,600.00	10,800.00	XXX2622 ～ XXX2624	D
XXX154 ～ XXX155		型式5	2	XXXK0197 ～ XXXK0198	3,700.00	7,400.00	XXX2548 ～ XXX2549	D
XXX156 ～ XXX158		型式6	3	XXXS3246 ～ XXXS3248	3,800.00	11,400.00	XXX3270 ～ XXX3272	D
XXX159 ～ XXX160		型式7	2	XXXS3244 ～ XXXS3245	3,900.00	7,800.00	XXX3217 ～ XXX3218	D
XXX161		型式8	1	XXX10290	4,000.00	4,000.00	XXXE286	D
				合 計		117,300.00		B

シルバー賞

発注システムをVBからDelphiへ移植しリニューアル
―商品投入から店舗展開までの“見える”進捗管理システムを実現

川島 寛 様
株式会社タツミヤ
管理部 情報システム課

株式会社タツミヤ
http://tatsumiya1969.co.jp/

婦人服専門店として、北は北海道から南は沖縄まで、全国に約 400 店舗を展開。独自の情報システムの構築により、全店舗へのすばやいトレンド商品の供給を実現し、豊かなファッション文化を提案している。

1.発注システム開発の経緯

当社は、全国に約 400 店舗を有する婦人服の小売業を営んでいる。商品部がメーカーから商品を買い付け、商品の上代や下代を決定のうえ、買い付けた商品を各店舗に配分・供給する。この業務をサポートするシステムが「発注システム」である。

発注システムは、AS/400 のデータベースを使い、画面は VB で作られた画期的なシステムとして 12、3 年前にスタートした。しかし、Windows95 時代に VB で作成したシステムがベースのため、端末が Windows7 になった場合、起動できるかどうかの不安があった（実際、Windows7 のパソコンにインストールして確認したが、起動できなかった）。

VB のシステムを Windows7 に対応させるバージョンアップも検討したが、作成した開発会社に問い合わせたところ、システムの検証だけでも 1 千万円の費用がかかり、さらに、プログラム作成費用がそれに上乗せされるとの回答であった。

というわけで、Windows XP のパソコンがなくなるまでにはまだ時間的な猶予もあるので、VB システムを Delphi/400 に移植し、リニューアルする方法に挑戦することにした。

2.新システムへの要望

発注システムの改修作業にとりかかると、直ぐに商品部より新しい要望があった。それは、「検討中の発注」と「確定した発注」を明確に分けてほしいというものであった。

従来のシステムでは確定した発注しか入力していなかったが、実際にはメーカーまたは卸問屋に買い付けしたものの中で、発注システムに未登録のものが存在した。このため、突然確定した発注が出てきて、予算オーバーを招くことがしばしば発生した。この防止策として、買い付けに行ったときの発注内容を「検討中」と「確定」に分けて登録・表示することにより、それが現在どのような位置付けになっているかを把握できることが必要となっていた。

また、バイヤー（買付担当者）とアシスタントとの緊密な情報交換をとるための手段が発注システムにはなかった。そのため、買い付けの状況を記載したエクセルシートのシステムへの取り込みも求められた。

今回の開発は、単に VB から Delphi/400 への移植ではなく、商品投入から店舗展開までの一連の進捗を管理できるシステムとすることを目指した。

入力画面で仕入、売上、在庫の各状況を把握できるようにし、仕入予算に対して現在の受け払い状況と発注状況を合算したものを表示可能にした。

3.発注システム開発の課題と解決サポート

「VB からの移植」と題したが、実際の開発はまったく新しいシステムを構築していくようなものであった。

図1　仕入予定画面

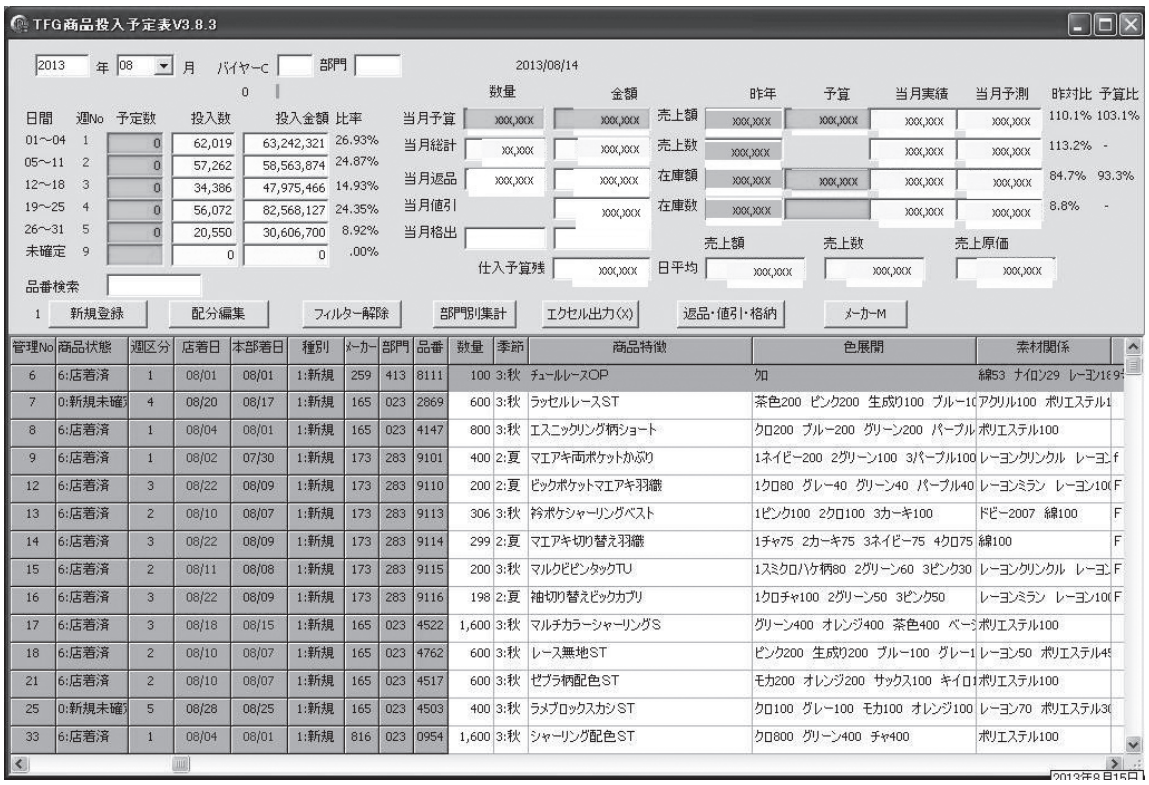


図2　仕入入力画面

「従来できていた機能がDelphiではできないのでなくします」では許されなかった。機能を失うことなく、さらに使い勝手のよさを追求していくために、ミガロ．サポートデスクに相談することがしばしばあった。

例えば、従来のVBシステムの入力画面では、表と組み合わせてコンボボックスを使うことができた。Delphi/400で多用したStringGrid自体にはコンボボックスの機能はないが、表のある項目に達したらコンボボックスを表示する方法について知りたかった。

サポートデスクに問い合わせ、回答を得ることで、技術的な問題を解決することができた。ミガロ．はサポート体制が充実しており、サポート担当の方には深く感謝している。

4.発注システムの改善内容

新しい発注システムでは、以下のようなさまざまな改善を行った。【図1】【図2】

・バイヤーへの情報提供

従来は、発注した商品は直接店舗に配送されることが普通だったが、最近は海外メーカーからの仕入れが増えた。一度本部に商品を送り、本部から店舗へ商品を発送するというパターンが増え、物流の形態が変わってきた。

これに伴い、バイヤー（買付担当者）が納期情報、発送状況、メーカーへの返品、値引、売上状況、6か月先までの仕入状況を把握できるようにした。

・マスター情報

発注システムは商品情報の入り口となる。マスターデータを正確に区分して登録できるようにした。

・商品画像データの登録

商品画像を、デジカメからファイルシステムにスムーズに保管できることが要求された。

VBにはドラッグ＆ドロップで画像をコピー＆ペーストできる機能があるが、Delphiにも Imageコンポーネントで画像を貼り付ける機能があり、これを利用した。

また、画像関係ではデジカメのデータを直接サーバーに保管するのではなく、保管前にデータを圧縮するようにした。画像を圧縮するコーディング方法については、ミガロ．サポートに助言を得た。

・商品の色による分類

今回、商品の色についても検討した。色コードは、商品分類のキー項目とせず、商品の情報項目として色展開がわかるような登録方法とした。

理由は、色コードで商品を分類した場合、商品マスターのレコード数が増加しすぎること、および色に対応した区分け作業が発生し業務に混乱をきたす可能性があったことによる。このため、色によるコード化はあえてせず、色展開を商品情報として見せる形にとどめた。

・発注状況管理

次に、発注システムで店舗に商品を供給するまでの「状況」を説明する。発注システムに「状況コード」を追加し、以下のように意味付けをした。【図3】

0 仕入検討中：商品が投入される
1 発注入力へ追加：予算範囲内であれば発注登録
2 配分準備：配分編集画面で店舗への配分を検討
3 配分：配分が決定
4 削除：取り消したい場合に選択
（状況が0、1、2のとき取消可能）
5 伝票発行：伝票発行レコード
6 店舗へ：レコードが店舗に渡り仕入が発生する

※0と1の違い：0には発注番号がなくブランクである。1になると発注番号ができる。

・納入区分管理

次に納入区分について説明する。納入については、前述したように従来はメーカー直送のケースが大半であったが、海外メーカーとの取引の増大やその他の事情により、一度本部に納入し本部から各店舗に商品を発送するというケースが出てきた。

納入方法は、以下の3通りのケースにわかれる。

1：経由
（事前に伝票とプライスをメーカーに送っておき、商品にセットして本部に納入。その後、本部から店舗に発送）

2：納品
（商品が本部に到着後、本部で店舗に配賦する伝票を作成し、プライスを付けて発送）

3：直送
（商品をメーカーから直接店舗に発送）

例えば「2：納品」の場合、本部での納品作業が発生し、作業人員の確保が必要となる。作業予定を立てられるようにするため、本部への「納品日付」と店舗への「商品到着日付」を設定した。また、メーカーより納期が確定した場合は「納期確定」の表示を可能にした。

「納入区分集計のアウトライン」では、各ファイルの関連を示している。【図4】

5.システムの効果と今後の展望

発注システムは、全社システムの上流部分に当たる。そのため、新システムによりデータが正確に区分されて登録されることにより、他のシステムに対しても非常に有益な改善となった。

例えば、店舗システムへの画像や商品情報の供給がスムーズになった。さらに、発注システムの入力部分の完成により、全社でのデータ共有化が促進されてきた。仕入のワークフローでは、発注システムから店舗システムとの関連づけを行っている。

今後の展望としては、従来の発注システムはバイヤーのみのためのシステムと見られていたが、新システムではバイヤーのみではなく、アシストする担当者との発注情報共有がさらに重要になってきている。現在は入力部分が完成したところだが、今後、出力系などの周辺モジュールや、予算管理・配分の仕組みの改善を図っていく予定である。

M

図3　発注入力から伝票発行までの流れ

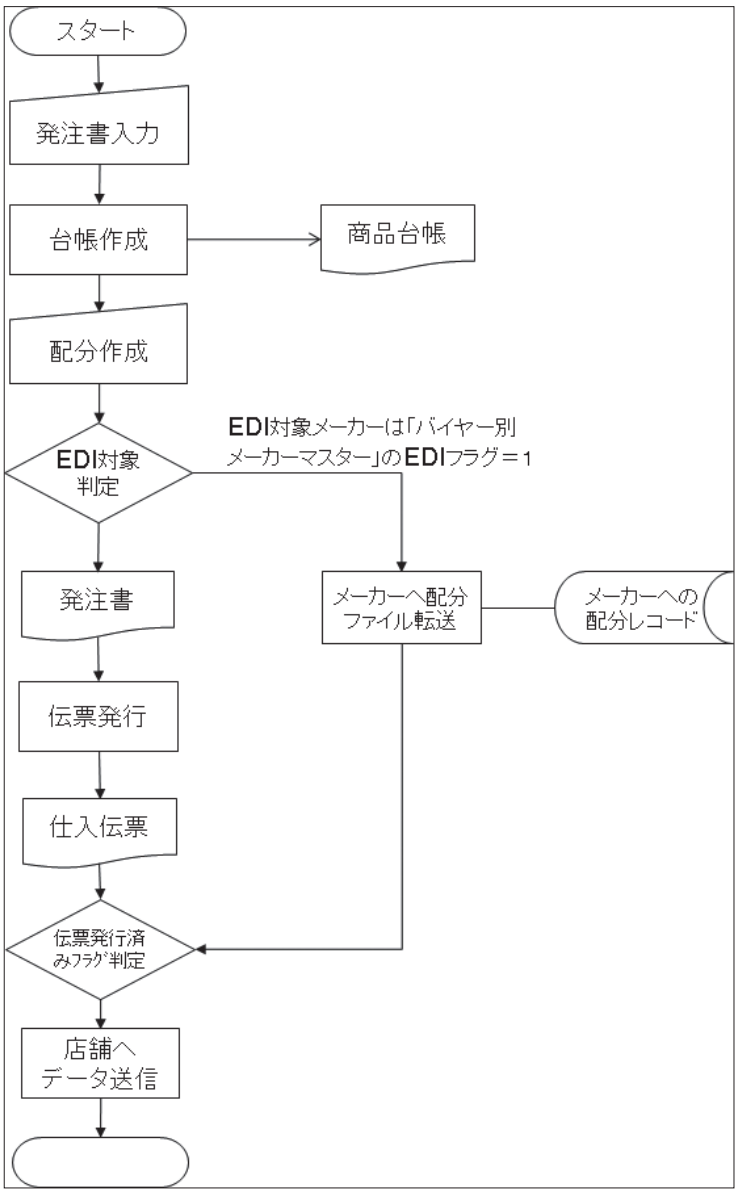
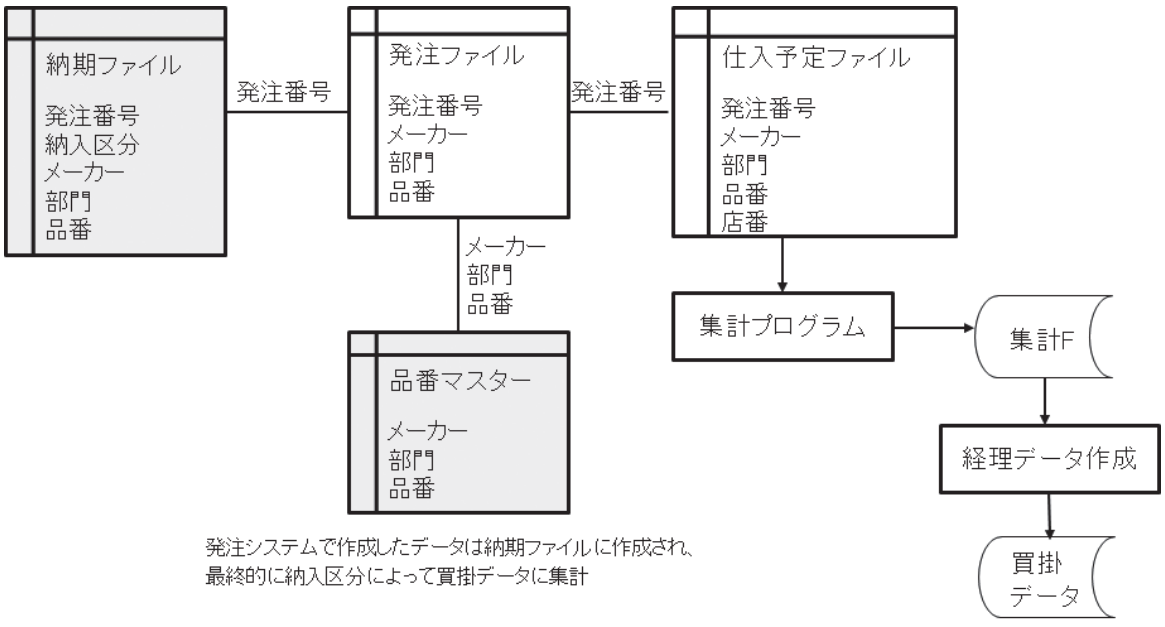


図4　納入区分集計のアウトライン

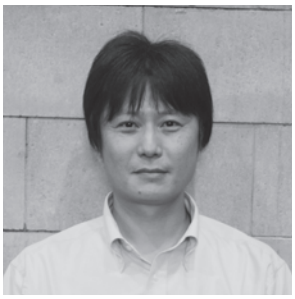


発注システムで作成したデータは納期ファイルに作成され、
最終的に納入区分によって買掛データに集計

優秀賞

5250画面を使用せずに AS/400 スプールファイルをコントロールする

白井 昌哉 様
太陽セメント工業株式会社
経理部 業務推進室 部長



太陽セメント工業株式会社
http://www.taiyo-cement.co.jp/

建築用ブロックや舗装材など、さまざまな用途に使用されるコンクリートブロックの製造を行っている。業界におけるパイオニア企業として、排気ガスを浄化するブロック、緑化ブロック、個性的で美しいブロックなど、常に時代が求めるニーズを読み取りながら製品を開発している。

業務課題

社内の受注出荷業務を Delphi/400 によりシステム化したが、日々発行する出荷伝票を出力する時や用紙の差し替え時に、5250 画面を立ち上げメッセージに応答する作業が社内に浸透しなかった。

毎日夕方になると 10 か所ある拠点から「伝票が出ない」などの電話があり、日々対応に追われていた。

業務担当者からは、Delphi 画面と 5250 画面を並行的に使うことに対する抵抗もあった。

技術課題

Delphi で、5250 画面のコマンド「WRKOUTQ xxxOUTQ」の照会結果のような画面を作成して、印刷ボタンをクリックすると、自拠点の印刷待ち行列を対象として、自動的に AS/400 がメッセージに応答して帳票を印刷する、というシステムを作る計画を立てた。

技術的には、以下が課題となった。

【課題 1】

拠点ごとに OUTQ を作成したが、どの拠点が、どの OUTQ 情報を照会するか。

【課題 2】

WRKOUTQ 照会画面のようにいったん画面が止まり、業務担当者が印刷待ち行列を目視確認し、次のアクションで自動的に印刷をさせるためには、どのような工夫が必要か。

技術課題の解決策

ListSpool400 コンポーネントを利用し、課題を解決した。

【課題 1】

業務担当者のログインユーザー ID に対して OUTQ を紐づけするテーブルを、AS/400 上に作成して、ListSpool400 の OUTQ パラメーターにセットすることで実現可能となった。

【課題 2】

WRKOUTQ のような照会画面は、ListSpool400 を Open し、ListSpool400 の必要項目を DBGrid にフィールドを割り当て、目視確認ができるようにした。

印刷ボタンのクリック時に、ListSpool400 に各行（各待ち行列）を 1 レコードずつ読み込み、Call400 で CL プログラムを起動し、パラメーターを送り CL プログラム内のコマンド CHGSPLFA にパラメーターをセットすることにより、印刷装置への転送が可能となった。

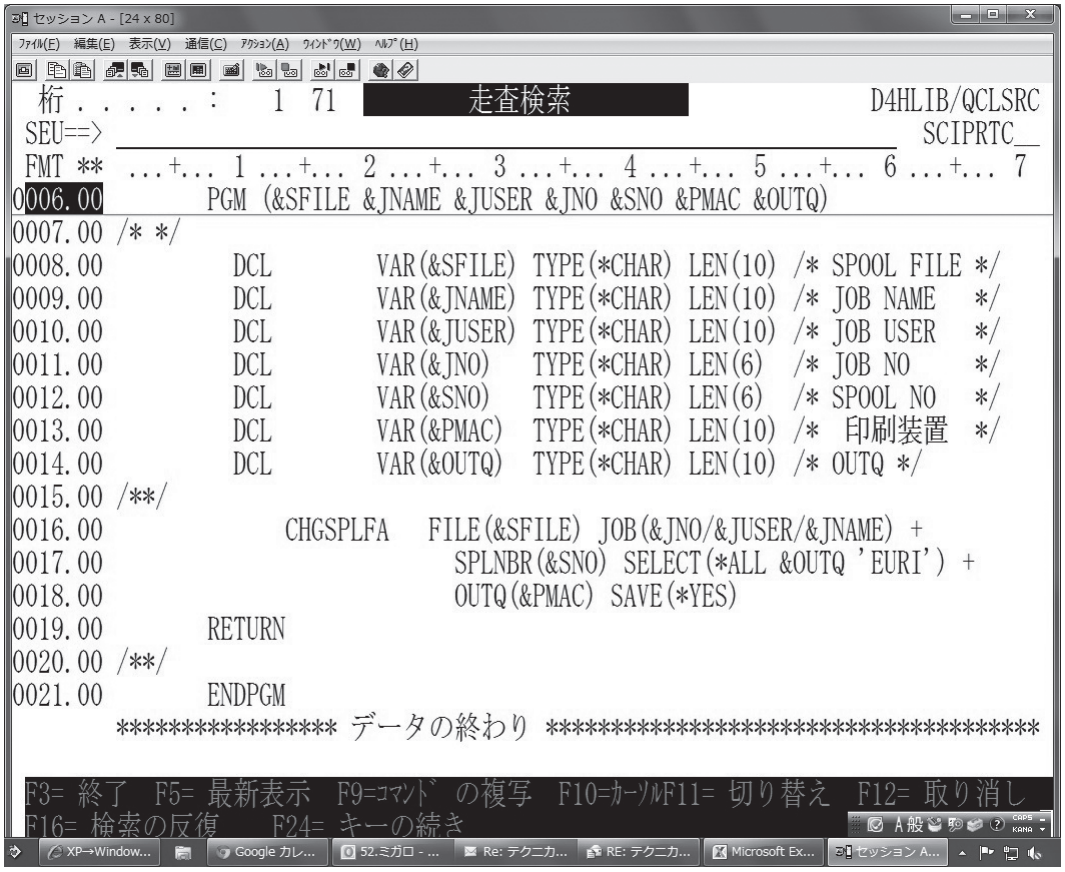
スプールのメッセージは、あらかじめ AS/400 で、コマンド WRKRPYLE でメッセージ自動応答を組み込んでおく必要がある。【図 1】【ソース 1】【ソース 2】

業務課題の解決と効果

まず、各拠点の業務担当者から日々の電話がなくなった。業務担当者は、プリンターに対し用紙の差し替えのみ行うという単純作業で帳票が印刷できる。

図1　売上伝票印刷状況画面

ソース1　CLプログラム



この効果がシステム利用全体に影響し、「コンピュータはわからない」という担当者レベルの思い込みを払拭でき、業務担当者からシステムに対する新たな提案がされるようになった。

結果、システム全体が徐々にバージョンアップしてきていることは、システム管理者からしてみれば、大きな喜びであり、今後の開発にも力を注ぐことができる。

Delphi/400 でのシステム開発を通じ、今まで 5250 画面ではできなかったことができるようになった点も多数あり、Delphi/400 を選択してよかったと思っている。

M

ソース2　Delphiソース

```
//画面表示
procedure TfrmSpoolList.acDisplayExecute(Sender: TObject);
begin
  inherited;
  with qrySCMCOD  do
  begin
    Close;
    ParamByName('SCDSCC').AsString := dmMain.LoginInfo.SCCODE;
    Open;
    if not(Eof and Bof) then
    begin
      ListSpool4001.OutQName:=Trim(qrySCMCOD.FieldByName('SCDCVL').AsString);
ListSpool4001.LibraryName:=Trim(qrySCMCOD.FieldByName('SCDCR1').AsString);
PRTName := Trim(qrySCMCOD.FieldByName('SCDCR2').AsString);
    end;
  end;
  //SPOOL-LIST
  cdsSpoolList.Open;
end;
//伝票発行
procedure TfrmSpoolList.acDenPrintExecute(Sender: TObject);
var
  p_sFile,p_jNAME,p_jUSER,p_jNO,p_sNO,p_MAC,p_OUTQ:String;
  p_sNOw:integer;
  PrtFrm: TForm;
  PrtFont: TFont;
begin
  inherited;
  // 指定用Font設定
  PrtFont := TFont.Create;
  PrtFont.Color := clRed;
  PrtFont.Size  := 15;
  PrtFont.Style := [fsItalic];
  //用紙替のメッセージ
  messageBeep(MB_ICONEXCLAMATION); //警告音
  // 作成関数によるメッセージ表示
  PrtFrm := CreateFontMessageDialog('！プリンターの用紙を【売上伝票】に取り替えてください。', mtInformation, [mbOK, mbCancel], PrtFont);
  if PrtFrm.ShowModal = mrOK then
  begin
    if cdsSpoolList.RecordCount > 0 then
    begin
      with cdsSpoolList do
      begin
        Open;
        First;
        while not(Eof) do
        begin
          //フィールドセット
          p_sFile:=Trim(cdsSpoolListName.AsString);
          p_jNAME:=Trim(cdsSpoolListJobName.AsString);
          p_jUSER:=Trim(cdsSpoolListUserName.AsString);
          p_jNO  :=copy(cdsSpoolListJobNumber.AsString,1,6);
          p_sNO  :=copy(cdsSpoolListSpoolFileNumber.AsString,1,6);
          p_snow :=StrToInt(p_sNO);
          p_sNO  :=IntToStr(p_sNOw);
          p_MAC  :=Trim(PRTName);
          p_OUTQ :=Trim(cdsSpoolListOutqName.AsString);
          // 保存CLの呼び出し
          with call4SCIPRTC do
          begin
            Value[0] := p_sFile;
            Value[1] := p_jNAME;
            Value[2] := p_jUSER;
            Value[3] := p_jNO;
            Value[4] := p_sNO;
            Value[5] := p_MAC;
            Value[6] := p_OUTQ;
            Execute;
          end;
          //次のレコード
          Next;
        end;
        First;
      end;
    end;
  end;
```

優秀賞

Delphi/400を利用した承認フロー導入によるIT内部統制構築

塚本 圭一 様

ライオン流通サービス株式会社
企画部 企画チーム

ライオン流通サービス株式会社
http://www.lion-logi-s.co.jp/

1963 年、ライオン株式会社 100% 出資の物流子会社として設立。ライオングループにおいて、グループ製品の物流全般を担っており、効率化および物流品質の向上を目的に掲げ合理化を推進している。

業務課題

物流管理会社である当社は、各運送会社との契約情報に基づき、運送運賃を計算している。

しかし、契約情報を IBM i に登録する際、入力担当者が登録を誤ることにより、運送運賃を誤算出するなどの問題があった。

そこで、契約情報の登録を入力担当者まかせにせず、入力内容の確認や承認申請に関して、その承認行為の進捗管理や承認依頼の到着を知らせる業務通知機能の実現が必要となった。これは、内部統制上も求められていた。

技術課題

IBM i で稼働するワークフローが検討された。ソフトの導入も検討したが、他のワークフローソフトへの移行が必要となった際の情報の継承性が懸念される。そのため、IBM i のワークフローシステムを自社で構築することを検討した。

しかし、CL 言語や RPG 言語だけでは開発が難しい。業務通知機能についても、メッセージ通知のみでは承認依頼が来たことがわからない可能性もあり、実現が困難だった。

また、承認者にあたる人が、必ずしもエミュレーター画面に馴染みがあるとは限らないので、ビジュアルインターフェースも要求された。

技術課題の解決策

当社では、IBM i に蓄積された物流データの「見える化」を実現させるために、すでに Delphi/400 を導入していた。そこで、Delphi/400 を用いた簡易ワークフローソフト構築の検討を進めた。

具体的には、フローの順序や誰に申請をするか等を制御するためのテーブルを構築し、Delphi/400 での簡易ワークフローの実現に成功した。

メール通知機能に関しても、コーディングが容易であったため、業務通知機能としてメール送信機能を搭載した。メール本文にリンク先として、実行ファイルを記述することにより、次のステップへさらに効率よく業務が進む仕様とした。【ソース 1】

業務課題の解決と効果

申請行為や承認行為に関して、簡易ワークフローの構築を実現した。

その結果、入力担当者まかせになっていた運送運賃の契約情報登録に関して、内部統制をとれるようになった。

また、業務通知のためのメール送信機能を実現したことにより、承認作業に遅れが出ることもなく、進捗管理もできることから遅延のない業務が実現できた。

さらに、メニュー構成や登録画面が Windows ライクな画面となることで、エミュレーター画面に不慣れな人でも、操作説明がなくとも使用することが可能となった。【図 1】【図 2】

M

ソース1　メール送信

```
//仮登録申請実行（メール送信）
SMTP := TIdSMTP.Create(nil);
try
  SMTP.Host     := 'EXSV13.RYUTSU.LOCAL';
  SMTP.Port     := 25;
  SMTP.Connect;
  Msg := TIdMessage.Create(SMTP);
  try
    Msg.OnInitializeISO          := IdMessage1.InitializeISO;
    Msg.ContentType              := 'text/plain';
    Msg.CharSet                  := 'ISO-2022-JP';
    Msg.ContentTransferEncoding  := 'BASE64';              // BASE64 エンコーディング
    Msg.From.Name                := Edt_USNM.Text;         // 送信者の氏名
    Msg.From.Address             := Edt_TMAIL.Text;        // 送信者のメールアドレス
    Msg.Recipients.EMailAddresses := Edt_SMAIL.Text;       // 送信先のメールアドレス
    Msg.Subject                  := '申請処理依頼';
    Msg.Body.Text                := '遠距離輸送運賃仮登録分の申請処理をお願い致します。' + #13#10 +
                                    '申請処理は下記リンクをクリックし、「開く」を押して実行して下さい。' + #13#10 + #13#10 +
                                    '<FILE://C:¥Program Files¥ＳＰＬＵＳ¥SPL0145_SE.EXE>';
    SMTP.Send(Msg);
  finally
    Msg.Free;
  end;
  SMTP.Disconnect;
finally
  SMTP.Free;
end;
```

図1　メニュー画面

図2　契約登録画面

Migaro.
Technical Report
2013

ミガロ.SE 論文／ミガロ. テクニカルレポート

尾崎 浩司

株式会社ミガロ.

RAD事業部 営業推進課

FastReportを使用した帳票作成入門

帳票作成ツール「FastReport」がバンドルされた XE3。
スプールのような桁数制約がなく、Delphi/400 から
直接レーザープリンターに綺麗な帳票が出力できる。

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
2013 年 04 月 RAD 事業部配属

現在の仕事内容
ミガロ. 製品の素晴らしさをアピールするためのセミナーやイベントの企画・運営等を主に担当している。

はじめに

Delphi/400 VersionXE3 には、新たに帳票作成ツールとして「FastReport」がバンドルされた。IBM i(AS/400)ユーザーであれば、アプリケーションの帳票出力に「OUTQ」（スプール）を使用した帳票が一般的だと思うが、FastReport を使用すれば、スプールのような桁数制約のある帳票でなく、Delphi/400 から直接レーザープリンターに制約のない綺麗な帳票を出力するプログラムが作成可能となる。

今回は、この FastReport を使用した帳票作成手順をご紹介したいと思う。

FastReportの特徴

FastReport は、露 Fast Reports 社が開発した帳票出力用のアドオンコンポーネントである。VCL（Visual Component Library）として使用できるため、ほかのコンポーネント同様、ツールパレットから部品を貼り付けるビジュアル開発が可能である。【図 1】

コンポーネントとして帳票作成が可能なため、帳票出力ライブラリや帳票デザインを実行モジュール（Exe ファイル）に含めることが可能である。つまり、アプリケーション実行のための特別なランタイムやレイアウトファイルが不要になるので、PC 環境の影響を受けにくい帳票が作成できる。

また、帳票デザインは、専用のレポートデザイナを使用し、Delphi におけるフォーム設計同様にビジュアル設計が可能で、画像の挿入やグラフ、バーコード等の表現力豊かなレポートを容易に作成できることも特徴である。

さらに、作成した帳票は、プレビューならび印刷はもちろん、外部ファイルへの出力もサポートされているため、PDF ファイルの作成も容易に実現できる。

帳票作成手順

まず初めに FastReport を使用した簡単な帳票作成手順をご紹介する。画面に入力した値を帳票プレビュー表示する、という簡単なアプリケーションである。【図 2】

・TfrxReport コンポーネント

FastReport で帳票出力するために必ず使用するのが、TfrxReport コンポーネントである。【図 3】

貼り付けた frxReport1 をダブルクリックすると、レポートデザイナ画面が開く。レポートデザイナ画面は、主に次のような画面構成となっている。【図 4】

①デザイナ：帳票用コンポーネントを貼りつけてレイアウトを作成
②コンポーネントパレット：帳票用コンポーネントを格納
③レポートツリー：帳票の構造をツリー形式で表示
④オブジェクトインスペクタ：帳票用コンポーネントのプロパティを設定

このように FastReport は、Delphi

図1

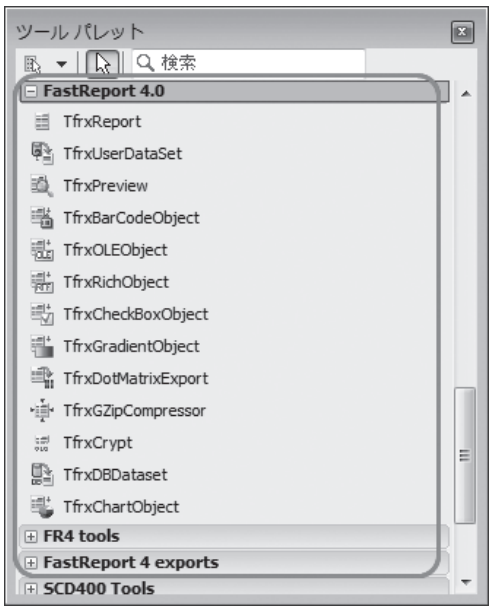

ツール パレット
検索
FastReport 4.0
TfrxReport
TfrxUserDataSet
TfrxPreview
TfrxBarCodeObject
TfrxOLEObject
TfrxRichObject
TfrxCheckBoxObject
TfrxGradientObject
TfrxDotMatrixExport
TfrxGZipCompressor
TfrxCrypt
TfrxDBDataset
TfrxChartObject
FR4 tools
FastReport 4 exports
SCD400 Tools

図2

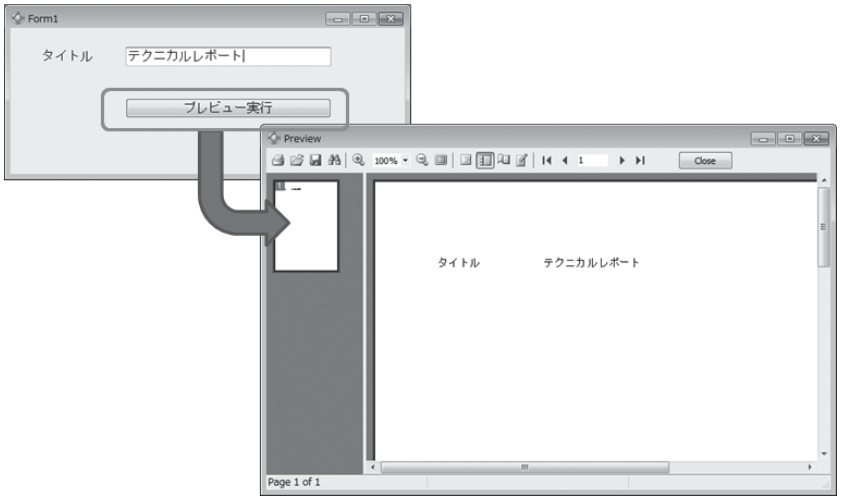
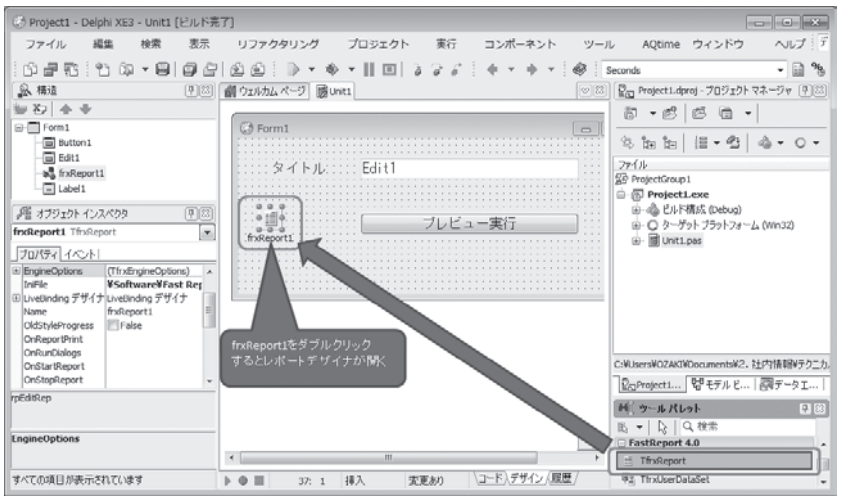

Form1
タイトル
テクニカルレポート
プレビュー実行
Preview
Close
タイトル
テクニカルレポート
Page 1 of 1

図3


Project1 - Delphi XE3 - Unit1 [ビルド完了]
Form1
タイトル:
Edit1
プレビュー実行
frxReport1
frxReport1をダブルクリック
するとレポートデザイナが開く
TfrxReport

の統合開発画面（IDE）同様の手法で設計することができる。

・帳票レイアウト設計

では実際の設計だが、今回は「タイトル」と書かれた表題と、画面上のEdit1コンポーネントに指定された文字列（変数）を出力する2つのコンポーネントを使用する。

具体的には、ともにText Objectコンポーネントを貼りつける。Text Objectコンポーネントをデザイナに貼りつけると、出力する文字列を編集するウィンドウが開く。ここに出力する文字列を指定すればよい。この文字列を編集するウィンドウでは、単なる文字列だけでなく、変数等も指定可能である。変数を指定する場合、Insert Expressionボタンを押下し、表示されたウィンドウのVariablesタブにあるExpression欄に変数名を記載すればよい。【図5】

コンポーネントを貼りつけて文字列を定義したら、あわせてフォントの指定も行おう。オブジェクトインスペクタからFontプロパティを選択し、日本語フォント（例：MSゴシック）を選択すればよい。このとき文字セットを「日本語」に指定することがポイントになる。

一通り帳票レイアウトの設計が完了すれば､そのままレポートデザイナを「×」ボタンで終了する。（ここではファイル保存は不要）

これで帳票レイアウト設計は終了である。後は､フォーム上のボタン（Button1）を押下したときに帳票出力するロジックを記述すればよい。【ソース1】

プレビューの表示は、ShowReportメソッドを実行する。そしてプログラムで変数にセットするのは、Scriptプロパティの Variablesプロパティに値をセットすればよい。たったこれだけで、図2のような帳票出力プログラムが作成できる。

データベースを使用した帳票作成

帳票の基本的な作成手順は説明したが､実際の帳票作成では､主にデータベースから取得した値を出力項目としてセットすることが多い。では、FastReportでデータベースを扱う帳票はどのように作成するかを説明する。

・TfrxDBDataSetコンポーネント

データベースから取得した値（データセット）を使う帳票では、TfrxDBDatasetコンポーネントを使用する。画面に貼りつけたfrxDBDataset1のDataSetプロパティに、出力したいデータセットを指定すればよい。【図6】

データセットの準備ができたら、先ほどと同様にfrxReport1をダブルクリックし、レポートデザイナ画面を立ち上げる。メニューバーより［Report｜Data］を選択すると先ほどフォーム画面に貼りつけたfrxDBDataset1が選択できるので、選択してOKを押下する。すると、レポートデザイナ画面の右側にあるData Treeに､選択したデータセットの項目（フィールド）が一覧表示される。【図7】

これで、帳票でデータセットが利用できるようになる。

・データセットの帳票出力

次に、定義したデータセットを帳票に出力する手順を説明する。FastReportでは、帳票をヘッダー、明細、フッター等のエリアごとにバンドと呼ばれるものを作成し、出力項目を設定する。

具体的には、コンポーネントパレットよりバンドコンポーネントを選択し、PageHeaderとMaster Dataを貼りつける。【図8】

Master Dataを貼りつけた際には、使用したいデータセットを選択するダイアログ画面が表示されるので、frxDBDataset1を選択する。

これで準備が整ったので、後は貼りつけた各バンドに出力項目を定義していく。PageHeader部には、Text Objectsを貼りつけて列タイトルを定義する。そして、Master Data部には、レポートデザイナ画面右側のData Tree部から出力項目を定義すればよい。帳票設計結果は、都度レポートデザイナ上でプレビューすることも可能である。【図9】

一通り帳票設計ができたので、後はプレビュー処理（ShowReportメソッド）を記述すれば完成である。【図10】

このように、データセットを使用した帳票も簡単に生成できることがわかる。

データベース帳票の応用例

データセットを使用する帳票の作成手順を見てきたが、ここからは帳票レイアウト作成の応用例を説明する。

前述の作成した帳票は「気象官署ごとの年間降水量一覧」である。この帳票をさらに拡張していこうと思う。

・地域別：計算項目の出力

まず、各地域の気象官署ごとに、年ごとの年間降水量についてその平均値を右側に追加する。【図11】

これは簡単である。Master Dataバンド部に、新たにText Objectを追加すればよい。このText Objectには、先ほど紹介した固定テキストや変数以外に計算式も追加できる。

今回の場合、各年の年間降水量を合計したものを8で割り、結果を整数値に置き換えるという計算式を記載すればよい。【図12】

・年別：データベースの集計

次に、全国の年別平均降水量を出力できるように実装する。【図13】

これは、フィールドごとの全レコードの平均値を取ればよい。最終レコード出力後に情報を出力するため、新たにバンドをFooterとして追加する。そして追加したFooterバンドにText Objectを追加すればよい。

ここでは、Insert Aggregateボタンを押下し、Function欄に平均を表すAVGを選択し、集計対象となるバンドとデータセットおよびフィールドを選択する。【図14】

これで、最終レコードに平均値が表示されるようになる。

なお、今回は全レコードの集計だったため、Footerバンドを用いた実装としたが、これ以外にも、GroupHeaderバンドおよびGroupFooterバンドを使用すると、合計だけでなく、小計や中計を出力したり、改ページ条件を指定した帳票も作成可能である。

・1000ミリ未満：条件指定の出力

3つ目の応用例は、出力項目に対する

図4

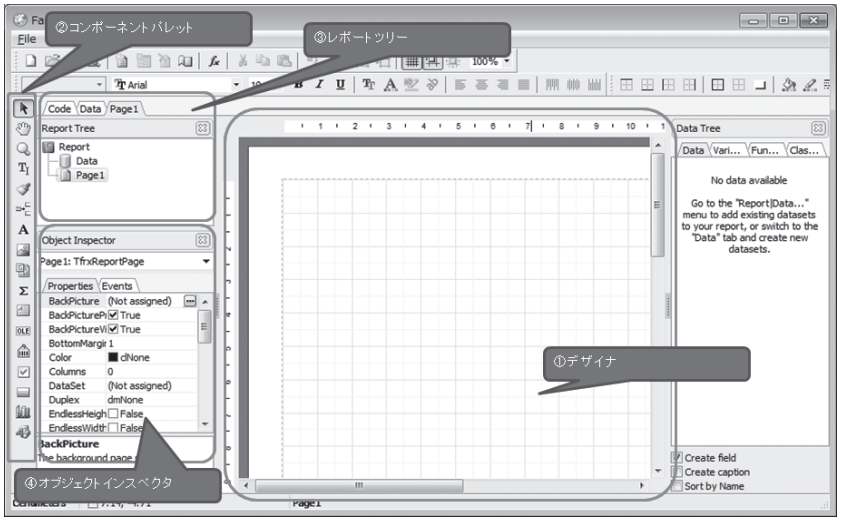
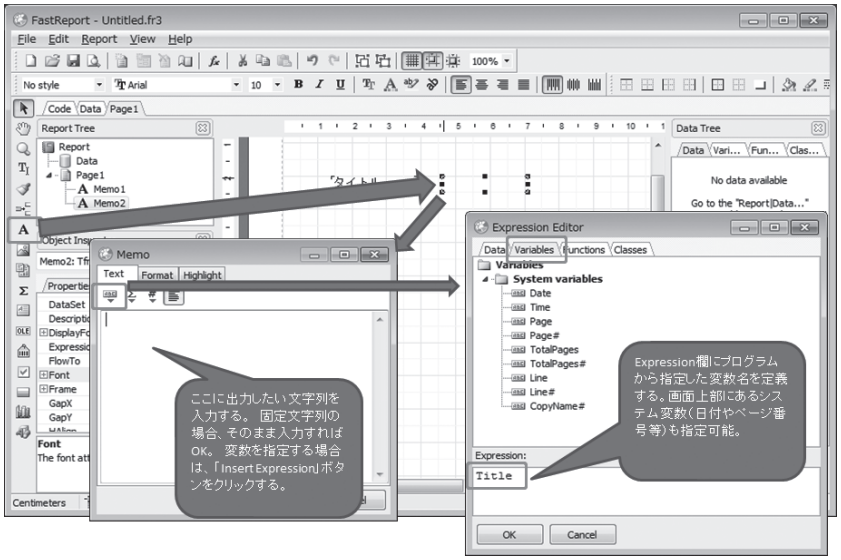


図5



ソース1

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  //画面の入力値を帳票にセット
  frxReport1.Script.Variables['Title'] := Edit1.Text;

  //帳票プレビューを表示
  frxReport1.ShowReport;
end;
```

条件指定を紹介する。年間降水量が少ない（例えば今回は、1000ミリ未満）箇所を目立つようにしていく。【図15】

これは、出力対象とするText Objectコンポーネントに、出力条件指定を付加することで実現可能である。処理対象のText Objectコンポーネントを選択し、表示される編集ウィンドウよりHighlightタブを開く。ここに条件が指定できるので、データベースの値が1000未満の場合に背景色に設定すればよい。【図16】

今回は、応用例として計算項目の出力方法、データベースの集計方法ならびに条件指定を行う出力方法をご紹介した。ほかにも多様な設計が可能である。

作成した帳票の出力方法

ここまでは、データベースを使用した帳票作成手順を紹介したが、次に作成した帳票を出力する方法を説明する。

・プレビュー画面

これまで、標準のプレビュー画面を表示していたが、このプレビュー画面上部には、いろいろな機能ボタンがついている。大変便利だと思うが、場合によってはボタンを表示させたくない場合もある。

実はプレビュー画面に表示させたいボタンは、TfrxReportコンポーネントのPreviewOptionsプロパティのサブプロパティである、Buttonsプロパティですべて制御可能である。これにより、例えばプレビューのみ許可して、印刷は不可といったような制御もできる。【図17】

さらに、TfrxPreviewコンポーネントを使用すると、フォーム上に独自のプレビュー機能を追加することもできる。

・pdfファイル出力

最後に、帳票を外部ファイルに出力する方法を説明する。Fast Reportではさまざまな形式の外部ファイルに帳票を出力することができるが、一番利用するのはやはりpdfファイルだと思う。

pdfファイルへ出力する場合、TfrxPDFExportコンポーネントを貼りつければよい。【図18】

実はこれだけで、PDF出力が実現できる。出力ロジックは、TfrxReportコンポーネントのExportメソッドを呼び出せばよい。【ソース2】

完成したプログラムを実行し、Button2として定義した「PDF出力」ボタンを押下すると、PDFを出力するための条件指定を行う画面が表示される。【図19】

条件指定画面を見るとわかるが、例えば、パスワード付のPDFファイルの作成等いろいろと設定できることがわかる。

ただし、とても便利だと思うが、見ての通り条件指定画面はすべて英語表記となるので、この画面を表示させたくない場合は、TfrxPDFExportコンポーネントのShowDialogプロパティをFalseにすればよい。TfrxPDFExportコンポーネントでは、条件指定画面で設定可能な項目がすべてプロパティで指定できるため、予め条件をプロパティに設定しておけばよい。

このように、PDF出力が簡単に行えるのもFastReportを使用するメリットである。

最後に

今回はDelphi/400で帳票作成する手法の1つとして、FastReportを紹介した。この帳票ツールを使用すると、画面のGUI化同様、表現力豊かな帳票も容易に設計・開発できる。また今回は紹介できなかったが、バーコードやグラフ（チャート）の埋め込みも容易に実装可能である。

最後にこのFastReportであるが、今回はDelphi/400 VersionXE3に付属のバンドル版を使用したが、帳票デザイン内でのイベントの利用といった高度な機能や、過去に作成したQuickReportおよびRaveReports等の帳票ツールの帳票デザインからのコンバートが利用可能な上位版（製品版）もある。FastReportのフル機能を使用したい方は、ぜひこの上位版の導入も検討いただければ幸いである。

M

図6

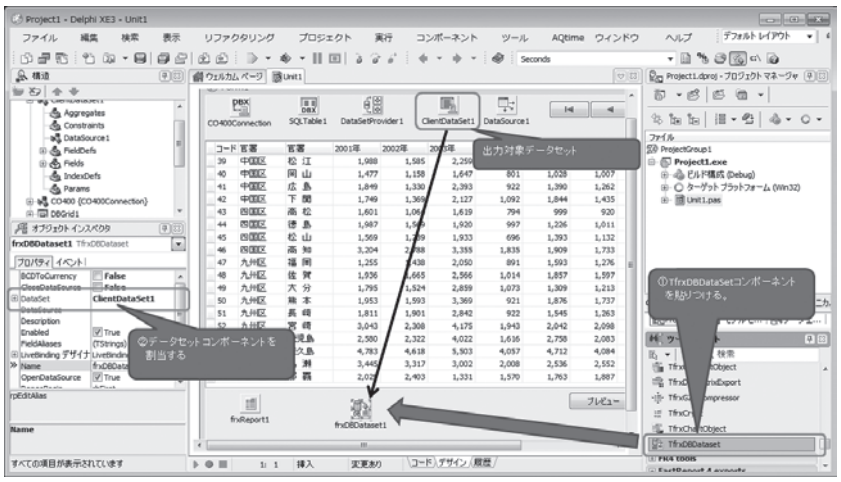

出力対象データセット
①TfrxDBDataSetコンポーネントを貼りつける。
②データセットコンポーネントを割当する

図7

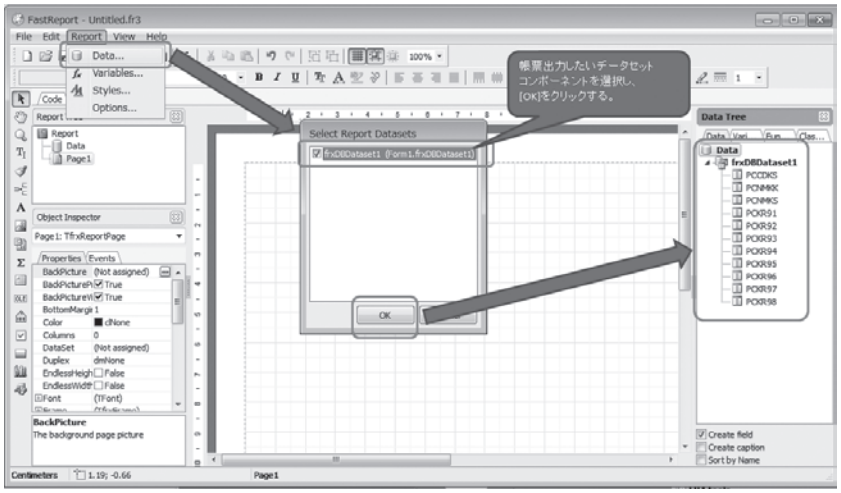

帳票出力したいデータセットコンポーネントを選択し、[OK]をクリックする。
Select Report Datasets
OK

図8

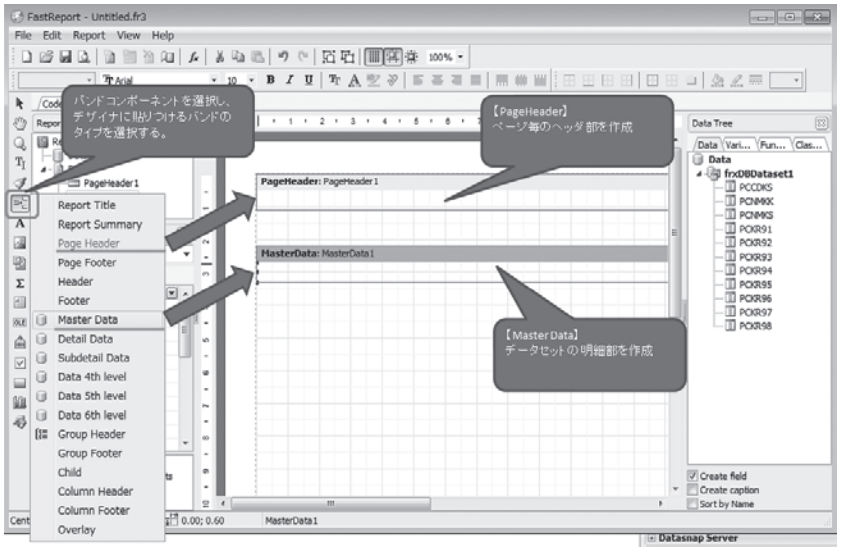

バンドコンポーネントを選択し、デザイナに貼りつけるバンドのタイプを選択する。
【PageHeader】
ページ毎のヘッダ部を作成
【MasterData】
データセットの明細部を作成
Report Title
Report Summary
Page Header
Page Footer
Header
Footer
Master Data
Detail Data
Subdetail Data
Data 4th level
Data 5th level
Data 6th level
Group Header
Group Footer
Child
Column Header
Column Footer
Overlay

図9

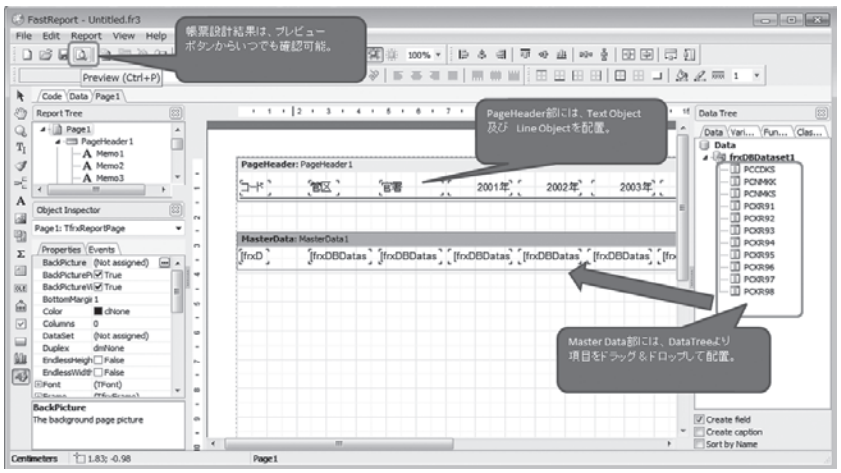

帳票設計結果は、プレビュー
ボタンからいつでも確認可能。
PageHeader部には、Text Object
及び Line Objectを配置。
Master Data部には、DataTreeより
項目をドラッグ＆ドロップして配置。

図10

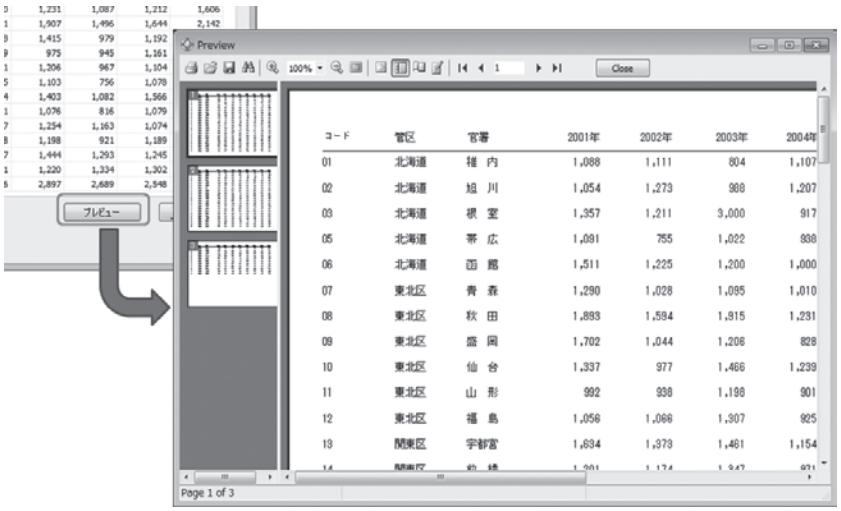

Preview
Close
プレビュー
Page 1 of 3

図11

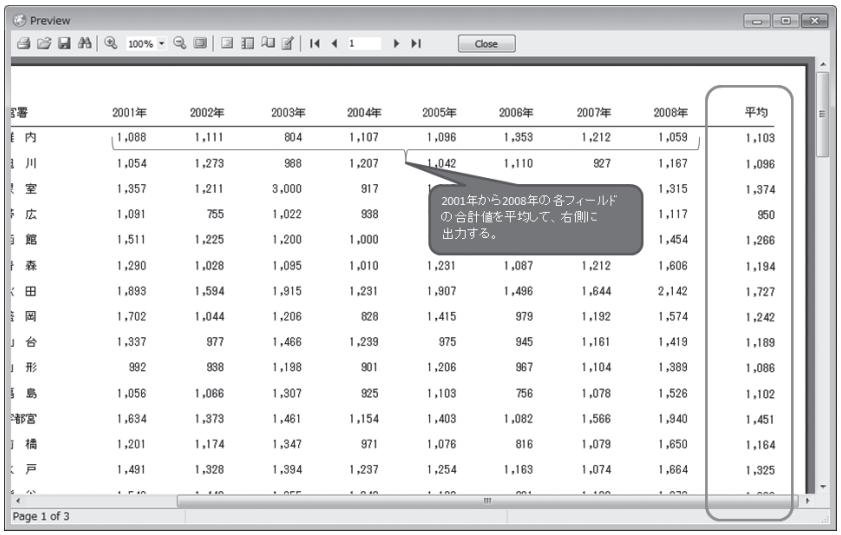

Preview
Close
2001年から2008年の各フィールド
の合計値を平均して、右側に
出力する。
Page 1 of 3

図12

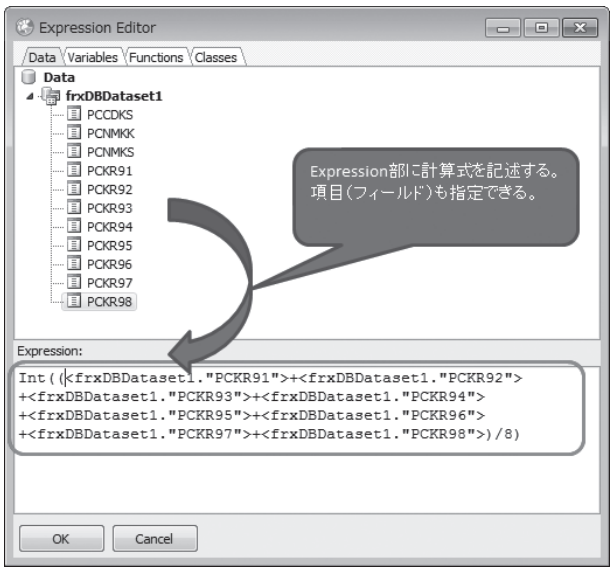

Expression Editor
Data
Variables
Functions
Classes
Data
frxDBDataset1
PCCDKS
PCNMKK
PCNMKS
PCKR91
PCKR92
PCKR93
PCKR94
PCKR95
PCKR96
PCKR97
PCKR98
Expression部に計算式を記述する。
項目(フィールド)も指定できる。
Expression:
Int((<frxDBDataset1."PCKR91">+<frxDBDataset1."PCKR92">
+<frxDBDataset1."PCKR93">+<frxDBDataset1."PCKR94">
+<frxDBDataset1."PCKR95">+<frxDBDataset1."PCKR96">
+<frxDBDataset1."PCKR97">+<frxDBDataset1."PCKR98">)/8)
OK
Cancel

図13

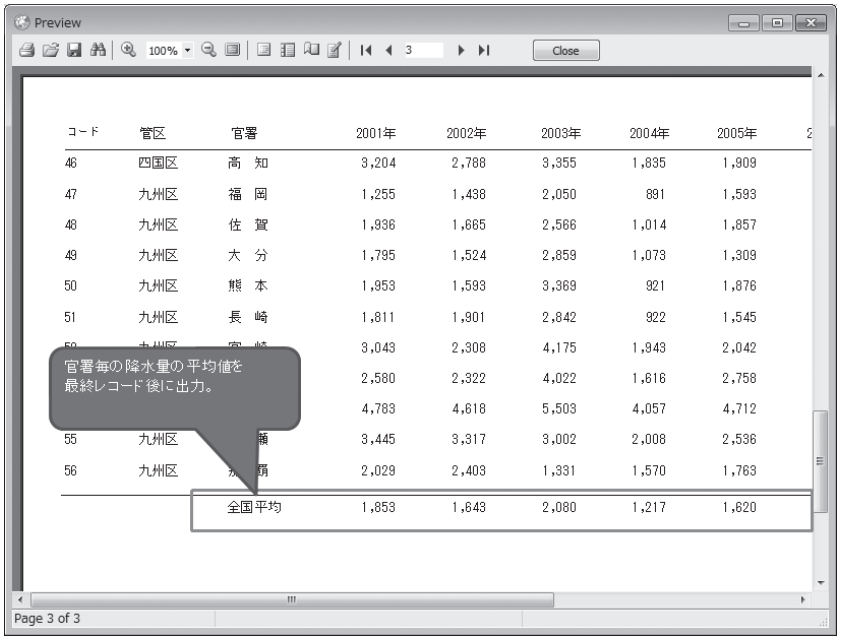

Preview
Close
コード 管区 官署 2001年 2002年 2003年 2004年 2005年
48 四国区 高 知 3,204 2,788 3,355 1,835 1,909
47 九州区 福 岡 1,255 1,438 2,050 891 1,593
48 九州区 佐 賀 1,936 1,665 2,566 1,014 1,857
49 九州区 大 分 1,795 1,524 2,859 1,073 1,309
50 九州区 熊 本 1,953 1,593 3,369 921 1,876
51 九州区 長 崎 1,811 1,901 2,842 922 1,545
3,043 2,308 4,175 1,943 2,042
2,580 2,322 4,022 1,616 2,758
4,783 4,618 5,503 4,057 4,712
55 九州区 3,445 3,317 3,002 2,008 2,536
56 九州区 2,029 2,403 1,331 1,570 1,763
全国平均 1,853 1,643 2,080 1,217 1,620
官署毎の降水量の平均値を
最終レコード後に出力。
Page 3 of 3

図14

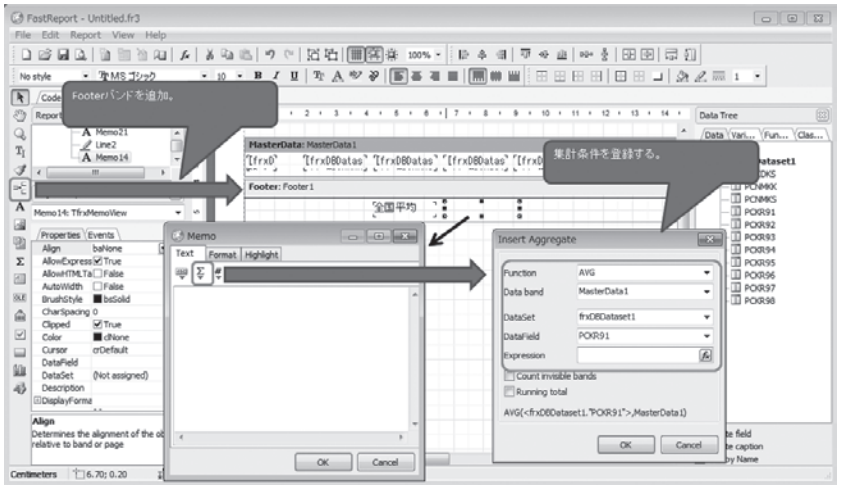

FastReport - Untitled.fr3
File Edit Report View Help
Footerバンドを追加。
集計条件を登録する。
Footer: Footer1
全国平均
Memo
Text Format Highlight
Insert Aggregate
Function AVG
Data band MasterData1
DataSet frxDBDataset1
DataField PCKR91
Expression
Count invisible bands
Running total
AVG(<frxDBDataset1."PCKR91">,MasterData1)
OK
Cancel

図15


Preview
Close
値が1000未満の部分のみ背景色をセットする。
Page 1 of 3

図16

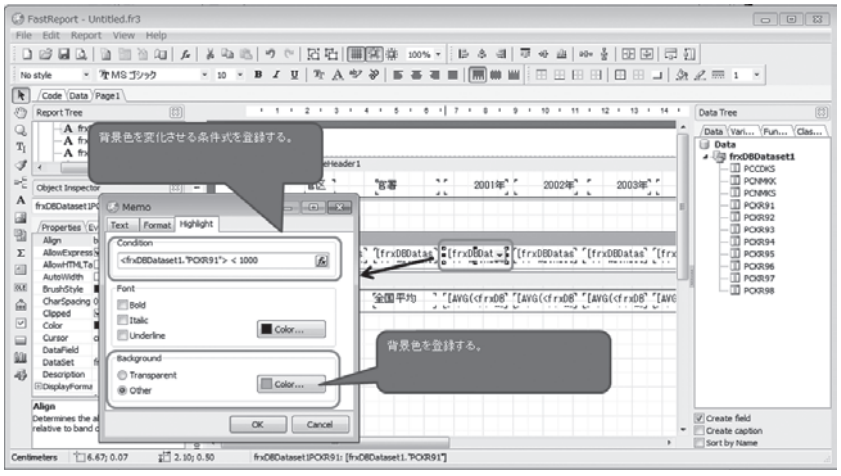
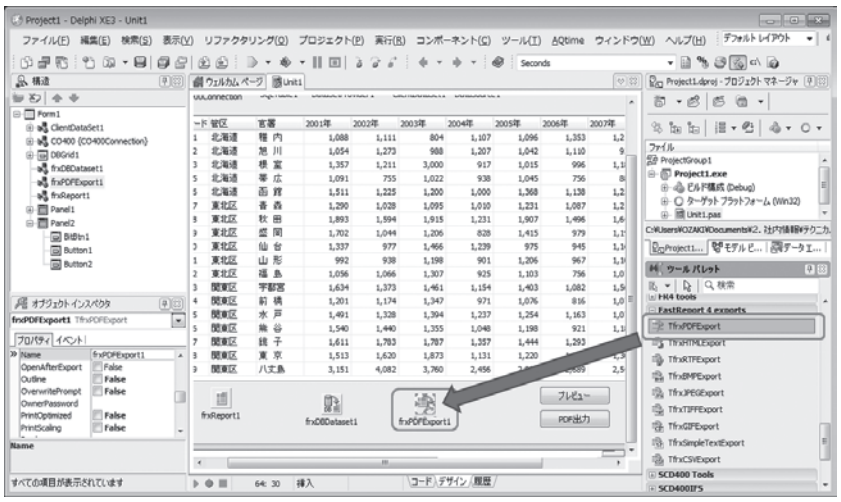

FastReport - Untitled.fr3
背景色を変化させる条件式を登録する。
背景色を登録する。

図17


Preview
Close
PreviewOptionsプロパティのサブプロパティであるButtonsプロパティを全てFalseにすると、ボタンが非表示となる。
Page 1 of 3

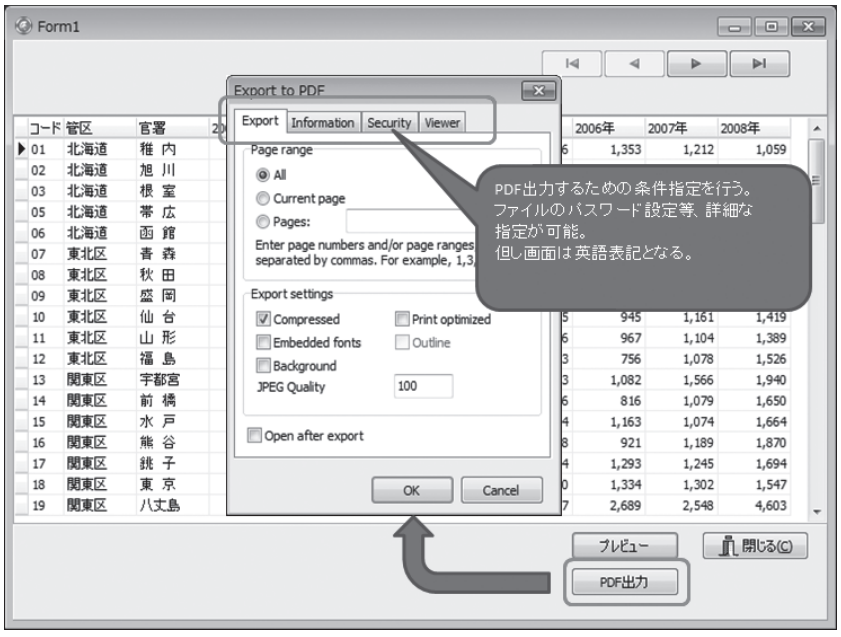
図18



ソース2

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  //PDF出力
  frxReport1.PrepareReport();
  frxReport1.Export(frxPDFExport1);
end;
```

図19



吉原 泰介

株式会社ミガロ.

RAD事業部 技術支援課 顧客サポート

Delphi/400で開発する64bitアプリケーション

Windows の 32bit/64bit 環境の移行段階にある今、
Delphi/400 のスマートな開発手法は最適解の 1 つ。

略歴
1978 年 03 月 26 日生
2001 年龍谷大学法学部卒
2005 年 07 月株式会社ミガロ. 入社
2005 年 07 月システム事業部配属
2007 年 04 月 RAD 事業部配属

現在の仕事内容
Delphi/400 と JC/400 の製品試験、および月 100 件に及ぶ問い合わせやサポート、セミナー講師などを担当している。

1.Windowsにおける 64bitアプリケーション

ここ数年で 64bit 端末の普及が急速に進み、Windows7 以降販売されている端末は、32bit よりも 64bit が多くなってきている。ビジネスにおいても、64bit 端末を導入している企業も増えてきており、社内端末の導入の際には 32bit/64bit の選択に悩む場面に遭遇しないとも限らない。

Windows の 32bit/64bit の選択において、検討や決断の大きなカギは端末自体のスペックだけでなく、その端末で利用したいアプリケーションが 64bit 環境に対応しているかが重要になってくる。

現在、Windows で使われているアプリケーションはまだ 32bit アプリケーションが圧倒的に多いが、かつて 16bit が主流であった Windows が現在では 32bit が主流になっているように、64bit も近い将来、Windows の標準環境となると予想される。

そうした背景もあり、本稿では今回、Delphi/400 での 64bit アプリケーション開発について紹介したいと考えた。

具体的には、64bit 環境の特徴から、64bit アプリケーション開発や開発環境のポイントを順に説明する。

1-1. 32bit/64bit環境の違い

Windows の 64bit 環境と 32bit 環境では何が違うのだろうか。

bit「ビット」とは、コンピュータが扱う情報の最小単位である。bit 数が大きければ大きいほど、処理できる情報量が多くなり、多くの処理を CPU で実行できるようになる。そのため、単純に 32bit よりも、64bit のほうがより高性能な処理を行うことができる。

また、ハードウェアの観点でも環境を考えてみる。CPU は常に高性能化し続けており、CPU の処理能力を十分に活用するためには、メモリも大容量が必要になってきている。

このメモリという観点も、32bit/64bit の環境を考えるうえでの重要なポイントになる。なぜなら Windows の OS では、bit 数やエディションによって使用できるメモリ容量に上限が決まっているからである。

標準的な 32bit の Windows OS と 64bit の Windows OS のメモリ上限を比較してみる。

【32bit Windows OS の最大メモリ】
Windows 8 Professional 32bit：
メモリ上限 4GB

【64bit Windows OS の最大メモリ】
Windows 8 Professional 64bit：
メモリ上限 512GB

このように同じ Windows 8 でも、bit 数の違いによって使用できるメモリの上限に大きな差がある。

・32bit Windows OS

32bit の Windows OS では通常、メモリは 4GB までしか使うことができない。OS 等で使用する部分を除くと、実

図1

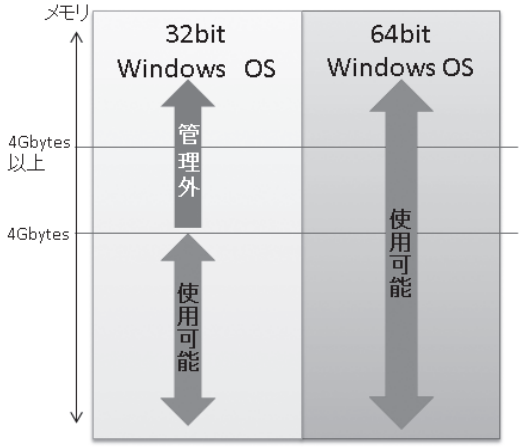
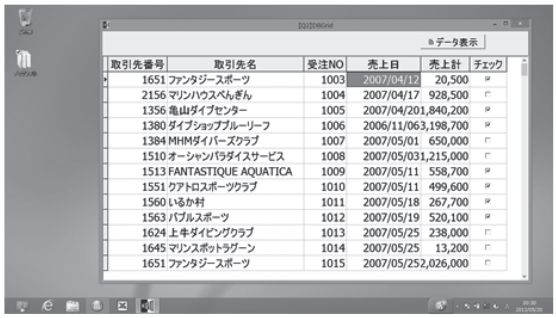


図2



取引先番号	取引先名	受注NO	売上日	売上計	チェック
1651	ファンタジースポーツ	1003	2007/04/12	20,500	
2156	マリンハウスぺんぎん	1004	2007/04/17	928,500	
1356	亀山ダイブセンター	1005	2007/04/20	1,840,200	
1380	ダイブショップブルーリーフ	1006	2006/11/06	3,198,700	
1384	MHMダイバーズクラブ	1007	2007/05/01	650,000	
1510	オーシャンパラダイスサービス	1008	2007/05/03	1,215,000	
1513	FANTASTIQUE AQUATICA	1009	2007/05/11	558,700	
1551	クアトロスポーツクラブ	1010	2007/05/11	499,600	
1560	いるか村	1011	2007/05/18	267,700	
1563	バブルスポーツ	1012	2007/05/19	520,100	
1624	上牛ダイビングクラブ	1013	2007/05/25	238,000	
1645	マリンスポットラグーン	1014	2007/05/25	13,200	
1651	ファンタジースポーツ	1015	2007/05/25	2,026,000	

図3

【64bit Windows OS の互換機能システム】

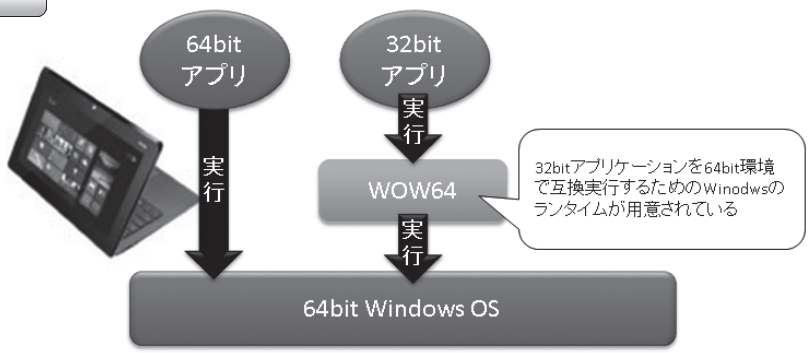
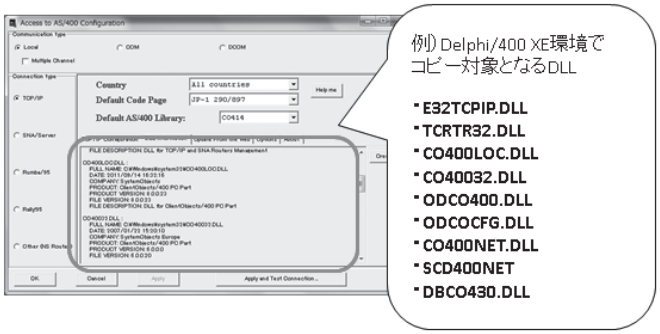


図4

【Delphi/400 XE Configuration】



質3.25GB程度になる。最近の端末は大容量のメモリ拡張ができるようになっているが、32bitのWindows OSを使用する場合、たとえ100GBのメモリを搭載しても実際には4GBしか利用することができない。【図1】

・64bit Windows OS

これに対して、64bitのWindows OSでは同じWindows 8でも512GB使うことができるため、スペックとしては、最大128倍のメモリを使用することができ、スペックの差が歴然としている。

ただ現状では、メモリが4GB搭載されていれば、それなりに高スペックな端末であるため、64bitを採用しなければ運用できないということはほとんどない。そのため、現在は32bitと64bitを選択できる移行段階にあると言える。

1-2. 32bit互換機能WOW64

ここまで32bitと64bitのWindows環境の違いを説明したが、アプリケーションについても考察する。

現時点では、まだ32bitアプリケーションが多いということを述べたが、それでは64bit Windowsを使う場合、これまで使用していた32bitアプリケーションがまったく動作しないのだろうか？

実は、64bit Windows上でも、32bitアプリケーションは互換で動作することができる。例えば、32bit端末で使用していたDelphi/400のアプリケーションは、64bit端末でも実行することができる。【図2】

・WOW64

その理由は、64bitWindowsに搭載されている「WOW64（Windows 32bit emulation on Windows 64bit）」と呼ばれる、32bitアプリケーションを互換実行するためのシステムのおかげである。

64bitアプリケーションは、もちろん64bitWindows上で実行することはできる。一方、32bitアプリケーションは通常、64bitWindows上では実行することはできない。それでは64bit端末を誰もが敬遠してしまう。

そのため、このWOW64では32bit用のランタイムが用意されており、これを利用することで、32bitアプリケーションでも64bitWindows上で実行できるようになっている。【図3】

WOW64は、32bit端末から64bit端末へ移行するにあたって、非常に重要なシステムだと言える。

・Delphi/400のモジュールコピー

しかし、ここで気をつけておかなければならないのは、WOW64はあくまでWindowsの32bit用ランタイム（実行環境モジュール）なので、Windows以外のソフトやアプリケーションのランタイムは含まれていないという点である。

Windows以外のアプリケーションのランタイムについては、WOW64で用意された\Windows\SysWOW64というフォルダーに必要なモジュールをコピーする必要がある。

例えば、XE3以前のDelphi/400を64bit端末で実行する場合には、Delphi/400のモジュールをSysWOW64にコピーする。

Delphi/400で必要となるモジュールは、Delphi/400 Configuration画面にあるDLL Informationというタブですべて確認できるようになっている。ここに表示されるSystem32フォルダーのdllを、32bit端末からSysWOW64にコピーするだけでよい。【図4】

最後に、もう1つ注意点を記しておく。32bitアプリケーションは、このWOW64によって32bit、64bitどちらの端末であっても使用することができる。

しかし、64bitアプリケーションは、32bit端末では互換実行することができないので、アプリケーションの使用には注意が必要になる。【図5】

1-3. 64bitアプリケーションの利点

32bitアプリケーションが互換動作するのであれば、64bitアプリケーションより便利と考える方も多い。実際に32bit/64bit端末が混在する運用環境では、それも正しい選択になる。

それでは、64bitアプリケーションのメリットについて、考えてみよう。64bitアプリケーションの最大のメリットは、前述したようにメモリを最大限に活用できるという点になる。

32bitアプリケーションがWOW64で実行できるとはいえ、メモリ使用の上限は32bitに準じているので、互換動作では2GB程度のメモリしか使用することができない。それに対して、64bitアプリケーションのメモリ使用の上限は8TBまで拡張されている。（もちろん端末に搭載されているメモリ容量には依存する。）

例えば、32bitアプリケーションでは、メモリが足りなくなった場合、ハードディスクの領域を使って回避したり、必要に応じてロードしたりするなどの効率的でない処理実行となってしまう。一方、64bitアプリケーションでは、十分なメモリさえ搭載していれば、すべてメモリ内で操作することができるので、CPUを活かした処理パフォーマンスを期待することができるということになる。

2.Delphi/400の64bitアプリケーション開発

前章までは64bitについての一般的な考察を行ってきたが、ここからはDelphi/400のアプリケーション開発について説明する。

Delphi/400ではバージョンXE3から、64bitアプリケーションも簡単に開発できる機能を実装している。（もちろん、これまでどおり32bitアプリケーションの開発も可能である。）

Delphi/400での64bit開発では、コンパイル機能を切り替えるだけで64bitアプリケーションを生成することができるようになっている。

つまり、開発者は64bitアプリケーションとしてのプログラム考慮や知識はほとんど必要がなく、Delphi/400側で開発を制御してくれる。そのため、作成したプログラムを32bit/64bitどちらのアプリケーションとしてコンパイルするかを選択するだけで開発することができる。

それでは、以降からDelphi/400の優れた64bit開発環境・手法について詳しく説明する。

図5

64bit ⇔ 32bit 互換注意点

	32bitアプリケーション	64bitアプリケーション
32bit端末	○	×
64bit端末	○	○

【64bitアプリを32bitWindowsで実行】

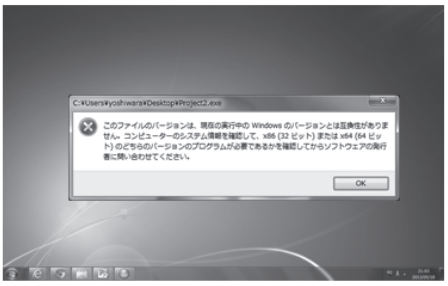
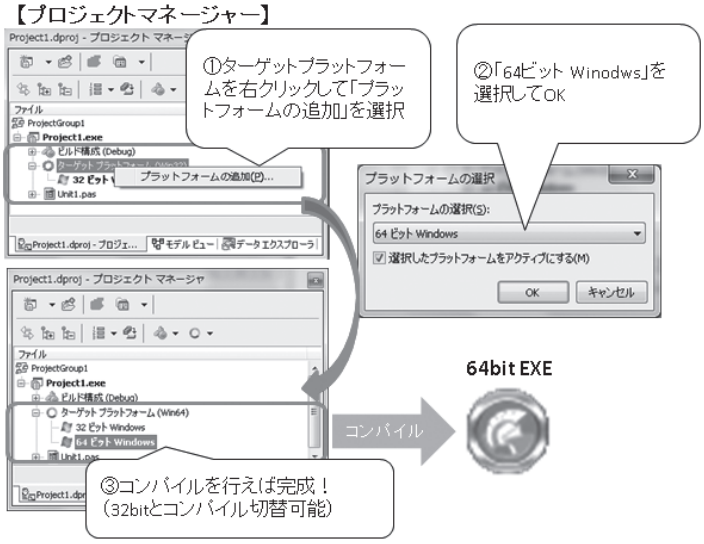


図6

【Delphi/400 開発環境】

図7

【プロジェクトマネージャー】



2-1. 64bitアプリケーション開発手法

Delphi/400での64bitアプリケーション開発では、特別なプログラミング知識や開発方法は必要ない。Delphi/400で開発経験がある方であれば、従来のアプリケーションとまったく同じ方法で開発することができる。【図6】

開発したプログラムをそのままコンパイルすれば、従来通りの32bitアプリケーションが完成するが、64bitアプリケーションを開発する場合には、ここで別のコンパイル操作を行う必要がある。

具体的には、開発環境のプロジェクトマネージャで右クリックを行うと「プラットフォームの追加」が選択できるので、「64ビットWindows」を追加する。そしてコンパイルを行えば、これだけで64bitアプリケーションが完成する。【図7】

もちろんプロジェクトマネージャで「32ビットWindows」を選択して切り替えれば、いつでも同じプログラムソースから32bitアプリケーションを生成することができる。同じプログラムソースで管理できるので、非常にスマートな開発と言える。【図8】

Delphi/400で開発した64bitアプリケーションは実行しても、画面の見た目で64bit/32bitを判別することは難しい。

確認したい場合には、タスクマネージャーでアプリケーションのプロセスを参照すれば、64bitアプリケーションであることを判別することができる。【図9】

同じプログラムソースからコンパイルしたアプリケーションでも、「32ビットWindows」でコンパイルしたEXEは「(32ビット)」と表示される。これは、WOW64で互換動作していることを示している。一方、「64ビットWindows」でコンパイルしたEXEは「(32ビット)」の部分が表示されないので、64bitネイティブで動作していることがわかる。

以上が、Delphi/400での64bitアプリケーションの開発手法である。

もちろん64bit開発上での注意点がないわけではない。

Delphi/400では64bitコンパイル機能によって、アプリケーションを生成しているので、例えば、既存のプログラムを64bit化するためには、64bitでコンパイルができるプログラムが必要である。そして、コンパイルのためには、プログラムソースおよび64bitに対応したコンポーネントが必要となる。

つまり、市販で購入したコンポーネントやフリーのコンポーネントを利用している場合には、それらのコンポーネントの対応状況を確認する必要がある。(Delphi/400 XE3ではコンポーネントも64bitに対応している。)

また、BDE関連のコンポーネント(TableやQuery等)は64bitには対応していない。そのため、dbExpressのコンポーネントに変更する必要があるので注意してほしい。

2-2. 64bitアプリケーション開発環境の考慮点

最後に、Delphi/400の開発環境について触れておきたい。Delphi/400 バージョンXE3では、開発環境として32bit端末、64bit端末のどちらでもインストールすることができる。

その開発環境の詳細を、32bit端末、64bit端末ごとにまとめた。【図10】

ここで、前述したが64bitアプリケーションは、32bit端末では実行することができないことを再度、思い出してほしい。つまり、32bit端末で64bitアプリケーションを開発する場合、コンパイルまでは可能であるが、アプリケーションの実行およびデバッグはできない。

そのため、32bit端末で64bitアプリケーションの開発を行う場合には、テスト用に64bit端末が必要となるので注意してほしい。なお、テスト用の64bit端末があれば、リモートでのデバッグも可能である。

3.まとめ

現時点では、まだ32bitアプリケーションでの開発・運用が一般的であるが、今後開発されるアプリケーションは、これまで以上に便利になり、より多くの画面項目で大容量のデータを処理するように進化していくことが予想される。そうしたアプリケーションを十分なパフォーマンスで動作させるためには、64bit環境に対応したアプリケーションが必須になってくる。

Delphi/400では、64bitアプリケーションが開発できるだけでなく、32bitアプリケーション、64bitアプリケーションを同じプログラムソースで管理することができる。そのため、同じアプリケーションを32bit用、64bit用に重複開発したり、煩雑なプログラム管理も必要がない。

Windowsの32bit/64bit環境の移行段階にある今、Delphi/400のスマートな開発手法は、開発者がこうした環境に対応するための最適解の1つと言えるだろう。

本稿では、64bit環境・アプリケーション、そしてDelphi/400での開発手法について説明してきたが、皆様の今後のシステム開発や環境検討の一助になれば幸いである。

M

図8

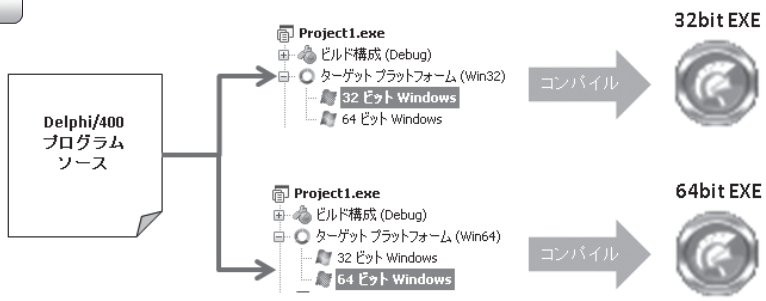


図9

【タスクマネージャー】

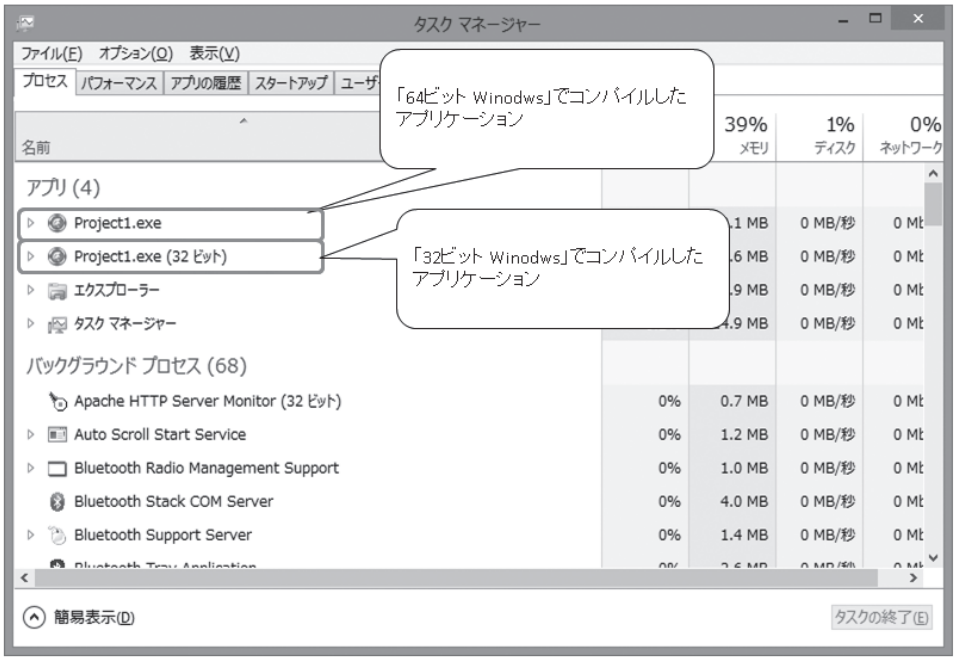
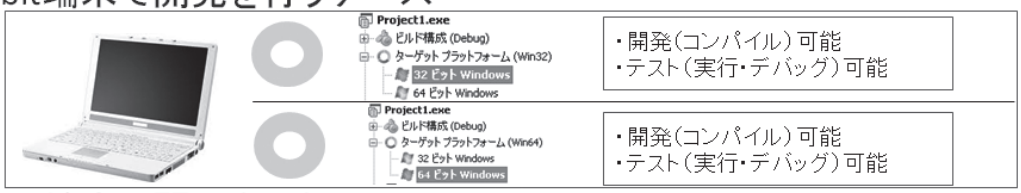
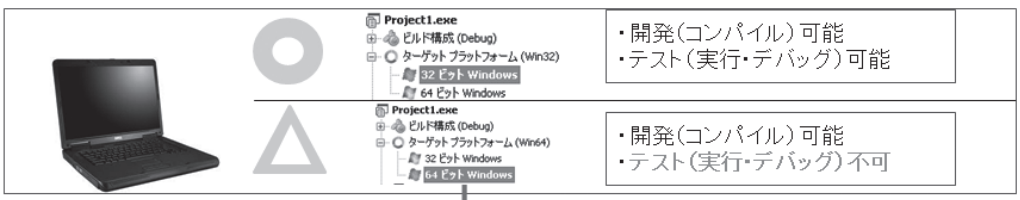


図10

64bit端末で開発を行うケース



32bit端末で開発を行うケース



	32bit 開発	32bit テスト	64bit 開発	64bit テスト
64bit端末	○	○	○	○
32bit端末	○	○	○	×

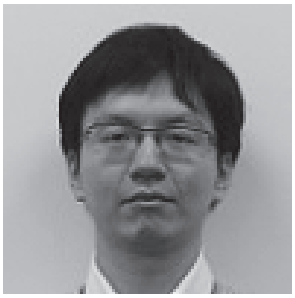
佐田 雄一

株式会社ミガロ.

システム事業部 システム1課

Webコンポーネントのカスタマイズ入門

**カスタムコンポーネントを開発できれば、さらなるWeb開発の効率化が見込める。
VCL for the Webならではのカスタムコンポーネント作成手法を述べる。**



略歴
1985年12月6日生
2009年甲南大学経営学部卒
2009年04月株式会社ミガロ. 入社
2009年04月システム事業部配属

現在の仕事内容
Delphi/400を利用したシステム開発や保守作業を担当。Delphiおよび Delphi/400のスペシャリストを目指して精進している。

1.はじめに

「VCL for the Web（IntraWeb）」はDelphi/400の開発機能の1つである。この機能を用いると、C/S（クライアント/サーバー型）アプリケーションの開発と同様の手法で、Webアプリケーションを作成することができる。

このVCL for the Webを活用することで、Web開発に馴染みの薄い開発者であっても、使い慣れたDelphi/400を用いて、容易にWebアプリケーションを作成することが可能になるだろう。

本稿では、Delphi/400を使用したWebアプリケーション開発のバリエーションをさらに広げていただけるよう、特にコンポーネントのカスタマイズ手法について取り上げ、くわしく紹介する。

2.カスタムコンポーネントを利用するメリット

Delphi/400では、開発機能の1つである“継承”を活用して、既存のコンポーネントから新しいコンポーネントを作成することができる。【図1】

このようにして作成されたコンポーネントは「カスタムコンポーネント」と呼ばれ、要件にあわせてさまざまな追加機能を組み込むことができる。

本稿では、VCL for the Webならではのカスタムコンポーネント作成手法について述べる。

なお、C/Sアプリケーションの開発におけるカスタムコンポーネントの作成手法については、『ミガロ. テクニカルレポート 2012』に、「カスタマイズコンポーネント入門～Delphi/400開発効率向上」を掲載している。わかりやすく解説されているので、ぜひ参考にしていただきたい。

3.Ajax・JavaScriptとの連携

VCL for the Web専用コンポーネントの特徴の1つとして、Ajaxと連携したAsyncイベントが利用できる点が挙げられる。

・Ajax

Ajaxとは「Asynchronous JavaScript + XML」の略で、JavaScriptによって実現する機能の総称である。Ajaxの仕組みを作成することは難しいが、Delphiではイベントで提供されているため、簡単に利用することが可能である。

例えば、C/Sアプリケーションの開発においては、TEditでフォーカスが抜けた際にOnExitイベントが実行される。一方、VCL for the WebアプリケーションのTIWEditにおいては、OnAsyncExitイベントが実行される。このイベントでは、HTML上でフォーカスが抜けたことをJavaScriptで検知し、非同期で処理を行うことが可能である。【図2】

・Asyncイベント

C/Sアプリケーションにおけるイベントの多くは、VCL for the WebアプリケーションではAsyncイベントで代替

図1

図1　コンポーネント継承の流れ

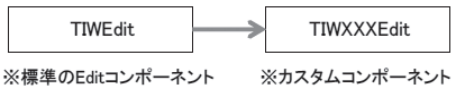


図2

図2　OnAsyncExit時に名称を取得する例

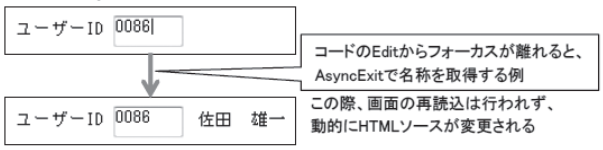


図3

図3　コンポーネントを新規作成する

図4

図4　フレームワークを選択する（Version XE3～）

されている。加えて、これらはAjaxの処理に依存しているため、中には実行タイミングや処理の種類が異なるものも存在する。

例えば、C/Sアプリケーションにおける TEdit の OnChange イベントは、1文字でも値が変更されたタイミングで実行される。それに対し、TIWEditのOnAsyncChangeイベントでは、フォーカスが抜けたタイミングで、値が変更されていれば実行される。

・JavaScript連携

VCL for the Webアプリケーションのコンポーネントは、JavaScriptと密接な関係がある。コンポーネントにJavaScriptを埋め込み、キー押下時やフォーカスセット時などの制御をJavaScriptで行うことが可能である。

コンポーネントのカスタマイズにおいて、実際に埋め込み、制御を行う手法については、次章でコンポーネントを作成しながら解説する。

※ Asynchronous =非同期

4.数値専用WebEditの作成例

カスタムコンポーネントを作成することで、処理の共通化を実現し、開発の効率化を図ることが可能になる。VCL for the Webのカスタムコンポーネントも同様であり、その特徴は前述の通りである。

ここからは、実際にVCL for the Webアプリケーション用のカスタムコンポーネントを作成していこう。

今回は例として、TIWEditを継承して「TIWNumEdit」という数値入力専用のWebEditを作成する。前述のAsyncイベントおよびJavaScriptを活用することで、C/Sアプリケーションのコンポーネントと同等の制御をWebアプリケーションでも実現できるだろう。

(1)コンポーネントの新規作成

C/Sアプリケーションで作成する際と同様に、コンポーネントソースを新規作成する。手順に沿って説明していこう。

なお、今回の開発環境はDelphi/400 Version XE3を使用している。

①コンポーネントの新規作成

メニューの［ファイル｜新規作成｜その他］を選択し、新規作成ウィンドウより「コンポーネント」を選択する。【図3】

②フレームワークの指定(Version XE3～)

フレームワーク指定のダイアログでは、VCL for Delphi Win32を選択する。【図4】

③継承元コンポーネントの指定

今回はTIWEditを継承するため、TIWEditを指定する。【図5】

④コンポーネント名とパレットページ、保存先の指定

クラス名に、今回作成するコンポーネント名であるTIWNumEditを指定する。

パレットページ名に、新規追加するコンポーネントパレットのパレットページ名を指定する。なお、今回はデフォルト値のSamplesを使用する。

ユニット名に、今回追加するカスタムコンポーネントのpasファイル名、および保存先を指定する。なお、今回は保存先をC:¥Projects¥TechRep¥SampleNumEdit.pasと設定する。【図6】

⑤コンポーネントの新規作成の完了

完了すると、図7の画面が表示される。【図7】

(2)宣言部の作成

今回の数値入力用WebEditでは、以下の機能を実装したいと思う。

・Value：Edit値を数値として設定/保持するための新規プロパティ。
・Text：既存のTextから変更し、読み取り専用プロパティにする。
・JavaScriptで不必要なキー押下を無効にする。
・フォーカスが抜けた時に、数値でない値がセットされていれば色を変更する。

これらを実現するため、ソース1のように宣言部を記述する。それぞれのプロパティやイベントの詳細については後述する。【ソース1】

① uses

HookEventsイベントの実装に必要なIWRenderContext、IWScriptEvents、色の指定を行うために必要なGraphicsを宣言部のusesに追加する。

② private宣言

入力された数値を内部で保管するための変数FValueをprivate部に宣言する。またSetValue、GetValue、GetTextについては後で自動生成されるため、この時点では記載不要である。

③ protected宣言

ScriptEventsの設定を行うためのHookEventsイベント、およびフォーカスが抜けた際に実行するEditExitイベントをprotected部に宣言する。また、HookEventsは上位クラスから継承されているため、宣言の後ろにoverride;を記述する。

④ public宣言

クラス生成時イベント（constructor）を使用するため、public部に宣言する。

⑤ published宣言

実際に画面に貼りつけた際にオブジェクトインスペクタに表示される変数Valueを宣言する。また、Textは上位クラスから存在するが、writeを指定しない（readのみとする）ことで、読み取り専用のプロパティになる。

各宣言部が完成したら、Ctrl + Shift + ［C］キーを同時に押下することで実装部、および先述したSetValue、GetValue、GetTextが自動生成される。

(3)Createイベントの作成

Createは、コンポーネントの生成時に実行されるイベントである。

画面にコンポーネントを配置した際や、ソース内で動的にコンポーネントを生成した際のプロパティの初期値は、ここで設定した値となる。

記述例については、ソース2を参考にしていただきたい。これらの中で注意が

図5

図5 継承元コンポーネントを指定する

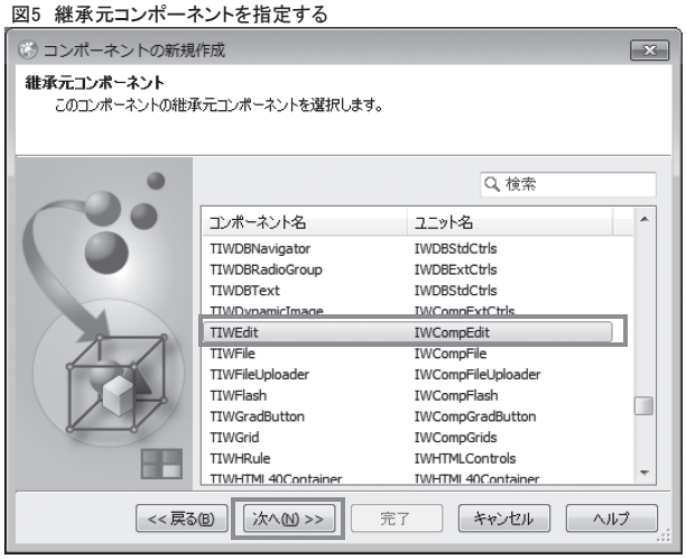


図6

図6 コンポーネント名とパレットページ、保存先を指定する

図7

図7 コンポーネントソース新規作成完了時のソース

必要なプロパティやイベントは、次の2点である。【ソース2】

・Font.FontName
Createイベント内でフォント名を指定する場合、「MSゴシック」のように日本語が入っていると正しく認識しないため、英語で指定する。

・EditExit
OnAsyncExitイベントにEditExitを代入することで、フォーカスが抜けた際にEditExitイベントが実行されるようになる。このため、EditExitイベントの引数はOnAsyncExitと同じものを設定しておく。

(4) HookEventsイベントのコーディング

コンポーネント内でJavaScriptを利用したイベントを実行させることができるScriptEventsプロパティを利用して、キー押下時に、数値入力と無関係のキー押下を無効にする処理を実装する。

これを実現するため、HookEventsイベントにおいて、ScriptEventsプロパティに対してJavaScriptを設定する。なお、本稿ではJavaScript言語の詳細説明については割愛するが、参考文献や解説しているWebサイトが多く存在するので、比較的修得がしやすい言語である。

ソースの記述については、ソース3を参考にしていただきたい。【ソース3】

(5) ValueとTextの読み取り/書き出し時イベントの設定

今回はEditの値を数値として保持させるため、既存のTextを読み取り専用プロパティに変更し、新たに数値型(Double)のValueプロパティを作成して値の入出力を行う。

記述例については、ソース4を参考にしていただきたい。【ソース4】

・GetText
GetTextはTextの値を読み取るためのイベントで、他のクラスからTextを呼び出すと、このイベントを通して値が渡される。Textではなくinherited Textと記述することで、画面にセットされている文字列をそのまま返すことができる。

・GetValue
GetValueはValueの値を読み取るためのイベントで、GetTextと同様に他のクラスからValueを呼び出すと、このイベントを通して値が渡される。

・SetValue
SetValueは逆にValueに値を書き込むためのイベントで、内部保管用のFValue変数および画面のTextにセットされた数値を代入する。

(6) EditExitイベントのコーディング

EditExit手続きを使用することで、Editからフォーカスが抜けた時に、Textに入力した文字列を数値に変換して内部保管値に代入する処理を行う。

その際、コピー&ペースト等によって数値でない文字列がセットされた状態でフォーカスが抜けた場合には、背景色を赤に変更し、内部保管値に初期値(0)を代入する。

また、数値の判定に成功した際は、背景色をもとに戻す処理を記述している。ただし、無色を示すclNoneをBGColorプロパティにセットしても色の変更が行われないため、他の色(今回の場合ウィンドウ色)を指定する必要がある。

ソース記述については、ソース5を参考にしていただきたい。【ソース5】

以上で、TIWNumEditコンポーネントのコーディングは完了である。

(7) インストールと画面への実装

作成したコンポーネントを実際に使用するには、パッケージを新規作成してインストールを行う。手順に沿って説明しよう。

①パッケージを新規作成する
メニューの［ファイル｜新規作成｜その他］を選択し、新規作成ウィンドウより「パッケージ」を選択する。【図8】

パッケージを作成したら、プロジェクトを任意の場所に名前を付けて保存する。(今回作成したユニットと同じディレクトリ内が望ましい。)今回はSamplePackage.bplとして保存する。

なお、既存のパッケージに今回作成したユニットを追加する場合は、ここの手順は不要である。

②パッケージにユニットを追加
次に、メニューの［プロジェクト｜プロジェクトに追加...］を選択し、今回作成したユニットをパッケージに追加する。【図9】

③インストール
図9のプロジェクトマネージャ内にあるSamplePackage.bplを右クリックし、再構築(ビルド)を行う。

再構築が正常に行えたことを確認したら、再度SamplePackage.bplを右クリックし、「インストール」を選択すると、コンポーネントのインストールが実行される。

インストールが正常に完了すると、ダイアログが表示される。【図10】

④画面への組み込み
インストールが完了したら、VCL for the Webの新規プロジェクトを作成すると、デザイン画面のツールパレットにTIWNumEditが追加される。【図11】

TIWNumEditを選択して画面内をクリックすると、図12のような項目が配置される。見た目はカスタム前のTIWEditと異なっているが、TIWEditと同じようにプロパティやイベントの設定を行うことが可能である。【図12】

⑤ライブラリパス設定と実行
作成したカスタムコンポーネントを使用したプロジェクトのコンパイルが通るようにするには、Delphi/400開発環境のライブラリパスの設定を行う必要がある。

メニューの［ツール｜オプション］を選択し、今回作成したSampleNumEdit.pasの保管先ディレクトリをライブラリパスの一覧に追加する。【図13】

ここまでの手順で、今回作成したTIWNumEditを使用する準備はすべて完了である。

アプリケーションにコンポーネントを組み込んでコンパイルすれば、TIWNumEditの機能動作を確認するこ

ソース1

ソース1 宣言部の記述

```
unit SampleNumEdit;

interface

uses
  System.SysUtils, System.Classes, Vcl.Controls, IWVCLBaseControl,
  IWBaseControl, IWBaseHTMLControl, IWControl, IWCompEdit,
  IWRenderContext, IWScriptEvents, Graphics;

type
  TIWNumEdit = class(TIWEdit)
  private
    { Private 宣言 }
    FValue: Double;                                          // 実際に値を保管している変数
    procedure SetValue(const Value: Double);                 // Valueに書き込む時の処理
    function GetValue: Double;                               // Valueが読み取られる時の処理
    function GetText: String;                                // Textが読み取られる時の処理
  protected
    { Protected 宣言 }
    procedure HookEvents(AContext: TIWPageContext40;
                         AScriptEvents: TIWScriptEvents); override; // JavaScriptの設定
    procedure EditExit(Sender: TObject; EventParams: TStringList); // フォーカスが抜けた時の処理
  public
    { Public 宣言 }
    constructor Create(AOwner: TComponent); override;        // コンポーネント生成時処理
  published
    { Published 宣言 }
    property Value: Double read GetValue write SetValue;     // 数値として値保持
    property Text: String read GetText;                      // 既存のTextを読み取り専用にする
  end;
```

HookEventsで使用するため、usesに追加

色(TColor)を使用するため、usesに追加

writeを指定しないことで、Textが読み取り専用になる

ソース2

ソース2 Createイベントの記述

```
constructor TIWNumEdit.Create(AOwner: TComponent);
begin
  inherited;
  Alignment      := taRightJustify; // 右揃え(Version2009以降で対応)
  Font.FontName  := 'MS GOTHIC';    // フォント名(英語で指定)
  Font.Size      := 10;             // フォントサイズ
  OnAsyncExit    := EditExit;       // OnAsyncExitのタイミングでEditExitを実行
  Self.BGColor   := clWindow;       // 背景色の初期値(ウインドウ色)
  FValue         := 0;              // 実数値の初期値
  inherited Text := '0';            // 画面に表示する文字列の初期値
end;
```

ソース3

ソース3 HookEventsイベントの記述

```
procedure TIWNumEdit.HookEvents(AContext: TIWPageContext40;
  AScriptEvents: TIWScriptEvents);
const
  // OnKeyDownスクリプトイベント
  // 入力可能文字列を指定する関数
  sScript='m = String.fromCharCode(event.keyCode); '
    +'if("0123456789abcdefghi¥b¥r¥t".indexOf(m, 0) >= 0){ '
    +'  return true; } '
    +'else if (event.keyCode == 46){ '  // Deleteキー
    +'  return true; }'
    +'else if (event.keyCode == 39){ '  // 右方向キー
    +'  return true; }'
    +'else if (event.keyCode == 37){ '  // 左方向キー
    +'  return true; } '
    +'else if (event.keyCode == 96){ '  // テンキーの0
    +'  return true; } '
    +'else if (event.keyCode == 190){ ' // 小数点
    +'  return true; }'
    +'else if (event.keyCode == 110){ ' // テンキーの小数点
    +'  return true; } '
    +'else if (event.keyCode == 189){ ' // マイナス
    +'  return true; } '
    +'else if (event.keyCode == 109){ ' // テンキーのマイナス
    +'  return true; } '
    +'else if (event.ctrlKey == true && event.keyCode == 67){ ' // Ctrl+C
    +'  return true; } '
    +'else if (event.ctrlKey == true && event.keyCode == 86){ ' // Ctrl+V
    +'  return true; } '
    +'else if (event.ctrlKey == true && event.keyCode == 88){ ' // Ctrl+X
    +'  return true; } '
    +'else if (event.ctrlKey == true && event.keyCode == 90){ ' // Ctrl+Z
    +'  return true; } '
    +'else{ return  false; } ';
begin
  inherited;
  // 編集不可の場合処理しない
  if (not Editable) or (not Enabled) or (ReadOnly) then
  begin
    Exit;
  end;

  // OnKeyDown時に、スクリプトイベントを実施する
  AScriptEvents.HookEvent('OnKeyDown', sScript);
end;
```

押下されたキーのキーコードやCtrlキーの押下有無をもとに判定を行う。 図6～図7と同様、trueが返らなかった場合は以降の処理が行われない

ここでScriptEventsをセット

とができるだろう。【図 14】

5.まとめ

近年、Web アプリケーションやスマートデバイス用アプリケーションへの需要が高まり、C/S アプリケーションだけではなく、Web アプリケーションでのシステム構築が多くなってきている。

今回は Web のカスタムコンポーネント開発手法を説明したが、カスタムコンポーネントを開発できるようになれば、さらなる Web 開発の効率化が見込めるだろう。

Web アプリケーション開発に難しさを感じておられる開発者の皆様にとって、本稿が新たなアプリケーション開発の一歩となる内容であれば幸いである。

M

ソース4

ソース4　GetText/GetValue/SetValueの記述

```
function TIWNumEdit.GetText: String;
begin
  Result := inherited Text; // 画面のTextの文字値を渡す
end;

function TIWNumEdit.GetValue: Double;
begin
  Result := FValue; // 実際に値を保管している変数FValueの値を返す
end;

procedure TIWNumEdit.SetValue(const Value: Double);
begin
  inherited Text := FloatToStr(Value); // 画面のTextに代入された値を表示
  FValue := Value; // 実際に値を保管している変数FValueに代入
end;
```

他のユニット等から「Text」プロパティが参照されると、この関数を通じて値を返す

他のユニット等から「Value」プロパティが参照されると、この関数を通じて値を返す

他のユニット等から「Value」プロパティに値が代入されると、この関数を通じてセットする

ソース5

ソース5　EditExit手続きの記述

```
procedure TIWNumEdit.EditExit(Sender: TObject; EventParams: TStringList);
var
  dCheck: Double; // 値変換用変数
begin

  if (not TryStrToFloat(inherited Text, dCheck)) then
  begin
    Self.BGColor := clRed;    // 入力値が数値でない場合、背景色を赤に変更
    FValue := 0;              // 実際に値を保管している変数FValueに0を代入
  end
  else
  begin
    Self.BGColor := clWindow; // 入力値が数値の場合、背景色をウインドウ色に変更
    FValue := dCheck;         // 実際に値を保管している変数FValueに入力値を代入
  end;
end;
```

TryStrToFloat関数は、実数変換成功時はTrueが返り、dCheckに変換値が代入される
変換失敗時はFalseが返る

図8

図8　パッケージの新規作成

図9

図9　パッケージにユニットを追加

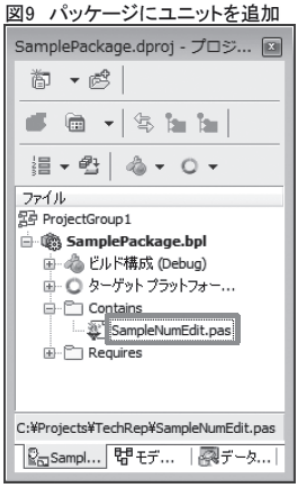


図10

図10　インストール完了

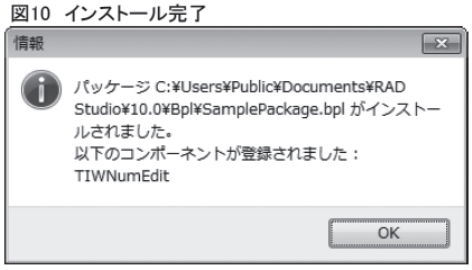


図11

図11　ツールパレット

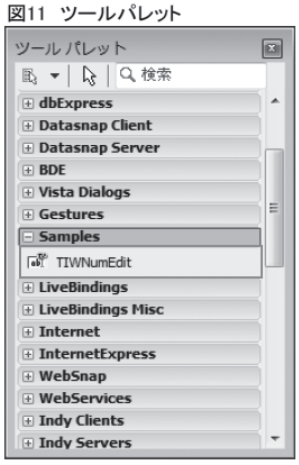
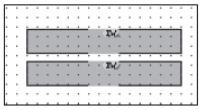


図12

図12　コンポーネント配置例



※カスタムコンポーネントを配置すると、
デザイン画面上の見た目が異なる

図13

図13 ライブラリパス設定

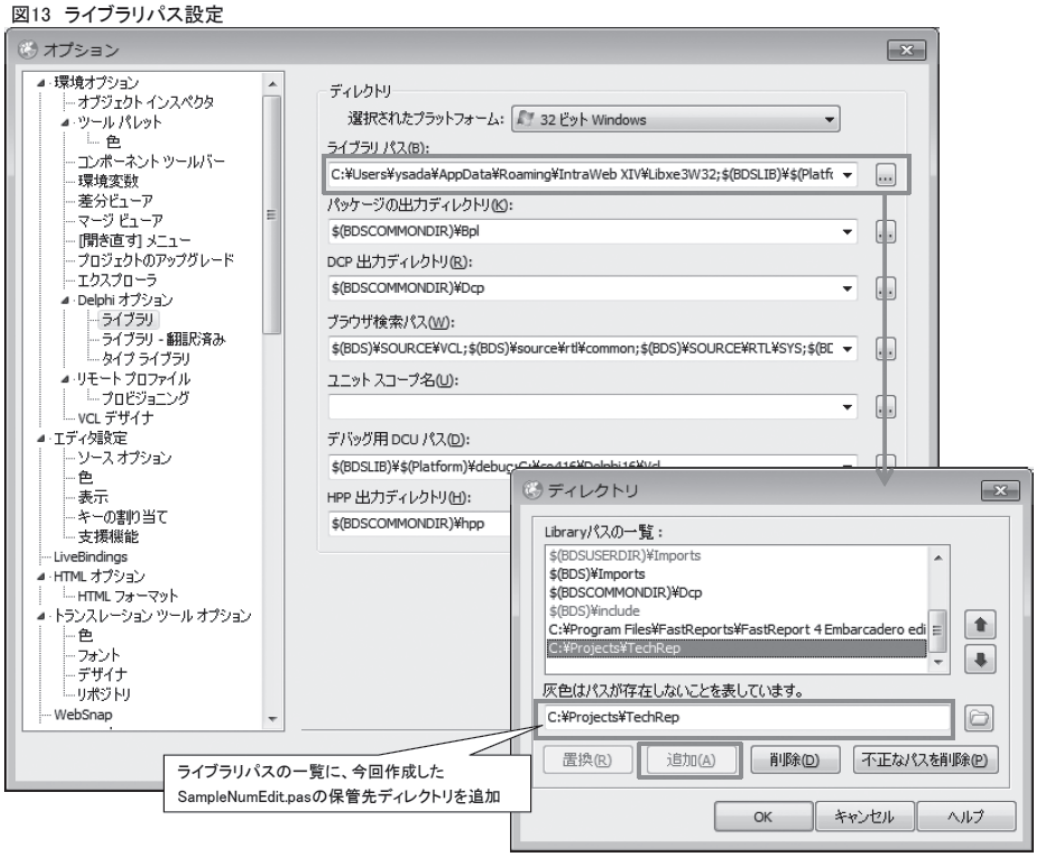
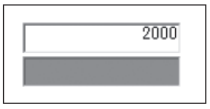


図14

図14 コンポーネント実行例



※ブランクは数値でないため、色が変化する

辻野 健 / 前坂 誠二

株式会社ミガロ.

システム事業部 システム2課

Indyを利用したメール送信機能開発

**ワークフローや注文システムで、確認メールを自動送信したい。
処理ロジックの後に、本稿のメール送信プログラムを追加設定するだけで実現可能だ。**

略歴　辻野 健
1988 年 06 月 10 日生
2011 年近畿大学 理工学部卒
2011 年 04 月株式会社ミガロ. 入社
2011 年 04 月システム事業部配属

現在の仕事内容
Delphi/400 を利用したシステム開発や保守作業を担当。日々開発スキルの向上を目指し、難易度の高いプログラムにチャレンジしている。

略歴　前坂 誠二
1989 年 03 月 21 日生
2011 年関西大学文学部卒
2011 年 04 月株式会社ミガロ. 入社
2011 年 04 月システム事業部配属

現在の仕事内容
Delphi/400 を利用したシステム開発や保守作業を担当。Delphi、Delphi/400 の開発経験を積みながら、日々スキルを磨いている。

1.はじめに

今日、ワークフローなどのシステムにおいて、一連の処理が完了したことをメールで自動送信するアプリケーションが多くなっている。

しかし、メール送信機能を開発するとなると、技術的に難しいイメージを持たれている方も多いのではないだろうか。

本稿では、Indy を使用し、容易にメール送信機能を実装する方法を紹介する。

2.Indyとは

「Indy」とは、オープンソースのネットワーク関連のコンポーネントのことであり、Delphi に標準で付属している。

TIdSMTP コンポーネントを用いたメールの送信や、TIdPOP3 コンポーネントを用いたメールの受信、TIdFTP コンポーネントを用いた FTP サーバーとの通信などさまざまな機能を実装することができる。

今回は、それらの機能の中で、TIdSMTP コンポーネントを用いてメール送信の実装方法を紹介する。まず初めに基本編として、シンプルな形式のメール送信方法から入り、次に応用編として、ファイルの添付などのメール送信方法を説明する。

なお、本稿で作成しているプログラムは、Delphi/400 XE を用い、Indy バージョンは Indy10_5040 を使用している。

3.メール送信プログラムの作成（基本編）

本章では、画面で入力した件名と本文を、指定したメールアドレスに送信するだけのシンプルなプログラムの作成方法について説明する。

今回は、SMTP サーバーには代表的なメールサービスである Gmail を使用して、プログラムを作成していく。

メール送信用画面の作成

VCL フォームアプリケーションより画面を新規作成し、図 1 のような画面を使用してメール送信プログラムを作成する。【図 1】

使用しているコンポーネントとプロパティの設定については、図 2 を参照してほしい。【図 2】

図 1 の画面の動作としては、送信ボタンを押下すると、To に指定したメールアドレスに、画面で入力した件名と本文のメールを送信する単純なものである。

送信ボタン押下時の処理のおおまかな流れは次の通りである。

① SMTP サーバーへ接続 →
②送信内容の設定と送信 →
③ SMTP サーバーの接続解除

接続設定

メール送信用画面作成後、SMTP サーバーへの接続を行うために TIdSMTP コンポーネントを Form1 に配置する。

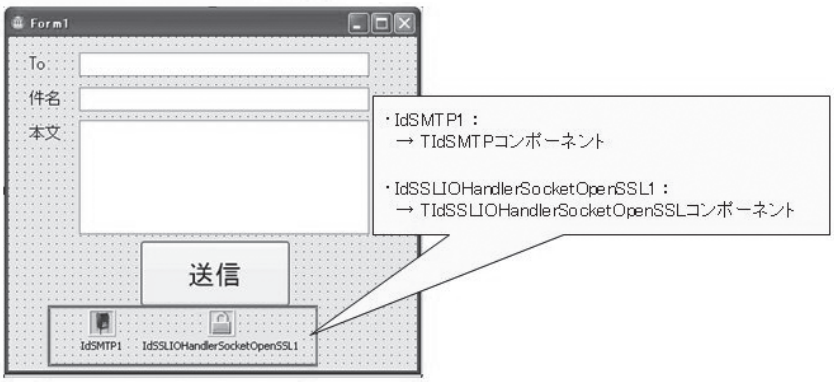
また、今回使用する Gmail のように、SSL を使用しているメールサービスの

【図1】

【図2】

使用コンポーネント	Nameプロパティ	Captionプロパティ
TLabel	Label1	To
TLabel	Label2	件名
TLabel	Label3	本文
TButton	Button1	送信
TEdit	Edit1	−
TEdit	Edit2	−
Tmemo	Memo1	−

【図3】



【図4】

・TIdSMTPコンポーネントのプロパティ設定

プロパティ名	設定値
Host	使用メールサービスのサーバー名 Gmailの場合は、'smtp.gmail.com'を設定
Port	ポート番号 今回は、SSLを使用するため465を設定
Username	使用メールサービスの メール送信を行いたいユーザーアカウント名
Password	上記のユーザーアカウント名で設定したパスワード
IOHandler	TIdSSLIOHandlerSocketOpenSSLの Nameプロパティ値
UseTLS	SSL未使用時は、utNoTLSSupport(デフォルト値) 今回は、utUseExplicitTLSを設定

SSL使用時のみ追加で設定

場合は、SSLの設定を行わなければメール送信が実行できない。そのため、TIdSSLIOHandlerSocketOpenSSLコンポーネントもForm1に配置する必要がある。【図3】

SSL（Secure Socket Layer）とは、インターネット上で送受信を行うデータを暗号化する技術のことである。データの送受信を暗号化することにより、第3者によるデータの盗聴や改ざんなどを防ぐことができるため、安全にデータ通信が行える。

・TIdSMTPコンポーネント

SMTPサーバーへの接続は図4のプロパティを設定し、Connectメソッドを実行するだけで接続できる。SMTPサーバーから接続解除を行う場合には、Disconnectメソッドを実行するだけである。【図4】

また、今回はメール送信にSSLを使用するため、IOHandlerプロパティに図3で配置したIdSSLIOHandlerSocketOpenSSL1を指定し、UseTLSプロパティにutUseExplicitTLSを設定する。

・TIdSSLIOHandlerSocketOpenSSLコンポーネント

SSLの設定についても、SMTPサーバーへの接続設定と同様に、送信ボタン押下時（TButtonのOnClickイベント）に行う。

SMTPサーバーへの接続の前には、図5のプロパティを設定する。【図5】

今回は、送信ボタン押下時（TButtonのOnClickイベント）の最初にSMTPサーバーの設定と接続処理を記述し、最後に接続解除処理を記述する。【ソース1】

メール送信内容の設定

SMTPサーバーへの接続が完了すると、次はメールの送信内容について設定する必要がある。送信内容を設定するために、TIdMessageコンポーネントをForm1に配置する。【図6】

・TIdMessageコンポーネント

送信情報の設定は、送信ボタン押下時（TButtonのOnClickイベント）のSMTPサーバーへの接続処理後に行う。

まず、Clearメソッドで初期化を行った後、図7のプロパティを設定する。今回、送信先のメールアドレス、件名、本文は画面の入力値をそれぞれセットする。【図7】

また、もし複数の宛先に送信したい場合は、複数の送信先のメールアドレスを「,（カンマ）」で区切ることにより実現可能である。

後は、送信情報を設定したIdMessage1を引数として、IdSMTP1のSendメソッドを呼び出すだけでメール送信を行うことができる。【ソース2】

ソース2でコンパイルと実行を行い、送信情報の設定を行った画面が図8である。そして、図8の画面から送信ボタンを押下し、メール送信を行った結果が図9である。【図8】【図9】

・文字コードとエンコーディング

図9を見ると、件名は画面で入力した内容が表示されているが、本文が「?????」という形で文字化けしているのがわかるだろうか。

このような本文の文字化けを防ぐためには、文字コードとエンコーディングの設定を追加で行う必要がある。図7のプロパティ設定にCharSetプロパティ、ContentTransferEncodingプロパティの設定を新たに追加する。【図10】【ソース3】

ソース3でコンパイルと実行を行い、図8の送信内容で、再びメール送信を行った結果が図11である。【図11】

図11では、IdMessage1のプロパティに文字コードとエンコーディングの設定を新たに追加したため、本文が文字化けせずに表示されている。このように、メール送信内容の設定では、TIdMessageコンポーネントで文字コードやエンコーディングの設定が重要である。

4.メール送信プログラムの作成（応用編）

本章では、前章の内容を応用したプログラムの作成方法を紹介する。その際、前章で作成したプロジェクトとソースを流用し、説明を行う。

ファイルの添付

メール送信において、送信するメールにファイルを添付する機会が多くある。ファイルの添付は、TIdAttachmentのCreateメソッドを使用するだけで、容易に実現が可能である。

今回は例として、Cドライブに AttachmentFileというフォルダーを作成し、その中にSAMPLE.jpgというファイルを配置する。そして、SAMPLE.jpgが実際に送信されたメールに添付されているかを確認する。

・TIdAttachmentのCreateメソッド

まず、TIdAttachmentを使用するためには、前章のソースのUses節にIdAttachmentFileを追加する。そして、メール送信処理の前にTIdAttachmentのCreateメソッドを呼び出す。

Createメソッドは、第1引数にTIdMessageコンポーネントのMessagePartsプロパティを指定する。第2引数には添付ファイルのパスを設定する。添付ファイルのパスは、絶対パス、相対パスどちらでも使用できる。また、添付ファイル名には日本語名を使用することも可能である。【ソース4】

ソース4のロジックで、実際に送信したメールにファイルが添付されるかを確認してみよう。

まずソース4でコンパイルと実行を行い、送信内容を図8で設定する。送信ボタンを押下した結果が図12である。【図12】

このように、送信メールにファイルを添付するためには、メール送信ロジックに、TIdAttachmentのCreateメソッドを呼び出すロジックを1行追加するだけで、容易に実装できる。

htmlの利用

さらに、Indyを使ったメール送信では、htmlを利用したメール送信も可能である。htmlを利用すると、文字の装飾や表の作成など、送信内容の幅が広がる。

htmlを利用するためには、メッセージのコンテンツタイプをhtml型に設定する必要がある。

【図5】

・TIdSSLIOHandlerSocketOpenSSLコンポーネントのプロパティ設定

プロパティ名	設定値
Host	使用メールサービスのサーバー名 【図4】のHostプロパティ値と同じ値を設定
Port	ポート番号 【図4】のPortプロパティ値と同じ値を設定
Destination	上記の 'Hostプロパティ値:Portプロパティ値' 今回は'smtp.gmail.com:465'を設定

【ソース1】

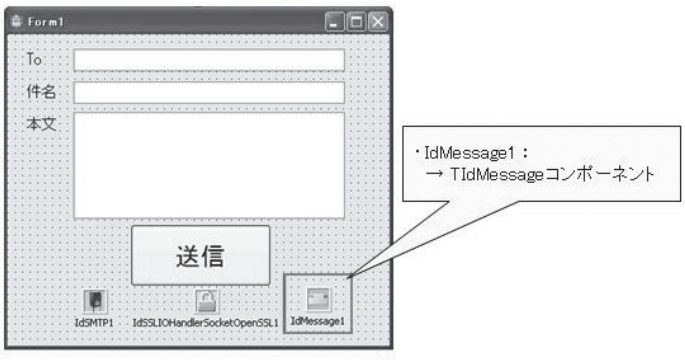
```
{*************************************************************************
 目的 : 送信ボタン押下時処理
 引数 :
 戻値 :
*************************************************************************}
procedure TForm1.Button1Click(Sender: TObject);
begin

  // SMTPサーバー接続設定
  IdSMTP1.Host     := 'smtp.gmail.com';              // サーバー名
  IdSMTP1.Port     := 465;                           // ポート番号
  IdSMTP1.Username := 'MigaroIndySample';            // ログインユーザーID
  IdSMTP1.Password := 'xxxxxxxxxx';                  // ログインパスワード


  IdSMTP1.IOHandler := IdSSLIOHandlerSocketOpenSSL1; // SMTPへSSLを設定
  IdSMTP1.UseTLS    := utUseExplicitTLS;             // 接続方式

  // SSL設定
  IdSSLIOHandlerSocketOpenSSL1.Host := IdSMTP1.Host; // サーバー名
  IdSSLIOHandlerSocketOpenSSL1.Port := IdSMTP1.Port; // ポート番号
  IdSSLIOHandlerSocketOpenSSL1.Destination :=
    IdSMTP1.Host + ':' + IntToStr(IdSMTP1.Port);     // 接続設定

  // SMTPサーバー接続
  IdSMTP1.Connect;

  // SMTPサーバー接続解除
  IdSMTP1.Disconnect;

end;
```

IdSMTP1のプロパティ設定

IdSSLIOHandlerSocketOpenSSL1のプロパティ設定

接続メソッドと接続解除メソッドの間に送信処理を記述していく【ソース2】～【ソース5】

【図6】



・IdMessage1：
→ TIdMessageコンポーネント

【図7】

・TIdMessageコンポーネントのプロパティ設定

プロパティ名	設定値
Recipient.EmailAdresses	送信先のメールアドレス
Subject	メールの件名
Body	メールの本文

そのために本稿では、まず先述のファイル添付で作成したロジックのUses節にIdTextを追加する。そして、ソース4の送信ボタン押下時処理のメソッド内において、TIdText型の変数TEXT1、TEXT2を定義しておく。

・ContentTypeプロパティ'multipart/mixed'

次に、IdMessage1のContentTypeプロパティを'multipart/mixed'に設定する。ContentTypeプロパティでは、どのようなコンテンツの種類で記述を行うかが指定できる。

'multipart/mixed'を指定すると、複数のコンテンツで記述を行うことが可能になる。今回の場合、複数のコンテンツとは、通常のメール文とhtml形式のメール文で記述を行うという意味になる。

メール文を設定するには、変数TEXTをTIdTextとして生成し、TEXTのContentTypeプロパティを通常のメール文の場合は'text/plain'、html形式のメール文の場合は'text/html'に設定する。

後は、本文の設定と文字化け防止のための、文字コードとエンコーディングの設定を忘れずに行えば、htmlを利用したメール送信が完成である。【ソース5】

最後に、ソース5でコンパイルと実行を行い、メール送信を行った結果が図13である。【図13】

このように、htmlを利用することで、文章だけでは伝えづらい内容も、視覚的にわかりやすく表現することができる。

5.補足

・SMTPサーバーへの接続と接続解除のタイミング

本稿では、送信処理の直前にSMTPサーバーへの接続を行い、送信処理の直後にSMTPサーバーとの接続解除を行っている。

一見、画面表示時に接続を行い、画面終了時に接続解除を行ったほうが、レスポンスがよくなるのではないかと思われる方もいるかもしれない。

しかし、本稿ではあえて、送信処理の直前と直後のタイミングで接続と接続解除を行っている。

その理由は、画面表示時にSMTPサーバーへの接続処理を行って、画面を終了せずに接続状態のまま一定時間が経過すると、SMTPサーバーとの接続タイムアウトが発生し、自動的に接続が解除されてしまうからである。つまり、送信処理自体が行えなくなるという可能性がある。

・他のメールサービスを使用する際の注意点

本稿では、SMTPサーバーとしてGmailを使用したため、記載のソースでメール送信機能を実現することができた。

ただし、Gmail以外のメールサービスでメール送信プログラムを作成する場合には、次の点に注意が必要となる。

例えば、本稿で記載したソースを参考に、自身が使用したいメールサービスのホスト名、ポート番号を設定し、メール送信処理を実装するとしよう。その際、SMTPサーバーへの接続は正常に行えるが、TIdSMTPのSendメソッドを呼び出しても、指定したメールアドレスに、メールが届かないケースやエラーが発生するケースがある。

そういった場合には、送信元メールアドレスを指定する必要がある。送信元メールアドレスの指定方法については、ソース6を参考にしてほしい。【ソース6】

6.最後に

今回は、Indyを利用したメール送信の実装方法を紹介した。

本稿で説明や記載している送信プログラムを参考にしたり、変更いただければ、実践的なメール自動送信の仕組みも容易に実装することができる。

例えば、注文システムで、注文確認メールを自動送信したい場合には、注文ボタンの処理ロジックの後に、本稿のメール送信プログラムを追加して送信内容を設定するだけで実現できる。

また、今回はGmailを例にメール送信機能の実装を行ったが、もちろん他のメールソフトでも利用することができる。

メールを連携した機能実装を検討される場合には、本稿の技術情報を役立てていただければ幸いである。

M

【ソース2】

```
{*************************************************************************
 目的 : 送信ボタン押下時処理
 引数 :
 戻値 :
*************************************************************************}
procedure TForm1.Button1Click(Sender: TObject);
begin

  // SMTPサーバー接続設定
  IdSMTP1.Host     := 'smtp.gmail.com';                  // サーバー名
  IdSMTP1.Port     := 465;                               // ポート番号
  IdSMTP1.Username := 'MigaroIndySample';                // ログインユーザーID
  IdSMTP1.Password := 'xxxxxxxxxx';                      // ログインパスワード

  IdSMTP1.IOHandler := IdSSLIOHandlerSocketOpenSSL1;     // SMTPへSSLを設定
  IdSMTP1.UseTLS    := utUseExplicitTLS;                 // 接続方式

  // SSL設定
  IdSSLIOHandlerSocketOpenSSL1.Host := IdSMTP1.Host;     // サーバー名
  IdSSLIOHandlerSocketOpenSSL1.Port := IdSMTP1.Port;     // ポート番号
  IdSSLIOHandlerSocketOpenSSL1.Destination :=
    IdSMTP1.Host + ':' + IntToStr(IdSMTP1.Port);         // 接続設定

  // SMTPサーバー接続
  IdSMTP1.Connect;

  IdMessage1.Clear;

  IdMessage1.Recipients.EMailAddresses := Edit1.Text;    // 送信先メールアドレス
  IdMessage1.Subject                   := Edit2.Text;    // 件名
  IdMessage1.Body.Text                 := Memo1.Text;    // 本文

  // メール送信
  IdSMTP1.Send(IdMessage1);

  // SMTPサーバー接続解除
  IdSMTP1.Disconnect;

end;
```

IdMessage1のプロパティ設定

【図8】

【図9】

【図10】

・TIdMessageコンポーネントのプロパティ追加設定

プロパティ名	設定値
CharSet	'UTF－8'
ContentTransferEncoding	'BASE64'

【ソース3】

```
{*******************************************************************************
 目的 : 送信ボタン押下時処理
 引数 :
 戻値 :
*******************************************************************************}
procedure TForm1.Button1Click(Sender: TObject);
begin

  // SMTPサーバー接続設定
  IdSMTP1.Host     := 'smtp.gmail.com';                  // サーバー名
  IdSMTP1.Port     := 465;                               // ポート番号
  IdSMTP1.Username := 'MigaroIndySample';                // ログインユーザーID
  IdSMTP1.Password := 'xxxxxxxxxx';                      // ログインパスワード

  IdSMTP1.IOHandler := IdSSLIOHandlerSocketOpenSSL1;     // SMTPへSSLを設定
  IdSMTP1.UseTLS    := utUseExplicitTLS;                 // 接続方式

  // SSL設定
  IdSSLIOHandlerSocketOpenSSL1.Host := IdSMTP1.Host;     // サーバー名
  IdSSLIOHandlerSocketOpenSSL1.Port := IdSMTP1.Port;     // ポート番号
  IdSSLIOHandlerSocketOpenSSL1.Destination :=
    IdSMTP1.Host + ':' + IntToStr(IdSMTP1.Port);         // 接続設定

  // SMTPサーバー接続
  IdSMTP1.Connect;

  IdMessage1.Clear;

  IdMessage1.CharSet                  := 'UTF-8';        // 文字セット
  IdMessage1.ContentTransferEncoding  := 'BASE64';       // エンコーディング

  IdMessage1.Recipients.EMailAddresses := Edit1.Text;    // 送信先メールアドレス
  IdMessage1.Subject                   := Edit2.Text;    // 件名
  IdMessage1.Body.Text                 := Memo1.Text;    // 本文

  // メール送信
  IdSMTP1.Send(IdMessage1);

  // SMTPサーバー接続解除
  IdSMTP1.Disconnect;

end;
```

IdMessage1のプロパティ設定に新たに追加する

【図11】

【ソース4】

```
{*****************************************************************************
 目的 : 画面表示時処理
 引数 :
 戻値 :
*****************************************************************************}
procedure TForm1.FormShow(Sender: TObject);
begin

  // SMTPサーバー接続設定
  IdSMTP1.Host     := 'smtp.gmail.com';                  // サーバー名
  IdSMTP1.Port     := 465;                               // ポート番号
  IdSMTP1.Username := 'MigaroIndySample';                // ログインユーザーID
  IdSMTP1.Password := 'xxxxxxxxxx';                      // ログインパスワード

  IdSMTP1.IOHandler := IdSSLIOHandlerSocketOpenSSL1;     // SMTPへSSLを設定
  IdSMTP1.UseTLS    := utUseExplicitTLS;                 // 接続方式

  // SSL設定
  IdSSLIOHandlerSocketOpenSSL1.Host := IdSMTP1.Host;     // サーバー名
  IdSSLIOHandlerSocketOpenSSL1.Port := IdSMTP1.Port;     // ポート番号
  IdSSLIOHandlerSocketOpenSSL1.Destination :=
    IdSMTP1.Host + ':' + IntToStr(IdSMTP1.Port);         // 接続設定

  // SMTPサーバー接続
  IdSMTP1.Connect;

  IdMessage1.Clear;

  IdMessage1.CharSet                    := 'UTF-8';      // 文字セット
  IdMessage1.ContentTransferEncoding    := 'BASE64';     // エンコーディング

  IdMessage1.Recipients.EMailAddresses := Edit1.Text;    // 送信先メールアドレス
  IdMessage1.Subject                   := Edit2.Text;    // 件名
  IdMessage1.Body.Text                 := Memo1.Text;    // 本文

  // ファイルを添付
  TIdAttachmentFile.Create(IdMessage1.MessageParts, 'C:¥AttachmentFile¥SAMPLE.jpg');

  // メール送信
  IdSMTP1.Send(IdMessage1);

  // SMTPサーバー接続解除
  IdSMTP1.Disconnect;

end;
```

ファイル添付処理を追加するためには、Uses節にIdAttachmentFileを追加しなければならない

【図12】

【ソース5】

```
{*****************************************************************************
 目的 : 送信ボタン押下時処理
 引数 :
 戻値 :
*****************************************************************************}
procedure TForm1.Button1Click(Sender: TObject);
var
  TEXT1, TEXT2 : TIdText;
begin

  // SMTPサーバー接続設定
  IdSMTP1.Host     := 'smtp.gmail.com';                    // サーバー名
  IdSMTP1.Port     := 465;                                 // ポート番号
  IdSMTP1.Username := 'MigaroIndySample';                  // ログインユーザール
  IdSMTP1.Password := 'xxxxxxxxxx';                        // ログインパスワード

  IdSMTP1.IOHandler := IdSSLIOHandlerSocketOpenSSL1;       // SMTPへSSLを設定
  IdSMTP1.UseTLS    := utUseExplicitTLS;                   // 接続方式

  // SSL設定
  IdSSLIOHandlerSocketOpenSSL1.Host := IdSMTP1.Host;       // サーバー名
  IdSSLIOHandlerSocketOpenSSL1.Port := IdSMTP1.Port;       // ポート番号
  IdSSLIOHandlerSocketOpenSSL1.Destination :=
    IdSMTP1.Host + ':' + IntToStr(IdSMTP1.Port);           // 接続設定

  // SMTPサーバー接続
  IdSMTP1.Connect;

  IdMessage1.Clear;

  IdMessage1.CharSet                    := 'UTF-8';        // 文字セット
  IdMessage1.ContentTransferEncoding    := 'BASE64';       // エンコーディング

  IdMessage1.Recipients.EMailAddresses := Edit1.Text;      // 送信先メールアドレス
  IdMessage1.Subject                   := Edit2.Text;      // 件名

  IdMessage1.ContentType               := 'multipart/mixed';  // コンテンツタイプ

  // メッセージのみ送信時
  TEXT1 := TIdText.Create(IdMessage1.MessageParts, nil);
  TEXT1.ContentType      := 'text/plain';                  // コンテンツタイプ
  TEXT1.CharSet          := 'UTF-8';                       // 文字セット
  TEXT1.ContentTransfer := 'BASE64';                       // エンコーディング
  TEXT1.Body.Text        := Memo1.Text;                    // 本文

  // ファイルを添付
  TIdAttachmentFile.Create(IdMessage1.MessageParts, 'C:¥AttachmentFile¥SAMPLE.jpg');

  // HTML形式での送信
  TEXT2 := TIdText.Create(IdMessage1.MessageParts, nil);
  TEXT2.ContentType      := 'text/html';                   // コンテンツタイプ
  TEXT2.CharSet          := 'UTF-8';                       // 文字セット
  TEXT2.ContentTransfer := 'BASE64';                       // エンコーディング

  // 本文(HTML)
  TEXT2.Body.Text := '<font size="5" color="#ff0000">文字のサイズと色を指定します</font>' +
                     '<br/><br/>' +
                     '<table border=4 width=250 align=center>' +
                     ' <caption>【テーブルの例】</caption>' +
                     ' <tr bgcolor="#cccccc">' +
                     '  <th><br></th>' +
                     '  <th>列-A</th>' +
                     '  <th>列-A</th>' +
                     '  <th>列-B</th>' +
                     ' </tr>' +
                     ' <tr align=center>' +
                     '  <td>行-1</td>' +
                     '  <td>A1</td>' +
                     '  <td>A1</td>' +
                     '  <td rowspan=2>B1-B2</td>' +
                     ' </tr>' +
                     ' <tr align=center>' +
                     '  <td>行-2</td>' +
                     '  <td>A2</td>' +
                     '  <td>A2</td>' +
                     ' </tr>' +
                     ' <tr align=center>' +
                     '  <td>行-3</td>' +
                     '  <td>A3</td>' +
                     '  <td colspan=2>A3-B3</td>' +
                     ' </tr>' +
                     '</table>';

  // メール送信
  IdSMTP1.Send(IdMessage1);

  // SMTPサーバー接続解除
  IdSMTP1.Disconnect;

end;
```

●通常の本文の設定
・ContentTypeプロパティを
'text/plain'に設定している

●html形式の本文の設定
・ContentTypeプロパティを
'text/html'に設定している

htmlで以下の処理を記述している
①色とサイズを指定した文字列の設定
②テーブルの作成

【図13】

【ソース6】

※【ソース3】を参考にしている。

```
{*******************************************************************************
 目的 : 送信ボタン押下時処理
 引数 :
 戻値 :
*******************************************************************************}
procedure TForm1.Button1Click(Sender: TObject);
begin
  // SMTPサーバー接続設定
  IdSMTP1.Host     := 'smtp.mail.yahoo.co.jp';          // サーバー名
  IdSMTP1.Port     := 465;                              // ポート番号
  IdSMTP1.Username := 'MigaroIndySample';               // ログインユーザーID
  IdSMTP1.Password := 'xxxxxxxxxx';                     // ログインパスワード

  IdSMTP1.IOHandler := IdSSLIOHandlerSocketOpenSSL1;    // SMTPへSSLを設定
  IdSMTP1.UseTLS    := utUseExplicitTLS;                // 接続方式

  // SSL設定
  IdSSLIOHandlerSocketOpenSSL1.Host := IdSMTP1.Host;    // サーバー名
  IdSSLIOHandlerSocketOpenSSL1.Port := IdSMTP1.Port;    // ポート番号
  IdSSLIOHandlerSocketOpenSSL1.Destination :=
    IdSMTP1.Host + ':' + IntToStr(IdSMTP1.Port);        // 接続設定

  // SMTPサーバー接続
  IdSMTP1.Connect;

  IdMessage1.Clear;

  IdMessage1.CharSet                  := 'UTF-8';       // 文字セット
  IdMessage1.ContentTransferEncoding  := 'BASE64';      // エンコーディング

  IdMessage1.From.Address             :=
    'MigaroIndySample@yahoo.co.jp';                     // 送信元メールアドレス

  IdMessage1.Recipients.EMailAddresses := Edit1.Text;   // 送信先メールアドレス
  IdMessage1.Subject                   := Edit2.Text;   // 件名
  IdMessage1.Body.Text                 := Memo1.Text;   // 本文

  // メール送信
  IdSMTP1.Send(IdMessage1);

  // SMTPサーバー接続解除
  IdSMTP1.Disconnect;

end;

end.
```

送信元メールアドレスの指定

小杉 智昭

株式会社ミガロ.

システム事業部 プロジェクト推進室

Windowsテキストファイル操作ノウハウ

**他システムとの連携役としてのCSVデータ出力、内部統制対応としてのログ出力。
IBM i上のファイルとは一味違う、Windowsのファイル操作を試みる。**

略歴
1973年05月26日生
1996年関西大学工学部卒
2002年03月株式会社ミガロ. 入社
2002年03月RAD事業部配属
2007年04月システム事業部配属

現在の仕事内容
Delphi/400を利用した受託開発とシステム保守、導入支援を担当している。

1.はじめに

最近は利用者向けソフトの普及や機能向上もあり、システム開発側で利用者の求める出力をすべて取り揃えなくても、ベースとなるCSVデータを用意しておけば、PC上で表計算ソフト等を使って自由に編集して利用するユーザーが増えてきた。

また、複雑化したシステムのエラー原因の究明や利用者の不正利用を防止する目的で、ログ出力機能の重要度が増してきている。そのような意図で出力されるログについては、データ容量が非常に大きくなることもあるので、よりディスク単価の安いPCサーバー等を活用することができる。

上記を受け、本稿では、他システムとの連携役としてのCSVデータ出力、および内部統制対応としてのログ出力に注目し、Delphi/400を使ってWindows上のファイル操作を行う、ということを取り上げる。具体的に、Windowsテキストファイルの操作ノウハウやポイントなどを紹介する。

2.ファイル操作の基本

Delphi/400を使ってWindows上のファイル操作を行うには、同じ内容を実現するとしても、さまざまな方法がある。

この章では、ファイル操作にはまずどのような手法があるのか、またそれぞれの手法の特徴はどういったものなのかを、OSに近いものから順にご紹介する。

・Windows APIを利用するファイル操作
データ型：例）THandle
命令群：例）FileOpen / FileClose / FileCreate

OSに一番近い形のファイル操作方法が、「Windows APIを利用したファイル操作」である。

この方法は「ファイルハンドル」と呼ばれるものを使い、ファイル操作を行う。そのため、ハンドルベースで各種APIを利用できるが、個々のAPIそのものは非常に限定された機能しか持たず、またOSバージョン等に依存することがある。

・低レベル命令群を利用するファイル操作
データ型：例）File / TextFile
命令群：例）AssignFile / CloseFile / Append / Reset / Rewrite

Delphi/400で提供されているファイル操作の命令群の中でも、よりOSに近い（これを指して「低レベル」と呼ばれている）基本命令群を利用する。このファイル操作方法が、低レベル命令群を利用するファイル操作である。

それらはDelphi/400におけるファイル操作の基本となっているが、個々の命令でできる機能は非常に限定されているため、ログ出力のような単純な機能で使用される。

・TStream継承クラスによるファイル操作
クラス：例）TFileStream

ソース1

```
1  procedure TfrmCSVSample.btnExportClick(Sender: TObject);
2  var
3    slFile: TStringList;   // 出力ファイルを示すクラス
4    sLine: String;         // CSVデータの一行を示す文字列
5    i: Integer;
6  begin
7    // TStringListクラスの作成
8    slFile := TStringList.Create;
9    try
10     // データの先頭へ移動
11     cdsSample.First;
12
13     // データの終了まで繰返処理
14     while not cdsSample.Eof do
15     begin
16       // 文字列を初期化する
17       sLine := EmptyStr;
18       // 同一レコードを項目の数だけ繰返処理
19       for i := 0 to cdsSample.FieldCount - 1 do
20       begin
21         // 項目属性毎に出力方法を変更する
22         case cdsSample.Fields[i].DataType of
23           // 文字列：     ダブルクォーテーションで囲む
24           ftString:
25             sLine := sLine + AnsiQuotedStr(cdsSample.Fields[i].AsString, '"');
26
27           // 整数・実数：そのまま文字列化する
28           ftInteger, ftFloat:
29             sLine := sLine + cdsSample.Fields[i].AsString;
30
31         else
32           // 上記以外：   エラーとする
33           sLine := sLine + '#ERROR#';
34         end;
35
36         // 項目毎の区切り文字としてカンマを追加する
37         sLine := sLine + ',';
38       end;
39       // カンマが１つ多くついているため、最後のカンマを削除する
40       Delete(sLine, Length(sLine), 1);
41
42       // 出来上がったCSVデータ（１行分）をStringListクラスに追加する
43       slFile.Add(sLine);
44
45       // 次のデータへ移動
46       cdsSample.Next;
47     end;
48
49     // 全レコード分のCSVデータを'Sample.csv'という名前でファイルに保存する
50     slFile.SaveToFile('Sample.csv');
51
52   finally
53     // TStringListクラスの廃棄
54     slFile.Free;
55   end;
56
57 end;
```

＜コード例①＞

Delphi/400では、さまざまなデータの集合を取り扱うクラスとして、「ストリーム」と呼ばれるクラスが提供されている。その中でも、ファイル操作に特化したのがTFileStreamというクラスになる。

このクラスを使用することで、ファイルのオープン・クローズ制御やファイル作成、モード指定等の煩雑な処理を簡単に実現できる。

また、他のストリーム系クラスとの共通機能を利用することで、メモリ上に展開したデータのファイル保存や、読み込んだファイル内容を通信データにのせてダウンロードする機能を提供するといったことに利用されている。

・TStrings継承クラスによるテキストファイル操作

クラス：例）TStringList

TStrings継承クラスは、どちらかというとファイル操作用ではなく、文字列操作用のクラスであり、それにファイル出力機能が備わっているものである。

しかしながら、CSVデータの編集機能を備えており、利用頻度が高い。そのため、TStrings継承クラスによるテキストファイル操作が行われている。

なお、この具体的な使用方法は、次章でご紹介する。

3.ファイル操作の具体例

前章で挙げたファイル操手法のいくつかを使って、実際にファイル出力する例をご紹介する。

CSVデータ出力の具体例

CSVとはデータの内容をテキスト化し、順番に出力したものである。出力する際、項目ごとの区切りはカンマで、レコードごとの区切りは改行で行う。

古くから利用されているにもかかわらず、詳細な仕様が決定されたのは、2005年と比較的新しいため、仕様確定前から使われていたルールが非常に多く存在する。

例えば、IBM i Client Access等でCSV出力を行うと、文字項目がダブルクォーテーションで囲われる。これをExcelで保存し直すと、通常の文字項目はダブルクォーテーションが外されてしまう。これは仕様の違いによる問題である。

今回は、基本的なCSVのルールに従うが、より互換性を高めるために、文字列を出力する場合は必ずダブルクォーテーションで囲うルールとして実装を考える。

CSVデータ出力の基本的な処理手順は以下のようになる。

【CSVデータ出力の処理手順】
・データの先頭に移動
・データを最終まで繰返処理
**　○項目ごとの繰返処理**
**　　◎項目の種類ごとに異なるルールでテキスト化**
**　　◎カンマを追加**
**　○出力用ワークに1レコード分のCSVデータを追加**
**　○次のデータへ移動**
・全データ分のCSVデータをファイル出力

では実際に、cdsSampleというTClientDataSetのデータを、TStringListクラスを使ってSample.csvファイルに出力する。この例を、ソース1として示す。【ソース1】

forを使った内側の繰り返し処理は、同一レコードの全項目に対する処理である。whileを使った外側の繰り返し処理は、全レコードに対する処理になる。また、網掛けの箇所でCSVファイルを保存している。

CSVファイルの場合、既存のCSVファイルが存在しても、追記せずに既存ファイルを破棄して新規にファイルを作成することが一般的である。そのため、TStringListクラスを使うのが適している。

しかし、すべてのデータをメモリ上に展開するため、あまりに膨大なデータを処理するのは向かない点に注意が必要だ。大量のデータを処理するなら、TFileStream等を利用するほうが適している。

ログ出力の具体例

ログ出力は、何らかの操作やエラー、警告といったものを時系列で絶えず出力していくことが肝要となる。また、複数の機能から同じログファイルに出力したり、前回の続きに出力することが行われる。そのため、処理開始時に既存のログ内容をメモリ上に展開して、処理終了時に編集されたログ内容を出力するといった使い方は避けるべきである。

今回は、「年月日時分秒ミリ秒 + 区切り文字 + 出力メッセージ」という出力フォーマットでログを出力する。

ログ出力の基本的な処理手順は以下のようになる。

【ログ出力の処理手順】
・ログファイルが存在するか確認する
**　○ログファイルが既存であれば追記モードでファイルを開く**
**　○ログファイルが存在しなければ新規作成する**
・出力フォーマットに合わせたログデータを作成する
・ログデータをファイルに出力する
・ログファイルを閉じる

では実際に、低レベルなファイル操作命令群を使ってSample.logファイルに出力する。この例を、ソース2として示す。【ソース2】

出力前にログファイルが既存かどうかを調べ、追記するか、新規作成するかを判断している。また、網掛けの箇所でログファイルに対する出力を行っている。

ログ出力を行う場合、ファイルに対する操作が単純であるため、低レベル命令群でも記述しやすい。また、ここで使った命令は、既存ファイルの読み込みやメモリ上への展開といったことを一切行っていないため、ファイルサイズの影響をほとんど受けない。

以上、CSVデータ出力とログ出力といった2つの機能を、TStrings継承クラスと低レベル命令群を使った手法で実装した。

このサンプルコードは、ファイル名が固定になっている等、紹介用に若干機能を割愛しているが、多少の手直しで十分利用していただけるであろう。

ソース2

```
1  procedure LogOutput(sMsg: String);
2  var
3    logFile: TextFile;
4    sLogMsg: String;
5  begin
6
7    // ファイル名とファイル変数を関連付ける
8    AssignFile(logFile, 'Sample.log');
9
10   // Sample.logファイルが存在するか確認する
11   if FileExists('Sample.log') then
12
13     // Sample.logファイルが存在する場合、追記モードでファイルを開く
14     Append(logFile)
15
16   else
17
18     // Sample.logファイルが存在しない場合、新規作成する
19     Rewrite(logFile);
20
21
22   // 出力フォーマットに合わせた形にメッセージを加工する
23   // 書式：yy/mm/dd hh:nn:ss.zzz> (Message)
24   sLogMsg := FormatDateTime('yyyy/mm/dd hh:nn:ss.zzz', Now) + '> ' + sMsg;
25
26   // 加工済みのメッセージをログファイルに出力する
27   WriteLn(logFile, sMsg);
28
29   // ログファイルを閉じる
30   CloseFile(logFile);
31
32 end;
```

＜コード例②＞

4.ユニコードテキストの出力

Delphi/400は2009以降のバージョンで、ユニコード対応が行われた。しかし、ファイル出力機能は、Delphi/400のバージョンにかかわらずANSIベースで行われる。

つまり、前章で紹介した例を使ってファイル出力すると、ユニコードにしか存在しない文字は「?」に変換されてしまう。

IBM iでは専用のCCSIDを指定する必要があるため、ユニコードが使われない場合もあるが、他システムとの連動等を考慮し、ユニコード対応する例も紹介しておこう。

ユニコードテキストの種類

そもそも「ユニコード」とは一体何だろうか。

この問いに正確に答えることは意外と難しい。非常に乱暴な表現となるが、“これまでコンピュータ上で扱えなかったたくさんの文字の集まりとその扱い方をまとめたルール”これを指してユニコードと呼ばれることが多いようである。

しかし、上記したように文字の集まりとその扱い方ルールという2種類を指しており、また、それぞれが複数の種類に分かれている。UCS-2やUTF-16といった用語を聞いたことはないだろうか。それらが、ユニコードと呼ばれるものの正体に近い。

コラム的に少しだけ細かい説明をしたいと思う。プログラムには直接関係がなく、若干複雑な内容であるため、UTF-8とUTF-16、エンディアンという3つの用語だけ覚えておけば、以下の2段落（※）は読み飛ばしてもかまわない。

※その1

まず、文字の集まりとしてのユニコードだが、厳密にはUnicode、UCS-2、UCS-4といったものがある。このうちUnicodeは、2013年8月現在でバージョン6.2まで拡張されている。現時点では、取り扱い可能な文字の多さで並べれば、UCS-2 ＜ Unicode ＜ UCS-4となるが、主流はUnicodeである。

※その2

続いて、文字の取り扱いルールとしてのユニコードだが、厳密にはUTF-8、UTF-16、UTF-32といったものがある。これらは本来、ユニコードを取り扱うルールであって、ユニコードそのものではない。ただし、実際のプログラムにはこちらの影響が大きく、開発者サイドには馴染みのあるものになっているだろう。

インターネットでは、アルファベット等でASCIIと互換性のあるUTF-8が主流であるが、漢字や仮名をUTF-8で取り扱うとほとんどが3バイト必要になるため、従来のShift_JISに比べても効率が悪くなってしまうことが多い。

そのためか、ユニコード対応版のDelphi/400は、文字データを基本的にUTF-16で取り扱っている。

UTF-16の場合、かなり特殊な文字以外はすべて2バイトで表現するようになっている。多バイトデータになるため、これをファイルに保存する際にはCPUアーキテクチャの影響を受けることになる。いわゆるエンディアンの問題である。エンディアンを明示的に指定しなければ、リトルエンディアンとなる。

上記のように、一口にユニコードと言ってもいろいろな種類があり、テキスト保存の際にはどういった形式で保存すればいいのか気を付ける必要がある。しかし、よく使われる形式は、UTF-8またはUTF-16のリトルエンディアンであろう。

ちなみに、Windows標準のメモ帳でUnicodeを選択すると、UTF-16のリトルエンディアンになる。

では、次から、前章のデータ出力例のユニコード対応、具体的にはUTF-16のリトルエンディアンで保存するケースを示していこう。

なお、ユニコード対応が行われたDelphi/400 2009以降のバージョンを前提としている。

CSVデータ出力例をユニコード対応する

TStringListは、エンコーディングを指定するだけでユニコード対応が可能である。修正したコード例を、ソース3として示す。【ソース3】

コード例に示したように、TStringListクラスのSaveToFileメソッドにパラメーターを追加するだけである。

もし、UTF-8で保存したければ、50行目にあるSaveToFileメソッドの第2パラメーターをTEncoding.UTF8に変更するだけである。

ログ出力例をユニコード対応する

低レベル命令群でユニコードの出力を行うのは若干面倒なことと、TStream継承クラスを使ったサンプルプログラムも示しておきたいので、こちらのユニコード対応はTFileStreamクラスを使用するように変更する。

基本的な処理手順は、前述の低レベル命令群を使ったものと同じである。ただし、ユニコード化の影響として、以下のポイントに気を付ける必要がある。

- **UTF-16のテキストには、BOM（バイトオーダーマーク）が必要となる。**
- **出力データを、バイト配列に変換してから出力する。**

TFileStreamクラスを使用したコード例をソース4として示す。【ソース4】

低レベル命令群からのTFileStreamへの置き換えとユニコード対応を同時に行ったため、大きく変わってしまったが、ボリュームは全体でも50行未満とそれほどでもないことがわかる。

ポイントとしては、TEncodingを使い、ログファイルを新規作成する場合にBOMを付加することと、出力したいメッセージを予め変換しておくことである。また、網掛けの箇所でメッセージの変換と出力を行っている。

なお、こちらの例をUTF-8に対応させるには、12行目のENC変数への代入をTEncoding.UTF8にするだけである。

5.まとめ

今回は、Windows上のテキストファイルを取り扱う代表的なパターンを2つ紹介した。さまざまなシステムが入り乱れる昨今、他システムとの連携役として

ソース3

```
procedure TfrmCSVSample.btnExportClick(Sender: TObject);
var
  slFile: TStringList;   // 出力ファイルを示すクラス
  sLine: String;         // CSVデータの一行を示す文字列
  i: Integer;
begin
  // TStringListクラスの作成
  slFile := TStringList.Create;
  try
    // データの先頭へ移動
    cdsSample.First;

    // データの終了まで繰返処理
    while not cdsSample.Eof do
    begin
      // 文字列を初期化する
      sLine := EmptyStr;
      // 同一レコードを項目の数だけ繰返処理
      for i := 0 to cdsSample.FieldCount - 1 do
      begin
        // 項目属性毎に出力方法を変更する
        case cdsSample.Fields[i].DataType of
          // 文字列：    ダブルクォーテーションで囲む
          ftString:
            sLine := sLine + AnsiQuotedStr(cdsSample.Fields[i].AsString, '"');

          // 整数・実数：そのまま文字列化する
          ftInteger, ftFloat:
            sLine := sLine + cdsSample.Fields[i].AsString;

        else
          // 上記以外：  エラーとする
          sLine := sLine + '#ERROR#';
        end;

        // 項目毎の区切り文字としてカンマを追加する
        sLine := sLine + ',';
      end;
      // カンマが１つ多くついているため、最後のカンマを削除する
      Delete(sLine, Length(sLine), 1);

      // 出来上がったCSVデータ（１行分）をStringListクラスに追加する
      slFile.Add(sLine);

      // 次のデータへ移動
      cdsSample.Next;
    end;

    // 全レコード分のCSVデータを'Sample.csv'という名前でファイルに保存する
    slFile.SaveToFile('Sample.csv', TEncoding.Unicode);

  finally
    // TStringListクラスの廃棄
    slFile.Free;
  end;

end;
```

＜コード例③＞

の CSV データ出力、および内部統制対応としてのログ出力の需要や重要度はますます高まってくると思われる。

また、Delphi/400 では、上記以外の固定長テキストや ini ファイルといったテキスト形式のファイルだけでなく、バイナリ形式のファイルも簡単に取り扱うことが可能である。

ぜひこの機会に、IBM i 上のファイルとは一味違う、Windows のテキストファイル操作を試していただければ幸いである。

M

ソース4

```
procedure LogOutput(sMsg: String);
const
  cCRLF = #$000D + #$000A;
var
  ENC: TEncoding;
  fslog: TFileStream;
  Preamble, Buffer: TBytes;
  sLogMsg: String;
begin

  // エンコーディングを変数化する(UTF-16 リトルエンディアン)
  ENC := TEncoding.Unicode;

  // Sample.logファイルが存在するか確認する
  if FileExists('Sample.log') then
  begin

    // Sample.logファイルが存在する場合、ファイルを開き、最後尾に位置付けする
    fslog := TFileStream.Create('Sample.log', fmOpenReadWrite);
    fslog.Position := fslog.Size;

  end
  else
  begin

    // Sample.logファイルが存在しない場合、新規作成し、BOMを付加する
    fslog := TFileStream.Create('Sample.log', fmCreate);
    Preamble := ENC.GetPreamble;
    if Length(Preamble) > 0 then
      fslog.WriteBuffer(Preamble[0], Length(Preamble));

  end;

  // 出力フォーマットに合わせた形にメッセージを加工する
  // 書式：yy/mm/dd hh:nn:ss.zzz> (Message)
  sLogMsg := FormatDateTime('yyyy/mm/dd hh:nn:ss.zzz', Now) + '> ' + sMsg;

  // 加工済みのメッセージをログファイルに出力する
  Buffer := ENC.GetBytes(sLogMsg + cCRLF);
  fslog.WriteBuffer(Buffer[0], Length(Buffer));

  // ログファイルを閉じる
  fslog.Free;;

end;
```

＜コード例④＞

吉原 泰介 / 國元 裕二

株式会社ミガロ.

RAD事業部 技術支援課 顧客サポート

JC/400 Webアプリケーションのユーザー管理・メニュー管理活用術

大人数で Web システムを利用する際、運用管理はユーザー数、アプリケーション数に比例して意外と大きな作業となってしまう。その作業軽減を図る。

略歴　吉原 泰介
1978 年 03 月 26 日生
2001 年龍谷大学法学部卒
2005 年 07 月株式会社ミガロ. 入社
2005 年 07 月システム事業部配属
2007 年 04 月 RAD 事業部配属

現在の仕事内容
Delphi/400 と JC/400 の製品試験、および月 100 件に及ぶ問い合わせやサポート、セミナー講師などを担当している。

略歴　國元 祐二
1979 年 03 月 27 日生
2002 年追手門学院大学文学部アジア文化学科卒
2010 年 10 月株式会社ミガロ. 入社
2010 年 10 月 RAD 事業部配属

現在の仕事内容
JC/400、SmartPad4i、および Business4Mobile の製品試験やサポート業務などを担当している。

1.Webアプリケーションの活用スタイル

Web アプリケーションはここ数年で、社外向けの公開サイトや BtoB システムだけでなく、社内基幹システムの一部でも活用されるようになってきた。

Web アプリケーションは、ブラウザさえあれば端末の設定なしに利用できる点が大きな特徴である。そのため、社外からも手軽に利用することができ、また利用端末が多い場合などはシステム運用面からも Web アプリケーションが採用されることが増えてきた。

こうした背景から、Web アプリケーションは、C/S（クライアント・サーバー型）アプリケーションと比べて、比較的大人数で利用するシステムで使われることが多い。

また、Web アプリケーションはブラウザだけで利用できる反面、ユーザー端末にはプログラムのインストールや設定を行わない。そのため、アプリケーションの管理は、ログオンするユーザーやメニューで制御することが重要になってくる。

そこで今回は、JC/400 Web アプリケーションを利用する際のユーザー管理、メニュー管理およびログオン手法の活用機能を紹介する。

2.ユーザー管理・メニュー管理

はじめに、JC/400 でのユーザー管理方法について紹介する。

JC/400 でのユーザー管理は、IBM i のユーザープロファイルをそのまま活用することができる。そのため、例えば IBM i の 5250 画面のアプリケーションで使用している場合であれば、同じユーザー / パスワードで制御することができるので、運用面でも管理しやすい。

また JC/400 では、IBM i 上で 5250 の管理メニューが用意されているため、Web サーバー等で細かいユーザー制御をする必要はない。管理メニューは、IBM i のエミュレーター上で「CALL JACI400/JACI400」コマンドで起動することができる。

このメインメニュー画面では大きく、次の管理を行うことができる。【図 1】

【メインメニュー】
1. アプリケーション処理
2. メニューの処理
3. ユーザーメニューの処理
4. ライセンスの処理

ユーザー管理とメニュー管理に関係するのは「2. メニューの処理」と「3. ユーザーメニューの処理」である。

ユーザーメニューの処理

先に「3. ユーザーメニューの処理」から説明する。

ここでは、ログオンで使用するユーザープロファイルを登録できる。【図 2】

ユーザープロファイルを登録すると、そのユーザーが JC/400 でログオンして使用できるアプリケーションをメニュー

図1

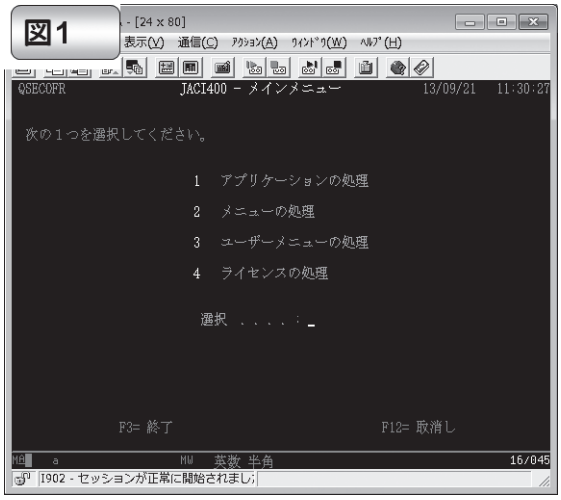


図1
- [24 x 80]
表示(V) 通信(C) アクション(A) ウィンドウ(W) ヘルプ(H)
QSECOFR JACI400 - メインメニュー 13/09/21 11:30:27
次の１つを選択してください。
1 アプリケーションの処理
2 メニューの処理
3 ユーザーメニューの処理
4 ライセンスの処理
選択 . . . . :
F3= 終了 F12= 取消し
MW 英数 半角 16/045
I902 - セッションが正常に開始されました。

図2


図2
- [24 x 80]
表示(V) 通信(C) アクション(A) ウィンドウ(W) ヘルプ(H)
JACI400 : ユーザーの追加 13/09/21 11:58:29
選択項目を入力し、実行キーを押して下さい。
USER PROFILE : MIGARO
F3= 終了 F12= 取消し
MW 英数 半角 05/028
I902 - セッションが正常に開始されました。

図3

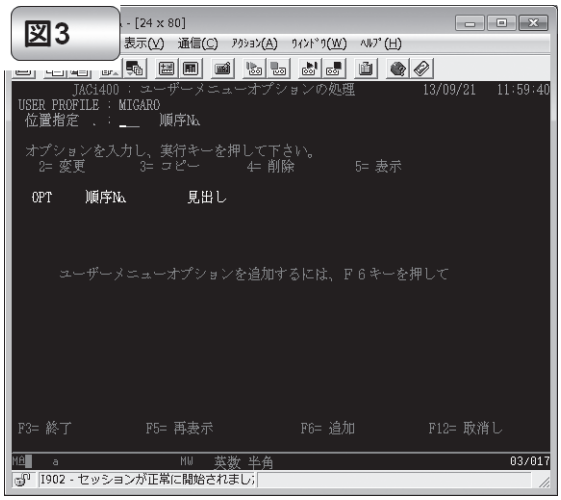


図3
- [24 x 80]
表示(V) 通信(C) アクション(A) ウィンドウ(W) ヘルプ(H)
JACI400 : ユーザーメニューオプションの処理 13/09/21 11:59:40
USER PROFILE : MIGARO
位置指定 . : 順序№
オプションを入力し、実行キーを押して下さい。
2= 変更 3= コピー 4= 削除 5= 表示
OPT 順序№ 見出し
ユーザーメニューオプションを追加するには、Ｆ６キーを押して
F3= 終了 F5= 再表示 F6= 追加 F12= 取消し
MW 英数 半角 03/017
I902 - セッションが正常に開始されました。

に登録していく。【図3】

・アプリケーション登録

アプリケーション登録は、F6で図4の「ユーザーメニューオプションの作成」画面に遷移して指定を行う。指定する内容は、以下のようになる。【図4】

①「メニュー番号」で、メニューの順番を設定する。
②「起動アプリの種類」では、通常JC/400で開発したRPG/COBOLアプリケーションを登録するため、「＊PGM」を設定する。（他の指定については、後述するのでここでは割愛する。）
③「メニュー上の表示」では、見出しに設定した名称で、メニュー画面での表示名を指定できる。
④では、JC/400で開発したRPG/COBOLアプリケーションのプログラム名やライブラリ名、初期プログラム（環境設定CL）を設定する。

ここまでの内容で登録すると、ユーザーメニューに新しいアプリケーションが登録される。【図5】

以上の設定で、JC/400のWebアプリケーションのユーザー登録、メニュー設定は完了である。では実際に、JC/400にログオンしてみよう。

JC/400のログオン画面から登録したユーザープロファイルでログオンを行うと、図5で登録したメニューが自動で表示できる。さらにメニューをクリックすると、図4の⑤「メニュー設定」で指定したアプリケーションが起動する。【図6】

これがJC/400での基本的なユーザーメニューの登録管理方法になる。

ユーザーごとに使用できるアプリケーションを設定できるため、柔軟にメニューを制御することができる。また、運用管理上もユーザープロファイルでIBM i 上の一括管理ができるため、IBM iの機能を有効に活用できる。

例えば、ユーザープロファイル自体を無効に設定すれば、JC/400でもログオンを規制することができる。また、ライブラリやファイルのアクセスもユーザープロファイルの権限範囲で制約が有効なので、セキュリティ的にも安心できる。

・起動アプリケーションの種類

ここで、図4の②「起動アプリの種類」について、便利な機能を追加で紹介しておきたい。

先の説明ではRPG/COBOLアプリケーションを「＊PGM」として登録したが、ファイルサーバーやWebサーバー上のファイルを起動する場合の「＊PCFILE」やWebサイトなどのURL「＊URL」を登録することもできる。

設定方法は、図4の⑥「起動ファイル」に、ファイルパスやURLを設定するだけである。

これらの機能を活用すれば、JC/400でログオンしたメニューから、サーバー上に配置しているPDFやExcel資料を起動したり、EXEアプリケーションを起動することができる。また、ホームページや他のWebアプリケーションを、メニューから起動・連携するといった使い方も可能である。【図7】

メニューの処理

次に「2. メニューの処理」を取り上げ、メニューの管理や運用について説明する。

Webアプリケーションは大人数で運用する場合も多い。そうした場合、例えば100ユーザー分のユーザープロファイルに対して、個別にアプリケーションの設定登録をしていく作業していてはかなりの時間がかかってしまう。

もちろんJC/400には、ユーザー登録情報をメニューごとコピーできる機能を備えている。しかし、登録後にアプリケーションを追加・削除する際には、やはりユーザー数分の手間がかかってしまう。

そこで、大人数のユーザーメニューを管理する便利な機能として、前述の図1の管理メニューから「2. メニューの処理」を使用する。

・メニューグループ

この機能で、先の「3. ユーザーメニューの処理」のアプリケーション登録、メニュー設定と同じように、「2. メニューの処理」から「メニューの作成」画面でメニューグループの作成を行う。【図8】

作成方法は、これも先の「ユーザーメニューオプションの作成」と同様の項目内容で、「メニューオプションの作成」になっている。【図9】

具体的には、この機能では、登録したアプリケーションを、メニューグループとして作成しておくことができるのである。

そして、作成したメニュー（グループ）を利用するには、ユーザーにアプリケーションを登録する際に、図4の②「起動アプリの種類」で「＊MENU」を設定する。また、⑤「メニュー設定」には、作成するメニュー（グループ）名を指定する。

なお、図10のように「ユーザーメニューの変更」画面からも設定できる。【図10】

では、JC/400にログオンしたときのメニュー表示を見てみよう。

図11のように「＊MENU」として登録されたメニューは、1つのメニューグループとして実装されており、クリックするとメニューグループを展開することができる。【図11】

このように、メニューグループで登録をしておけば、ユーザープロファイルごとに複数アプリケーションの登録作業を行う必要がなくなる。作成したメニューグループを設定するだけで、同じメニューを設定することができる。

もちろんアプリケーションを追加・変更する場合は、このメニューグループ自体を変更すれば、すべてのユーザーに適用される。

以上のように、JC/400ではこうしたユーザー管理・メニュー管理を提供している。それらを活用することで、Webアプリケーションで大人数のユーザー管理する場合でも、運用の手間がかからないようになっている。

3.ログオン方式とメニュースキップ機能

ここまでユーザー管理とメニュー管理の方法について紹介してきたが、ここからはIBM i へのログオン方式について説明する。

ログオンには大きく2種類の方式がある。

図4

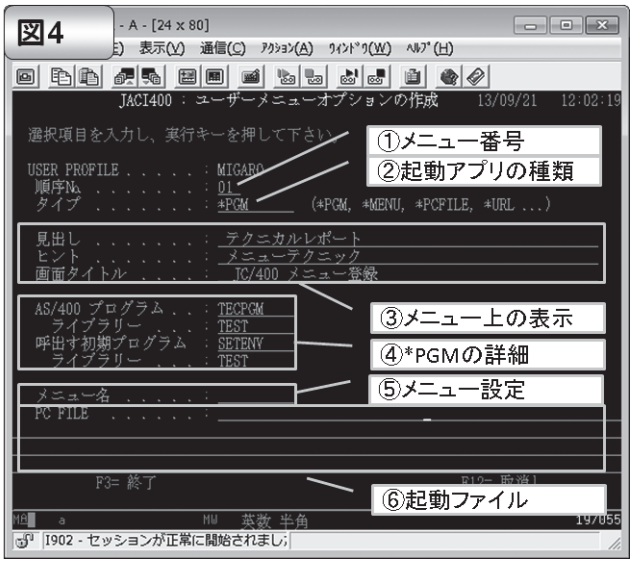


図4
JACI400 : ユーザーメニューオプションの作成
①メニュー番号
②起動アプリの種類
③メニュー上の表示
④*PGMの詳細
⑤メニュー設定
⑥起動ファイル

図5

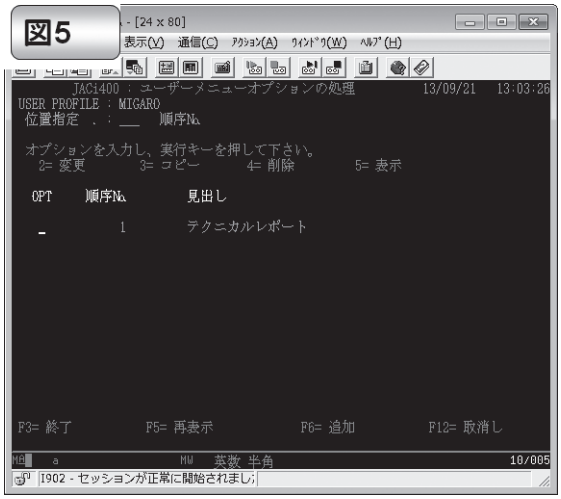


図5
JACi400 : ユーザーメニューオプションの処理
テクニカルレポート

図6


図6
JC/400ログオン画面
JACi400 Demonstration
登録したユーザープロファイルでログオン
JC/400メニュー画面
登録したアプリケーションがメニュー表示
テクニカルレポート

(1) 明示ログオン方式：ユーザー / パスワードを入力してログオンする
(2) 暗黙ログオン方式：固定のユーザー / パスワードで自動ログオンする

通常は（1）明示ログオン方式で、ユーザープロファイルによるログオンを行う。しかし、例えば取引先とのBtoBシステムなどで外部公開の運用をする場合には、取引先ユーザーごとにユーザープロファイルを用意することがセキュリティ的に難しい場合がある。

JC/400ではこうした場合に、(2) 暗黙ログオン方式で、IBM iへの接続認証を行うことができる機能がある。

JC/400では、オートログオン機能（自動ログオン）が用意されているため、これを利用すればユーザープロファイルの接続認証をパスすることができる。

・オートログオンの使用方法

オートログオン機能を使用する方法を説明する。

Webサーバー上の「JC400」フォルダーには、Signon.txtという設定ファイルが用意されている。

このSignon.txtにユーザープロファイルの情報（ユーザー / パスワード）を設定して保存しておけば、そのユーザープロファイルを利用して暗黙ログオン方式で運用することができる。

オートログオン機能でログオンする場合には、通常のログオン画面ではなく、オートログオン用画面を使用するため、起動するURLは変更する必要がある。【図12】

(1) 明示ログオン方式：ログオン画面URL
http://Webサーバー /jaci400/exec/jacilogon.html
(2) 暗黙ログオン方式ログオン画面URL
http://Webサーバー /jaci400/exec/jaciautologon.html

もちろんオートログオン画面は、HTMLで自由にデザインを変更することができる。

・メニュースキップの手法

最後に補足として、ログオンに関連して、メニューをスキップする手法を紹介する。

ログオンするユーザーに対して、メニューのアプリケーションを1つしか登録していない場合、メニュー画面は不要になる。

こうした場合には、Webサーバーの JACi400 Servlet Engine Administratorを起動し、「メニューなしの単一アプリケーション」をチェックしておくと、メニュー画面をスキップすることができる。【図13】

この設定後にログオンすると、メニュー画面を起動せずに、直接アプリケーション画面を起動することができる。【図14】

・URLから直接ログオン実行

JC/400では、こうしたログオン画面からのログオンだけでなく、URLから直接ログオンを実行できるインターフェースも備えている。

下記のURL指定でブラウザからアクセスすると、ログオンやメニューを省略して、直接アプリケーション画面を起動することができる。

【JC/400アプリケーション直接起動URL】
http://Webサーバー /jaciservlet/jaci400.Logon?
USERID= ユーザー名 &
PASSWD= パスワード &
PGMAPP= プログラム名 &
LIBAPP= ライブラリ名 &
PGMENV= 初期プログラム &
LIBENV= 初期プログラムライブラリ

この手法では、ユーザーメニューで登録するアプリケーション設定内容を、URLパラメーターで指定する方式になっている。

例えば、他のWebシステムからJC/400アプリケーションを連携して起動したり、開発時にメニューをスキップしてテストをする際に、この手法を活用すると便利である。

4.まとめ

今回は、JC/400のユーザー登録やメニュー管理機能から、ログオンの手法について紹介した。

冒頭でも少し説明したが、Webアプリケーションは大人数で運用するシステムで使用されることも多い。大人数でのWebシステムを利用する場合には、今回解説したような運用管理がユーザー数、アプリケーション数に比例して意外と大きな作業となってしまう。

JC/400をお使いいただいている皆様のWebシステム管理において、本稿でご紹介した管理機能が運用作業軽減に少しでもお役立ていただければ幸いである。

なお、本稿ではJC/400を中心に説明をしたが、JC/400のスマートデバイスオプションであるSmartPad4iでもまったく同じ仕組みで管理を行うことができる。

M

図7

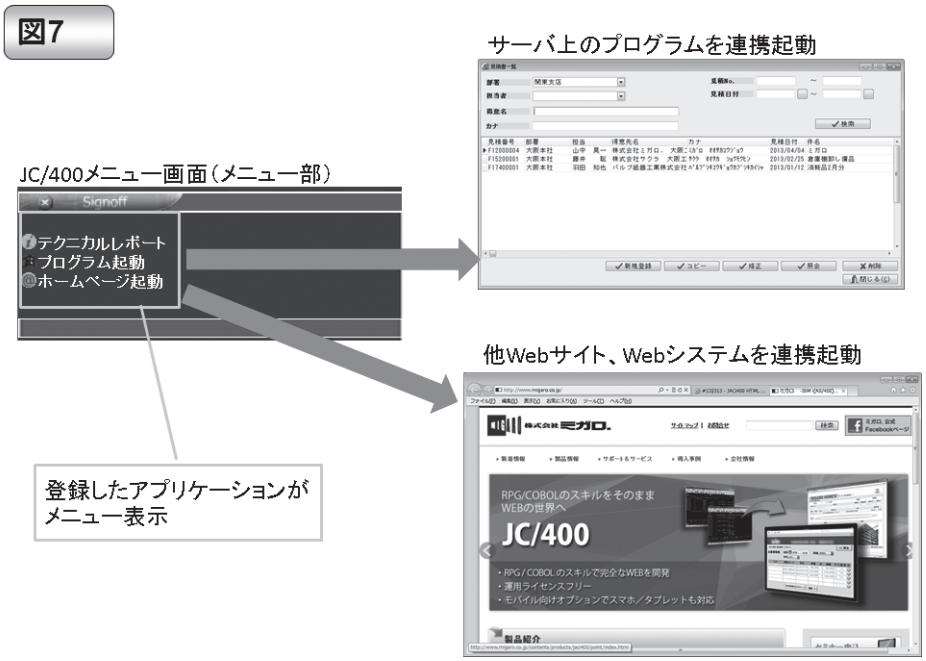


図7
サーバ上のプログラムを連携起動
JC/400メニュー画面（メニュー部）
Signoff
テクニカルレポート
プログラム起動
ホームページ起動
他Webサイト、Webシステムを連携起動
登録したアプリケーションが
メニュー表示
RPG/COBOLのスキルをそのまま
WEBの世界へ
JC/400
・RPG/COBOLのスキルで完全なWEBを開発
・運用ライセンスフリー
・モバイル向けオプションでスマホ／タブレットも対応

図8


図8
- [24 x 80]
表示(V) 通信(C) ｱｸｼｮﾝ(A) ｳｨﾝﾄﾞｳ(W) ﾍﾙﾌﾟ(H)
JACI400 : メニュー作成 13/09/21 14:27:15
選択項目を入力し、実行キーを押して下さい。
メニュー名 TECMENU
F3= 終了 F12= 取消し
MW 英数 半角 05/024
I902 - セッションが正常に開始されまし;

図9

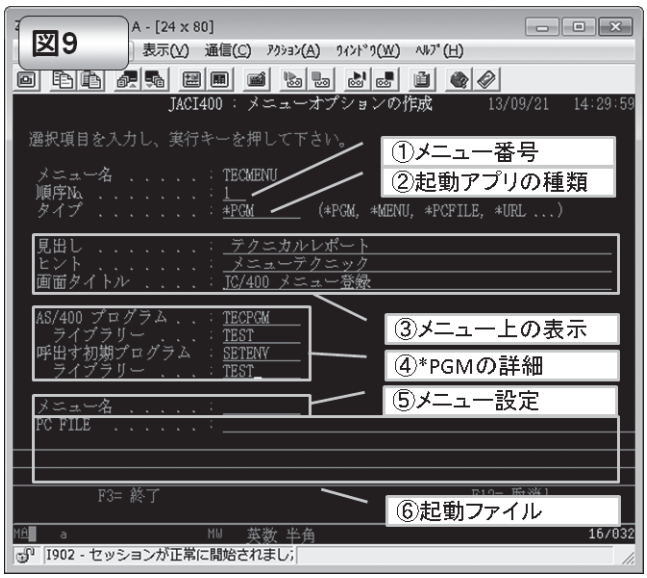


図9
A - [24 x 80]
表示(V) 通信(C) ｱｸｼｮﾝ(A) ｳｨﾝﾄﾞｳ(W) ﾍﾙﾌﾟ(H)
JACI400 : メニューオプションの作成 13/09/21 14:29:59
選択項目を入力し、実行キーを押して下さい。
メニュー名 . . . . . : TECMENU
順序No. . . . . : 1
タイプ . . . . . : *PGM (*PGM, *MENU, *PCFILE, *URL ...)
見出し . . . . . : テクニカルレポート
ヒント . . . . . : メニューテクニック
画面タイトル . . . . : JC/400 メニュー登録
AS/400 プログラム . . . : TECPGM
ライブラリー . . . : TEST
呼出す初期プログラム : SETENV
ライブラリー . . . : TEST
メニュー名 . . . . . :
PC FILE . . . . . :
F3= 終了
①メニュー番号
②起動アプリの種類
③メニュー上の表示
④*PGMの詳細
⑤メニュー設定
⑥起動ファイル
MW 英数 半角 16/032
I902 - セッションが正常に開始されまし;

図10


図10
A - [24 x 80]
表示(V) 通信(C) アクション(A) ウィンドウ(W) ヘルプ(H)
JACI400 : ユーザーメニューの変更 13/09/21 14:44:37
選択項目を入力して、実行キーを押して下さい。
USER PROFILE . . . . . : MIGARO
順序No. . . . . : 4
タイプ . . . . . : *MENU (*PGM, *MENU, *PCFILE, *URL ...)
見出し . . . . . : メニューグループ
ヒント . . . . . :
画面タイトル . . . . :
AS/400 プログラム . . :
ライブラリー . . . :
呼出す初期プログラム :
ライブラリー . . . :
メニュー名 . . . . . : TECMENU
PC FILE . . . . . :
F3= 終了 F12= 取消し
MA a MW 英数 半角 18/044
I902 - セッションが正常に開始されまし;
*MENUで設定
作成したメニューを設定

図11

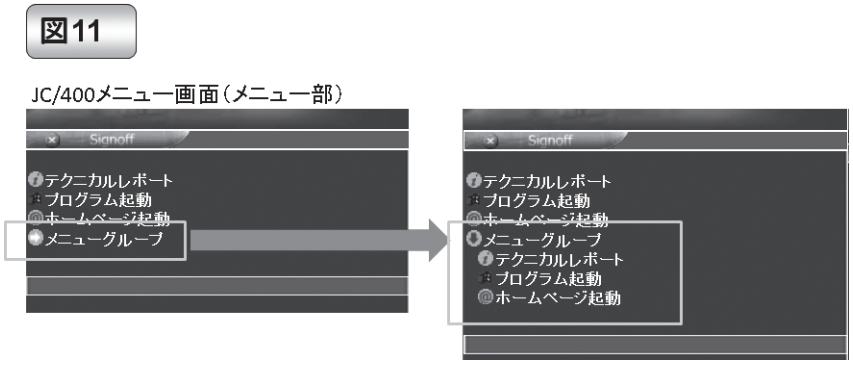


図11
JC/400メニュー画面（メニュー部）
Signoff
テクニカルレポート
プログラム起動
ホームページ起動
メニューグループ
Signoff
テクニカルレポート
プログラム起動
ホームページ起動
メニューグループ
テクニカルレポート
プログラム起動
ホームページ起動

図12

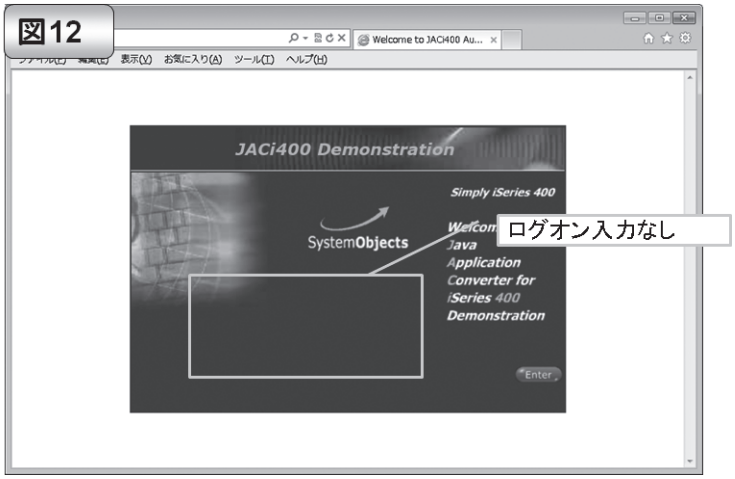


図12
Welcome to JACI400 Au...
表示(V) お気に入り(A) ツール(T) ヘルプ(H)
JACi400 Demonstration
Simply iSeries 400
SystemObjects
Java
Application
Converter for
iSeries 400
Demonstration
Enter
ログオン入力なし

図13

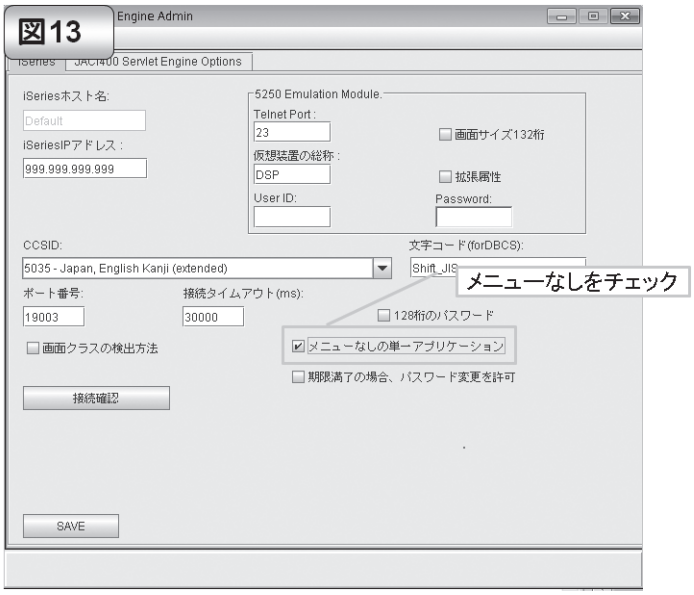


図13
Engine Admin
iSeries
JACi400 Servlet Engine Options
iSeriesホスト名:
Default
iSeriesIPアドレス :
999.999.999.999
5250 Emulation Module
Telnet Port :
23
画面サイズ132桁
仮想装置の総称 :
DSP
拡張属性
User ID:
Password:
CCSID:
5035 - Japan, English Kanji (extended)
文字コード(forDBCS):
Shift_JIS
メニューなしをチェック
ポート番号:
19003
接続タイムアウト(ms):
30000
128桁のパスワード
画面クラスの検出方法
メニューなしの単一アプリケーション
期限満了の場合、パスワード変更を許可
接続確認
SAVE

図14

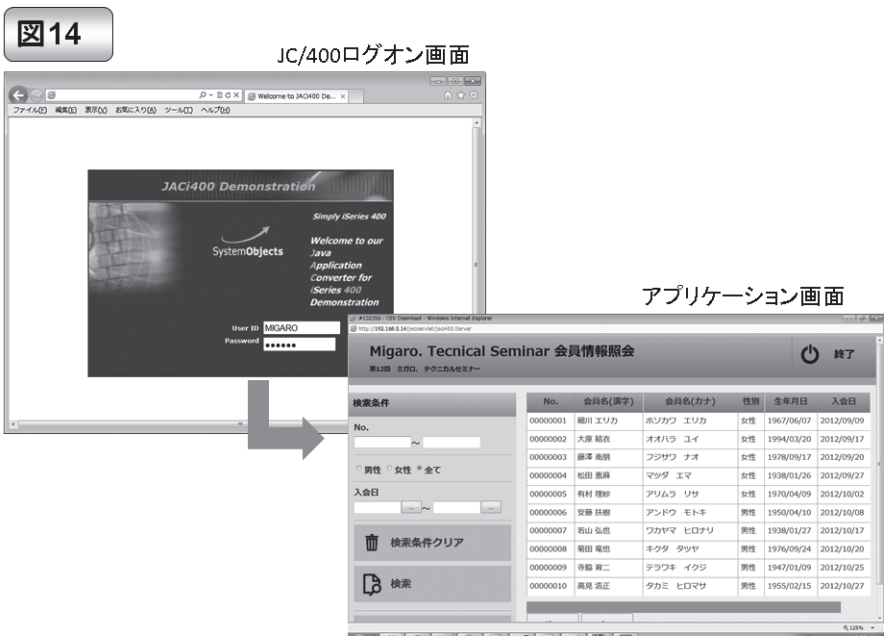


図14
JC/400ログオン画面
JACi400 Demonstration
Simply iSeries 400
SystemObjects
Welcome to our Java Application Converter for iSeries 400 Demonstration
User ID
MIGARO
Password
アプリケーション画面
Migaro. Tecnical Seminar 会員情報照会
終了
検索条件
No.
男性
女性
全て
入会日
検索条件クリア
検索
No.
会員名(漢字)
会員名(カナ)
性別
生年月日
入会日

Migaro. Technical Report
既刊号バックナンバー

電子版・書籍(紙)媒体で提供中!
http://www.migaro.co.jp/contents/support/technical_report/

No.1　2008 年秋

お客様受賞論文

●最優秀賞

直感的に理解できるシステムを目指して―情報の“見える化”の取り組み
石井 裕昭様／豊鋼材工業株式会社

●ゴールド賞

運用部間にサプライズをもたらした Delphi/400
春木 治様／株式会社ロゴスコーポレーション

●シルバー賞

JACi400 使用による Web アプリケーション開発工数削減
中富 俊典様／日本梱包運輸倉庫株式会社

Delphi/400 を利用した Web 受注システム
飯田 豊様／東洋佐々木ガラス株式会社

●優秀賞

Delphi/400 による販売管理システム（FAINS）について
藤田 建作様／株式会社船井総合研究所

技研化成の新基幹システム再構築
藤田 健治様／技研化成株式会社

SE 論文

はじめての Delphi/400 プログラミング
畑中 侑／システム事業部 システム２課

Delphi/400 と Excel との連携
中嶋 祥子／ RAD 事業部 技術支援課

連携で広がる Delphi/400 活用術
尾崎 浩司／システム事業部 システム２課

フォーム継承による効率向上開発手法
吉原 泰介／ RAD 事業部 技術支援課

API を利用した出力待ち行列情報の取得方法
鶴巣 博行／ RAD 事業部 技術支援課

Delphi テクニカルエッセンス Q&A 集
吉原 泰介／ RAD 事業部 技術支援課

JACi400 を使って RPG で Web 画面を制御する方法
松尾 悦郎／システム事業部 システム２課

あなたはブラインドタッチができますか？
福井 和彦／システム事業部 システム１課

No.2　2009 年秋

お客様受賞論文

●最優秀賞

JACi400 で 既存 Web サービスの内製化を実現
佐々木 仁志様／株式会社ジャストオートリーシング

●ゴールド賞

.NET 環境での Delphi/400 の活用
福田 祐之様／林兼コンピューター株式会社

●シルバー賞

5250 で動作する「中古車 在庫照会プログラム」の GUI 化
佐久間 雄様／株式会社ケーユー

●優秀賞

Delphi による 輸入システム「MISYS」の再構築
秦 榮禧様／株式会社モトックス

Delphi/400 による 物流システムの再構築
仲井 学様／西川リビング株式会社

Delphi/400 で開発し 3 台のオフコンを 1 台の IBM i へ統合
島根 英行様／シルフ

SE 論文

JACi400 環境でマッシュアップ！
岩田 真和／ RAD 事業部 技術支援課

Delphi/400 を利用したはじめての Web 開発
福岡 浩行／システム事業部 システム２課

Delphi/400 を使用した Web サービスアプリケーション
尾崎 浩司／システム事業部 システム３課

Delphi/400 によるネイティブ資産の応用活用
吉原 泰介／ RAD 事業部 技術支援課 顧客サポート

RPG でパフォーマンスを制御
松尾 悦郎／システム事業部 システム１課

MKS Integrity を利用したシステム開発
宮坂 優大・田村 洋一郎／システム事業部 システム１課

No.3　2010 年秋

お客様受賞論文

●最優秀賞

建物のクレーム情報管理システム「アフターサービス DB」について

大橋 良之様／東レ建設株式会社

●ゴールド賞

Delphi/400 で「写真管理ソフト」と「スプールファイルの PDF 化ソフト」を自社開発

寒河江 幸喜様／日綜産業株式会社

●シルバー賞

Delphi/400 で鉄鋼受発注業務を統一し 鉄鋼 EDI も実現

柿本 直樹様／合鐵産業株式会社

●優秀賞

Delphi/400 で EIS（Executive Information System）の高速化

小島 栄一様／西川計測株式会社

イントラでの PHP-Delphi-RPG 連携

仲井 学様／西川リビング株式会社

Delphi/400 を使った取引先管理システム

大崎 貴昭様／森定興商株式会社

SE 論文

Delphi/400 ローカルキャッシュ活用術

中嶋 祥子／ RAD 事業部 技術支援課

Delphi/400 帳票開発ノウハウ公開

尾崎 浩司／システム事業部 システム３課

Delphi/400 でドラッグ&ドロップを制御

辻林 涼子／システム事業部 システム２課

Delphi/400 のモジュールバージョン管理手法

前田 和寛／システム事業部 システム２課

Delphi/400　Web からの PDF 出力

福井 和彦・清水 孝将／システム事業部システム３課・システム２課

Delphi/400 で Flash 動画の実装

吉原 泰介／ RAD 事業部 技術支援課 顧客サポート

No.4　2011 年秋［創立 20 周年記念号］

お客様受賞論文

●最優秀賞

全社の経費処理業務を効率化した「e 総務システム」

鈴木 英明様／阪和興業株式会社

●ゴールド賞

「Web 進捗管理システム」でリアルタイム性を実現

堀内 一弘様／エスケーロジ株式会社

●シルバー賞

「営業奨励金申請書」をたった２日間で開発

簑島 宏明様／株式会社ケーユーホールディングス

液体輸送における「配車支援システム」の構築

桂 哲様／ライオン流通サービス株式会社

SE 論文

グラフ活用リファレンス

中嶋 祥子／ RAD 事業部 技術支援課

Web サービスを利用して機能 UP ！

福井 和彦・畑中 侑／システム事業部 システム２課

OpenOffice 実践活用

吉原 泰介／ RAD 事業部 技術支援課 顧客サポート

VCL for the Web 活用 TIPS 紹介

尾崎 浩司／システム事業部 プロジェクト推進室

JC/400 で JavaScript 活用

清水 孝将／システム事業部 システム１課

jQuery 連携で機能拡張

國元 祐二／ RAD 事業部 技術支援課 顧客サポート

No.5　2012 年秋［創刊 5 周年記念］

お客様受賞論文

【部門 1】

●最優秀賞

JC/400 による取引先との Web-EDI システム構築

久保田 佳裕様／極東産機株式会社

●ゴールド賞

Delphi と Excel を使用した帳票コストの削減

大久保 治高様／合鐵産業株式会社

もっと見やすく、もっと使いやすい画面を

新谷 直正様／株式会社アダル

【部門 2】

●優秀賞

Delphi/400 で確認業務の効率化

為国 順子様／ベネトンジャパン株式会社

取引先申請システムでの稟議書作成ワークフロー

大崎 貴昭様／森定興商株式会社

Delphi/400 で IBM i のストアードプロシージャを利用し、SQL 処理を高速化

島根 英行様／シルフ

SE 論文

InstallAware を使った Delphi/400 運用環境の構築

中嶋 祥子／ RAD 事業部 技術支援課 顧客サポート

カスタマイズコンポーネント入門 Delphi/400 開発効率向上

前田 和寛／システム事業部 システム 2 課

Delphi/400 スマートデバイスアプリケーション開発

吉原 泰介／ RAD 事業部 技術支援課 顧客サポート

DataSnap を使用した 3 層アプリケーション構築技法

尾崎 浩司／システム事業部 プロジェクト推進室

JC/400 でポップアップウィンドウの制御&活用ノウハウ

清水 孝将・伊地知 聖貴／システム事業部 システム 1 課

【創刊 5 周年記念】

ミガロ.SE 座談会―お客様と共に歩む、お客様への熱い思い

MIGARO.
TECHNICAL
REPORT

Migaro.Technical Report
No.6 2013 年秋
ミガロ.テクニカルレポート

2013 年 11 月 1 日 初版発行

◆発行
株式会社ミガロ.
〒 556-0017
大阪府大阪市浪速区湊町 2-1-57 難波サンケイビル 13F
TEL：06（6631）8601　FAX：06（6631）8603
http://www.migaro.co.jp/

◆発行人
上甲 將隆

◆編集協力
アイマガジン株式会社

◆デザインフォーマット
近江デザイン事務所

MIGARO. TECHNICAL REPORT

ミガロ.テクニカルレポート

株式会社ミガロ.

http://www.migaro.co.jp/

本社
〒556-0017
大阪市浪速区湊町2-1-57
難波サンケイビル 13F
TEL:06(6631)8601
FAX:06(6631)8603

東京営業所
〒106-0041
東京都港区麻布台1-4-3
エグゼクティブタワー麻布台 11F
TEL:03(5573)8601
FAX:03(5573)8602

ミガロ.facebook ページ
http://www.facebook.com/migaro.co.jp