MIGARO. TECHNICAL REPORT

ミガロ.テクニカルレポート

No.7
2014年秋

株式会社ミガロ.

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た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7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。

昨今は、スマートフォンやタブレットを業務で利用することが普通のこととなり、お客様のご要望もますます多様化しております。そのような時代の流れの中で、従来のPC向け開発と同様の手法でモバイル向けネイティブアプリを開発できるDelphi/400 Version XE5を本年7月にリリースすることができました。おかげさまで多くのお客様よりご興味をいただいており、引き続き本製品の情報提供や技術サポートに努めてまいります。

今回も従来と同様に、第1部は「Migaro.Technical Award 2014 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、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当たる「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による生産スケジューラーの再構築」や「Delphi/400を利用して個人別のオンラインメニューを構築した事例」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、「iOS/Android用のネイティブアプリ開発入門」や「JC/400、SmartPad4iでのスマートデバイス開発におけるユーザーインターフェースの工夫」など、さまざまな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7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。

2014年秋

株式会社ミガロ.
代表取締役社長
上甲 將隆

Migaro. Technical Award 2014

お客様受賞論文／ミガロ. テクニカルアワード

最優秀賞

Delphi/400による生産スケジューラの再構築

柿村 実 様

東洋佐々木ガラス株式会社
経営管理部 情報管理課

東洋佐々木ガラス株式会社
https：//www.toyo.sasaki.co.jp/

2002 年に東洋ガラスのハウスウェア部門と佐々木硝子が統合して設立。自社ブランド・オーダーメイド・輸出向けのガラス製ハウスウェア製品の製造・販売を行っている。大量かつ安定した品質のマシンメイド品から、職人が手造りする高級ハンドメイド品まで、幅広い製品を取り扱っている。

1.当社事業の概要

当社の業務内容は、ガラス食器の製造と販売である。当社の製品は、国内の百貨店や量販店､専門店の店頭に並ぶほか、ホテル、飲食店にて業務用の食器として扱われている。また、大手飲料メーカーの求めに応じ、オリジナルグラスの製造も行っており、幅広い顧客を持つ。

そのような幅広い顧客の要求に対応するため、当社の工場は、ガラス生地の生産から加工、最終製品の包装までの一貫生産を行う。ガラス食器を生産する国内では数少ない工場の１つである。機械を使用した製造が主であるが、一部ではハンドメイドでの製造も行っている。1つの工場でこれらすべてを抱えているのは、国内では当社工場のみである。

2.当社の情報システムの概要

発足当初から､基幹系システムとして、IBM i（System i ）を採用している。IBM i の長所は、システム障害が非常に少ないことである。このため、情報システム部門でのシステム維持に費やす作業の負担が少なく、少人数で基幹系システムを運営するのに打ってつけのマシンと考えている。オープン系やPC サーバー、または ASP やクラウドの利用が世間では進展しているが、当社での基幹系システムは、当面は IBM i の利用を続けていく方針である。

3.生産スケジューラ開発当初の背景

幅広い顧客の要求に応えるため、当社の工場では､ガラス生地を製造する工程、ガラス生地に印刷などの加工を施す工程、ガラス生地を箱詰めする包装工程など、さまざまな形態のラインが稼働しており、その数はのべ 50 種類以上にも及ぶ。上記に述べた通り、店頭での販売、ホテルや飲食店での業務用、各飲料メーカーのノベルティなどさまざまな用途の製品が混在し、市場への展開方法もそれぞれで大きく異なる。

当社の発足当初は、営業部門で記入した生産手配書を、それぞれの工程のスケジュール担当が転記入力して、スケジュール表を作成していたが、膨大な件数の生産手配書と、頻繁に発生するスケジュール変更に対応するには、人手では作業の負担が大きい。このことから、生産手配の登録からスケジュール表の作成までのシステム化を目指し「生産計画システム」の開発が始まった。その中で、生産スケジューラのプログラム開発も行うこととなった。

上に述べた、のべ 50 種類を超える稼働ラインの作業日程を調整するのに、5250 エミュレータ画面では 24x80 文字だけしか表示ができないため、一画面に表現できる情報量が限られ使い勝手が非常に悪い。

そこで、ユーザーが日程を操作する部分は、Visual Basic 6 によるプログラム開発を行うこととした。入力した生産手配書と製品展開のデータは IBM i への登録が必要である。

図1　スケジュールの調整画面

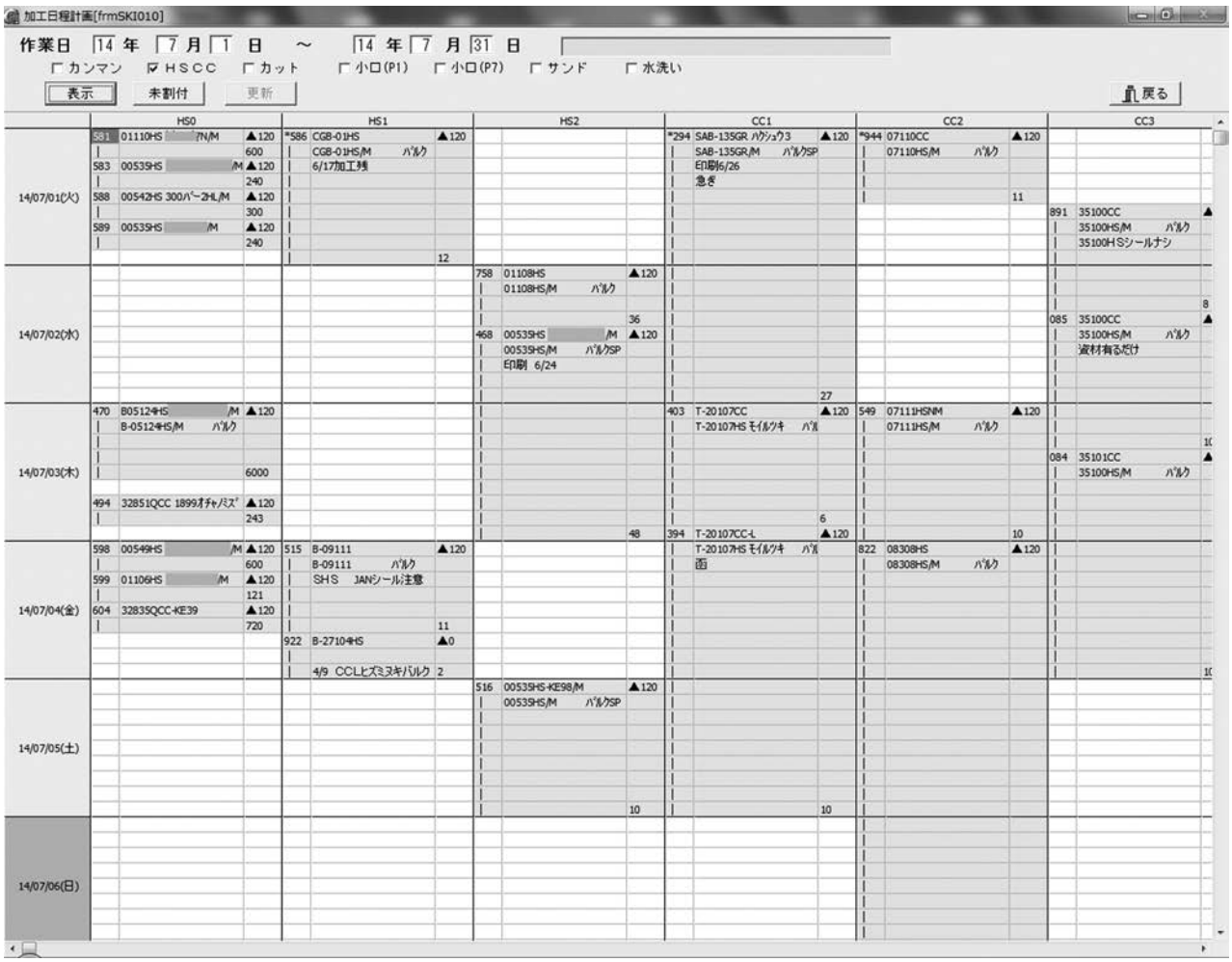


図2　スケジュール移動方法(1)　マウス右クリックから「繰上有り/無し」を指定

まず、5250エミュレータのファイル転送を用いて、IBM iからPCにデータを落とし込む。反対に、Visual Basic 6で開発したスケジューラによりPC上で立案したスケジュール表は、5250エミュレータのファイル転送を使ってIBM iに戻すこととした。この手法を用いて、ガラス生地の製造スケジュール表と加工スケジュール表のシステム化をそれぞれ行うこととした。

4.生産スケジューラの再構築の背景

システム化により、各作業ラインのスケジュール作表作業が簡便化された。特に日程調整がマウス操作だけでできるため、ユーザーから好評であった。

一方で、5250エミュレータのファイル転送を利用したIBM iとのデータの受け渡しでは問題が生じていた。IBM iとPC間で日程データを受け渡す方法は、エミュレータのセッション画面からの指示であるが、ユーザーが誤ってPC内の複数セッションからデータ受け渡しの指示を操作することにより、立案したデータが消失してしまう現象がたびたび発生した。

その都度、バックアップデータを戻し、同じ手配に対して再度スケジューリングをやり直す。生産状況によっては、復旧を急がねばならないケースもあるし、そもそもバックアップデータ量が膨大であり、IBM iのディスクを占有する状況にもなっている。スケジューリング作業の改善はなされたものの、システム維持管理の観点からは、非常に負担がかかる結果となってしまった。

5.Delphi/400の選定過程

旧プログラムではVisual Basic 6で扱うため、IBM iから受信したcsvファイルを、さらにMS Accessのテーブルとしてインポートするという複雑な手順を踏んでいた。この複雑さが、ファイル転送でのデータ消失の一因とも考えられた。

Delphi/400の長所は、一画面に表示できる文字数の制約がないことに加えて、IBM iのファイルを直接操作できることである。Visual Basic 6プログラムの長所である、ユーザーの操作性のよさを維持しながら、短所であったデータの信頼性を改善できるとして、Delphi/400による生産スケジューラの再構築を行うこととした。

6.技術課題

Visual Basic 6プログラムの中でも、マウス操作だけで作業日程を変更する操作部分や、新しい生産手配をスケジュールに割り付ける操作部分はユーザーに好評であり、Delphi/400でも同じ操作を再現できることがユーザーの要望であった。

ここで課題になったのは、生産手配の作業ラインもしくは日程を移動した場合、あとに残った日程を繰り上げるかどうか、反対に生産手配を割り込ませる場合に日程を繰り下げるかどうか、マウスのドラッグ＆ドロップ操作だけではプログラムで判別できないことであった。

クリック1つで、スケジュールが変わってしまうプログラムの動作では、誤動作が大量に発生して、ユーザーが意図した通りのスケジュール編集ができないという結果に結び付く。

同じことは、新しい生産手配を初めてスケジュールに落とし込む操作にもいえ、マウスの位置だけでは、画面上のどの手配が割り付け対象なのかプログラムからは判別できないという問題もあった。

7.解決策

以上のような課題を考慮しながら開発したのが、スケジュールの調整画面（加工日程計画）である。【図1】

割り付け済みスケジュールの日程変更は、割り付けを指定する操作により日程の移動方法を定めることとした。あらかじめ日程の移動方法を決めておけば、日程の移動先へマウスを移動して、ボタンをクリックするだけでも、その後のプログラムの動きは定まる。【図2、図3】

また、新しい生産手配をスケジュールに割り付ける操作については、ユーザーの負担にならない範囲で、キー操作を行うこととした。割り付けたい手配は、チェックボックスに指定して選択する。【図4】

指定した手配の割り付け方法は、ドラッグするだけでなく併せてshiftキーを押すことで、手配を割り付けるための操作であることを明確に表す。操作するキーが増えるものの、ユーザーが許せる範囲内とのことで、この操作方法を採用することとした。【図5】

8.業務課題の解決

今回の再構築にあたっては、個別のDelphi/400プログラムの機能面の整備だけでなく、システムの維持管理についても、併せて負担が少なくなるような取り組みを行った。

今までは、5250エミュレータのファイル転送機能およびVisual Basic 6で開発したスケジューラプログラムを個々のPCにインストールしており、それぞれのバージョンが異なる場合があった。そのためユーザーからプログラムの不具合報告を受けても、現象の確認と再現が困難であった。

今回、Delphi/400プログラムをすべて専用サーバーに保管して、ユーザーがPCを起動する際に、最新のプログラムに自動更新されるようにした。これによりユーザーごとのプログラム稼働環境の違いを取り除くことができるようになった。現在は全ユーザーのDelphi/400プログラムの動作環境はほぼ一致する。システムの維持管理の負担を減らす観点からは非常に効果を発揮している。

9.今後の課題

現在、スケジューラの再構築が実現できたのは、加工工程の日程スケジューラに関してであるが、製造工程のスケジューラについては、5250エミュレータのファイル転送と、Visual Basic 6によるプログラムが残っている。こちらについても、Delphi/400による再構築を行いたいと考えている。

M

図3　スケジュール移動方法(2)　スケジュールをマウスで移動し左クリックで確定

図4　手配からスケジュールへの割り付け(割り付け前)

図5　手配からスケジュールへの割り付け(割り付け後)

ゴールド賞

Delphi/400およびDelphiを利用したオンライン個人別メニューの構築
—CUIとGUIの融合による可能性を求めて

小山 祐二 様
澁谷工業株式会社
経営情報システム部
課長代理

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。近年では、無菌化などの技術力を活かし、再生医療事業も積極的に展開している。

はじめに

澁谷工業株式会社（以下、当社）は、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、お客様のニーズに合わせたパッケージングプラントを「ターンキー」で提供するビジネスを主体としている。また最近では「再生医療」事業にも進出している。

当社のホストコンピュータの変遷は、システム/32から始まり、現在のPureFlex Systemに至る。また最近の基幹システムの開発は、GUI（主にDelphi/400およびDelphi）で行っている。しかし今現在でも、5250画面上で起動するレガシー資産が多いのも事実である。

レガシー資産は、5250画面上で規定メニューから実行している。しかし、規定メニューによる各アプリケーション（以下、個別アプリ）の実行に、不満を抱いているエンドユーザー（以下、ユーザー）も少なくない。

なぜなら、規定メニューには、ユーザーによっては利用しない個別アプリも多いからである。またメニュー構造上、かなり下位階層でないと個別アプリが実行できないのも不満の理由の１つであった。

そこで本稿では、「Delphi/400およびDelphiを利用したオンライン個人別メニューの構築」と題し、ユーザーが自分の使い勝手に合わせ、オンラインメニュー構成をユーザー自身で登録可能とするシステムの構築内容をご紹介する。

規定メニューから個人別メニューへ

先にも述べたが、以前は規定メニューから、レガシー資産を実行していた。このレガシー資産は、基本的に5250画面上でのみ実行可能である。ただし、HATSやSC5250/SC5250Panelコンポーネントを利用すれば、擬似的に起動は可能である。しかし、前者は追加投資が必要となり、後者は場合によって対応できない機能もある。

これに対して、個人別メニューの発想は昔から持っていた。しかし、ユーザーが5250画面上で個人別メニューの登録を行うのは、あまりにもユーザービリティが悪いと判断し、着手に至らなかった。

しかし、ある日、ひらめいた。5250画面上で実現できないのならば、他の方法で実現すればよい、と。つまり、5250画面上で実現できないのであれば、Delphi/400およびDelphiを利用してGUIで実現すればよい、と。

個人別メニュー概要

ここで、「オンライン個人別メニュー」の概要を説明する。

【図1】のように、「個人別メニュー登録アプリケーション」（以下、個人別メニュー登録AP）、「個人別メニューアプリケーション」（以下、個人別メニューAP）、および「各種照会アプリケーショ

図1　システム構成概要図

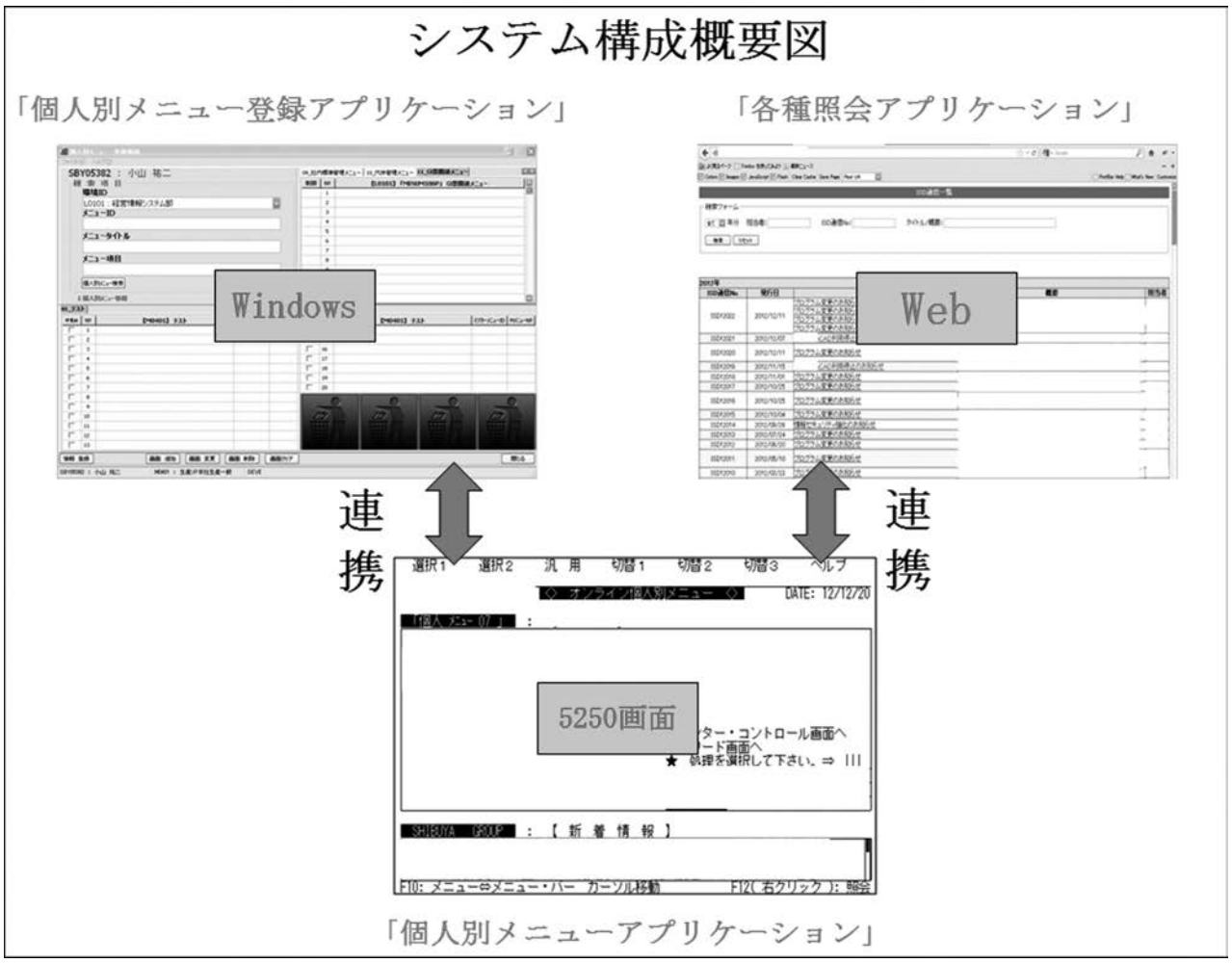
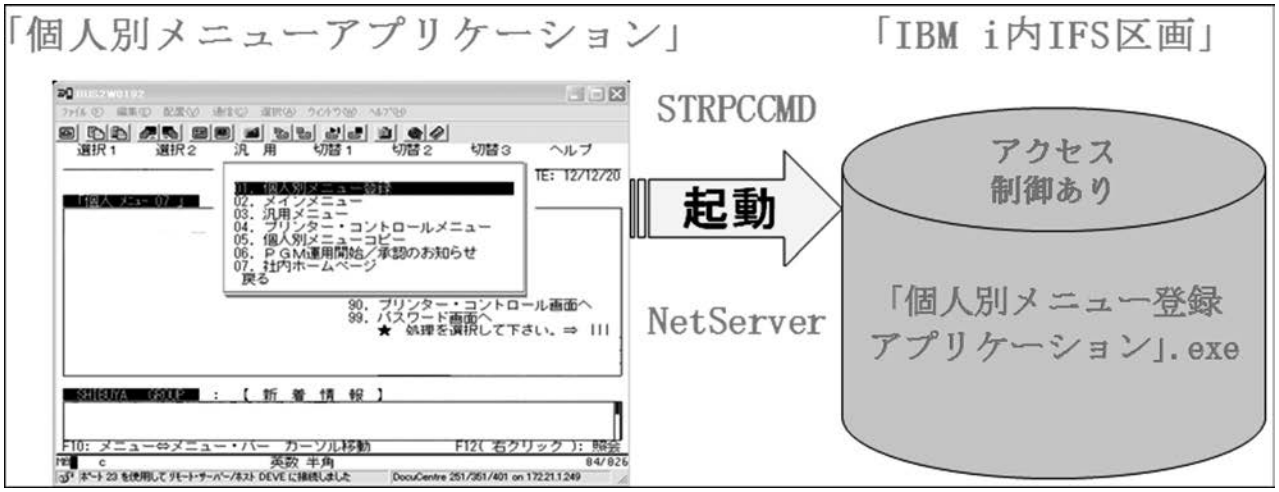


図2　5250メニューからIFS区画内アプリを実行



ン」（以下、各種照会 AP）を連携させてシステムを構成することとした。【図1】

(1) 個人別メニュー登録 AP

これは、使い勝手をよくするために、各ユーザーが使用する個別アプリの個人別メニューを自分自身で登録するものである。「個人別メニュー登録 AP」は、ユーザービリティを重視し、Delphi/400 により Windows 上で作成することとした。

(2) 個人別メニュー AP

これは、「個人別メニュー登録 AP」で登録した各個別アプリを実行するものである。「個人別メニュー AP」はレガシー資産を実行するため、5250 画面上での作成とした。また、「個人別メニュー登録 AP」も本画面から実行する。

(3) 各種照会 AP

今まで静的に管理していた情報をデータベース化し、その内容の閲覧を行うものである。「各種照会 AP」は、既存運用の流れを活かし、Web 上での作成とした。

問題点

しかし、いきなり問題が発生した。それは、どのような方法で 5250 画面から Delphi/400 アプリケーション（個人別メニュー登録 AP）を実行するかである。単純にユーザー所有の全 PC に、Delphi/400 アプリケーションをインストールすれば実現可能である。しかしその場合、アプリケーション管理が非常に難しくなる。特に IBM i の運用に慣れているため、なかなかその方法に踏み切れなかった。

そこでクローズアップされたのが、IFS（※ 1）である。この IFS は通常、ファイルサーバーや他プラットホームからのインターフェースとして利用されることが多い。

しかし、IBM i の運用を先進的に行っている企業様は、IFS 上に PHP や Java を配置していることに気がついた。

そこで、Delphi/400 アプリケーションも同じように利用できないかと考えた。調査した結果、「NetServer」（※ 2）および「STRPCCMD」（※ 3）の組み合わせで、5250 画面上から Delphi を起動できることがわかった。一方、Delphi アプリケーションから IBM i の連携は、Delphi/400 の機能により、まったく問題ない。【図 2】

※ 1　IBM i 上にある UNIX 互換のファイルシステム。Java や exe、Excel などを保管できる
※ 2　IFS をクライアント PC からネットワークドライブとして認識する IBM i のサービス
※ 3　5250 画面から PC コマンドの実行を行う CL コマンド

工夫点

新しいアイデアが次々に膨らむ中、構想が固まり実装に入った。ここで、今回の工夫点を挙げる。

(1)「個人別メニュー登録 AP」のマスター体系を既存メニュー体系と同じとし、該当メニューをクリックすれば、サブメニュー画面の遷移を可能とした。【図3】
(2) 規定メニューでは、実行して初めてわかった実行制御を、一目でわかるようにした。【図 3】
(3) 個人別メニュー登録（マスター → 個人別メニュー、個人別メニュー → ゴミ箱など）をドラッグ＆ドロップで処理可能とした。【図 4】
(4) 個人別メニュー AP では、Windows Like RPG（メニューバー、マウスでのアプリ実行など）や他 DB との連携、Web との連携、アプリ追加・変更・削除のお知らせ機能を追加した。【図 5】
（※各機能の詳細説明は省略）
(5) 個人別メニュー AP で見出し登録を行い、グルーピングを可能とした。【図5】

ユーザーからのリクエスト

構築完了後、ユーザーに対して説明会を開催した。しかし、ユーザーからの反応は今ひとつであった。その理由として、次のようなことがわかった。

構築した仕組みは、1 人で利用するぶんには申し分はない。しかし今のままであれば、他の利用者と会話ができない。つまり、十人十色の個人別メニューでは、同じアプリであっても全く違う場所に配置可能となる。そのため、他の利用者が登録している個人別メニューの内容がまったくわからなくなり、他の利用者がどの個別アプリを利用しているかの説明が、非常に困難となる。

そこで本稼働までに、【図 6】のようなメニュー経路を照会可能とした。また、マスターメニュー ID やメニュー№情報を付加した。そして個人別メニューパターンマスターを展開させて、同じ部署内で同じ内容の個人別メニューを簡単に作成可能とし、ユーザーの懸念を払拭した。

効果

ここで、個人別メニュー導入による効果を挙げてみる。ユーザー別、個人別メニュー登録状況としてまとめてみた。【図7】

ここでは、「個人別メニュー」の 1 画面内に登録した各個別アプリと、それに対応する旧運用による規定メニューの関係をまとめてみた。

例えば、「ユーザー 30」や「ユーザー 32」のように、その個別アプリが登録されている規定メニュー数をカウントした場合、10 以上のものがある。

この数字の意味は、旧運用で規定メニューから全該当アプリを実行した場合の「メニュー画面遷移数」となる。つまり、今回の「個人別メニュー」により、「ユーザー 30」の例で言えば、各メニュー間で経由する画面も含めれば 12 回以上の「メニュー画面遷移数」が 1 回で済むこととなる（個人別メニュー 1 画面に集約した規定メニュー数は、全運用ユーザー平均で、4.8 画面）。

今後の課題として、運用側マスター登録の煩雑さが挙げられるが、基本的にマスターを 1 度登録すればその後の更新はあまり必要ないため、このまま運用することとした。

最後に

現在、多くの企業様は、基幹システムをさまざまな状況下で構築 / 運用している。レガシー環境から脱却し、GUI システムに移行している企業様もおられる

図3　既存メニューとメニュー登録画面の比較

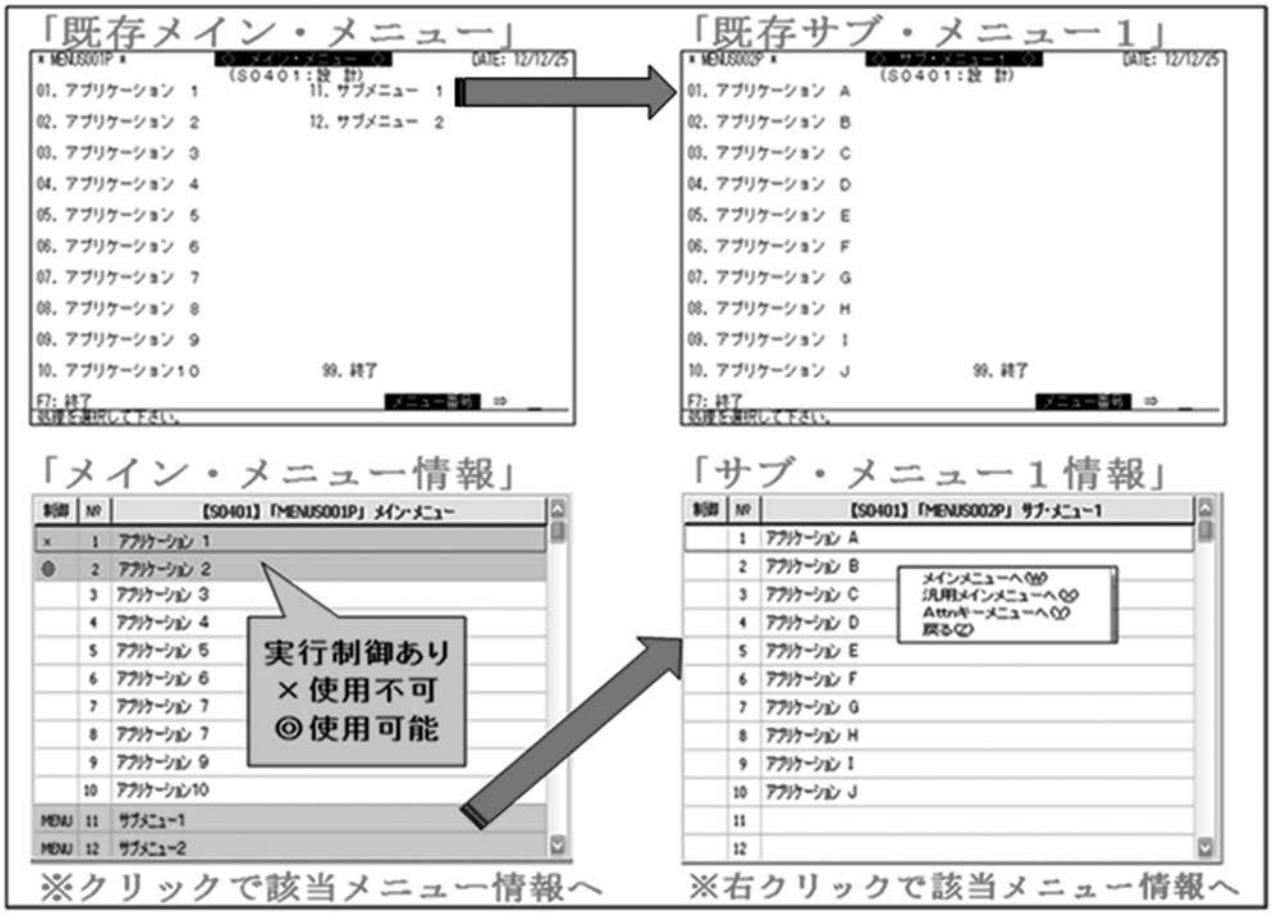
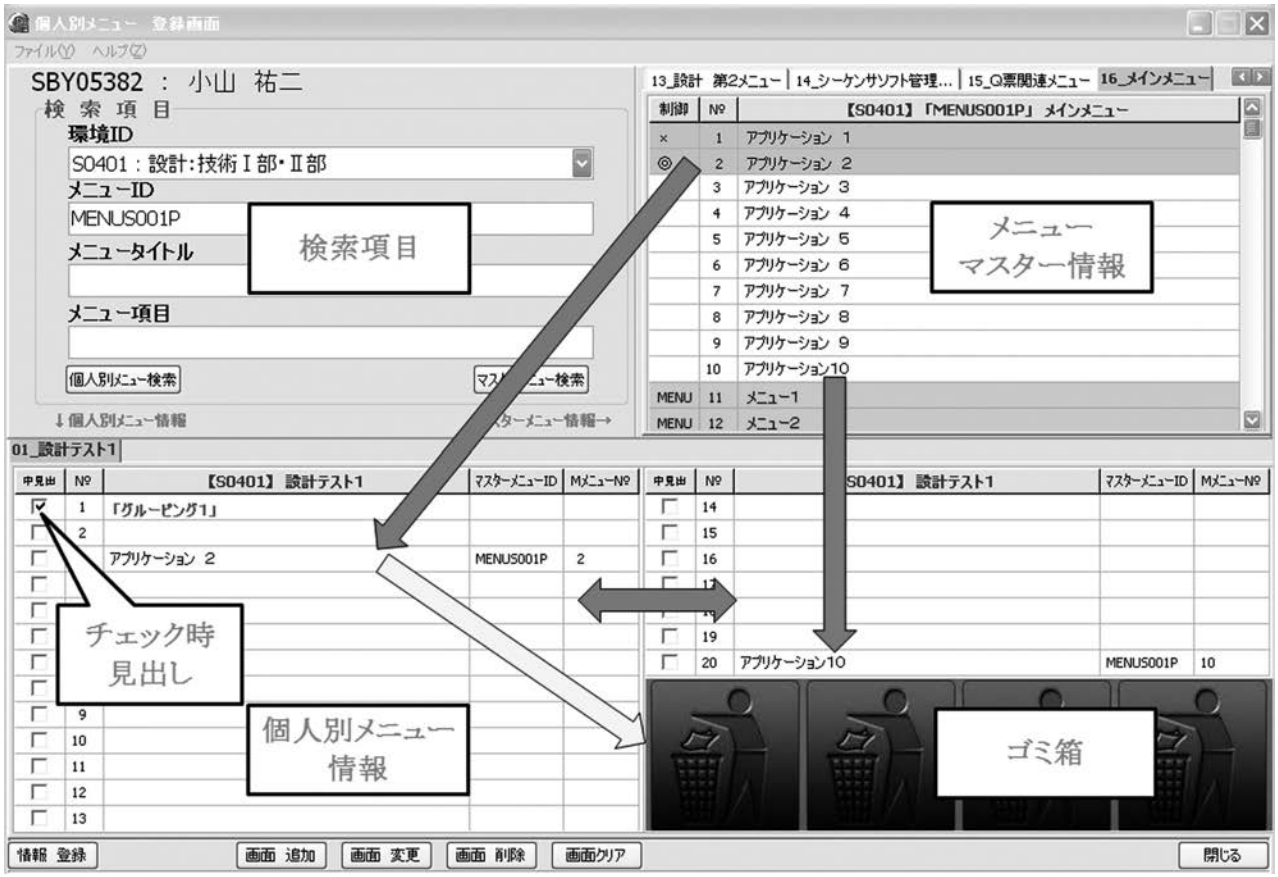


図4　個人別メニュー登録アプリ画面



だろう。

しかし旧資産との関係で、レガシー環境を捨てられない企業様も多いのではないかと思う。そのため現在も、レガシー資産を膨らませているのではなかろうか。

しかし今回説明した通り、5250 画面で実現不可能でも、(5250 画面と連動した) GUI で実現可能である。

私は 5250 画面のパフォーマンスのよさを活かし、CUI と GUI を融合した IBM i の運用の道もあると考えている。そして今後もその可能性を模索し、不必要なレガシー資産削減を実現していきたい。

だが、IBM i の情報は他のサーバーと異なり、非常に入手しにくい。そのため、今後もさまざまな課題が出てくると予想される。

この状況を打開する 1 つの方法として、企業間を超えたナレッジの共有が考えられる。自企業から同地区企業へ、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 つだと考える。ぜひ、本稿をトリガーとし、全国の企業様とナレッジ共有を実現させ、Delphi/400 および Delphi と共に、IBM i を今まで以上に有効かつ効率的に活用していきたい。

M

図5　個人別メニュー画面

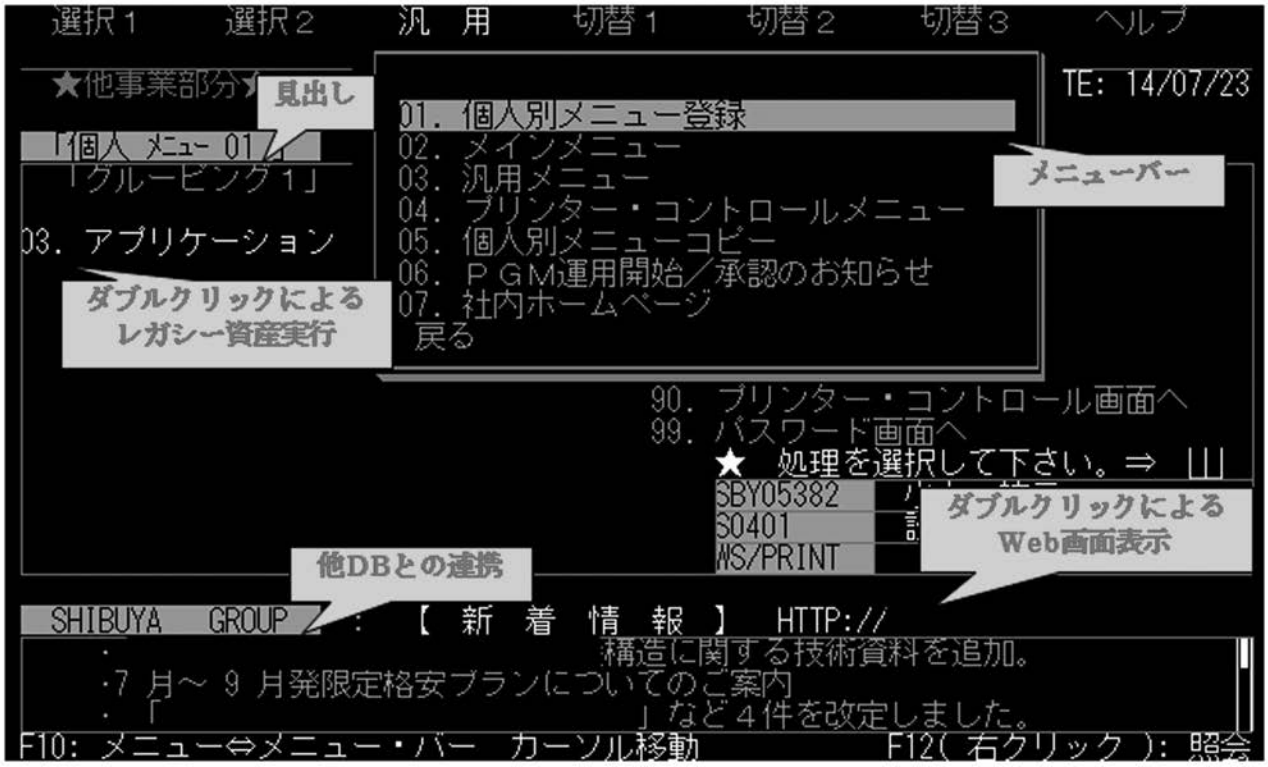


図6　メニュー経路情報･メニューNo.

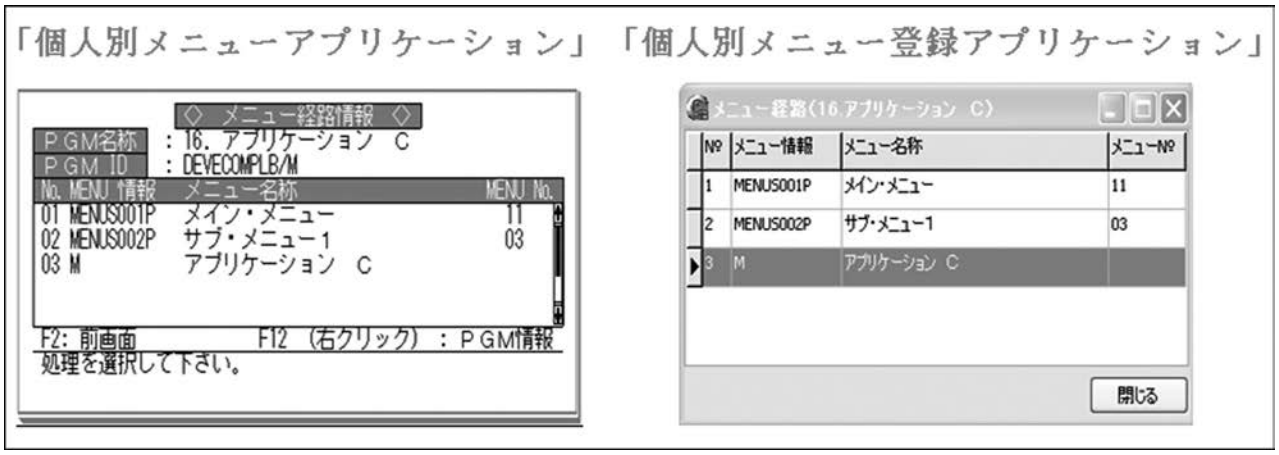


図7　エンドユーザー別　個人別ニュー登録状況

エンドユーザー	各ユーザーの「個人別メニュー」画面番号	「A: 個人別メニューに登録した個別アプリ」の数	Aが存在する規定メニュー数
ユーザー27	1	16	3
	2	16	6
ユーザー28	1	2	1
ユーザー29	1	14	8
	2	14	8
ユーザー30	1	16	4
	2	**14**	**12**
	3	8	4
ユーザー31	1	14	4
	2	9	5
	3	12	6
	4	7	3
	5	2	1
	6	3	2
	7	14	3
ユーザー32	1	**17**	**11**
	2	14	7
ユーザー33	1	14	9
ユーザー34	1	15	4
ユーザー35	1	5	3
ユーザー36	1	13	5

図8 個人別メニュー登録画面（補足）

シルバー賞

IBM iとDelphi/400のコラボレーション
―顧客マスターを再構築する

新谷 直正 様
株式会社アダル
情報システム室

株式会社アダル
http：//www.adal.co.jp/

業務用イス・テーブル・什器などの製造・卸販売、インテリア資材販売および設計・施行を行う。「世の中にないモノは作り出す」をポリシーとし、お客様が求めるものを提供するためには、1 つのスケッチから商品を作り上げる。 特注家具の製造・販売を得意としている。

やっぱり来た!! 営業部門から照会画面作成の要請が…

営業部門から「顧客マスターの照会画面を作ってほしい」との依頼が舞い込んで来たのは､RPG で作成した「在庫照会」や「受注入力」の 5250 画面を Delphi/400 で再構築し、ほっとひと息をついた頃だった。【図 1】

そして、この営業部門からの要求をクリアするには、Delphi/400 を使って単に 5250 画面を GUI 化するだけでは適わない、1 つの大きな課題があった。

顧客マスターファイルのフィールド追加～解決するための策は?

当社はこれまで、AS/400 用の流通基幹パッケージ「D-PACK Ⅱ」のマスター画面を基本的に使用してきたが、時代の流れとともに情報システム室では、顧客情報フィールドの少なさを課題として感じてきていた。その認識は、営業部門も同じだったようで、今回の要望は画面の GUI 化よりも、むしろ顧客（取引先）情報の追加であるとの理解に至った。

従来の取引先入力画面では、会社名や住所、担当部署などの管理情報と、確定済み取引条件（締日／支払日）などの基本情報のみが登録可能であった｡つまり、取引開始の判断の根拠となるような、定性的な評価情報の登録ができなかった。【図 2、図 3】

そこで、仕入先となる取引先については、業務委託先選定の妥当性の判断を効率的かつ合理的に行うためのマスター情報の整備を行うこととし、仕入先に関するより詳細な業務内容、技術スキル、その他の評価項目を顧客マスターに追加するとの方針を決めた。

ここで問題になったのは、顧客マスターのレイアウト変更による影響である。顧客マスター関連のプログラム量は膨大で、論理ファイルの削除・再作成や RPG のリコンパイルの手間などを考えると、短期間にはとうてい構築できそうにない。とはいえ、やはり時間をかけてでもそうすべきなのか、暗中模索の日々が続いた。

「5250画面」と「サブ情報ファイル」の情報をDelphi/400上でドッキング

そして悩んだ末の結論は、「問題解決までの時間短縮」＝「サブ情報ファイルの作成」であった。

営業部門からの要求通りに、新規項目を含むサブ情報ファイルを作成し、既存の顧客マスターファイルと同期をとる。これにより、メイン情報は従来の 5250 画面で入力、サブ情報は Delphi/400 画面で入力し、メインとサブ、2 つの情報を Delphi 上で一覧表示させて、新たな顧客マスター照会画面を構築することに成功した。【図 4、図 5】

なお、各コード項目に対応する値を取得する目的で、「都道府県マスター」「部課担当者マスター」「仕入分類マスター」

図1　在庫照会再構築


在庫照会
12/07/31
ｶﾀﾛｸﾞｺｰﾄﾞ A05　ﾍﾟｰｼﾞ 80
品番　品名　摘要
入荷予定日　入荷数量
現在在庫　受注　売止　出荷可能
C0605-BH　LEM ﾊｲ 支柱 ・ﾍﾞｰｽ ﾊｲｽﾜｰﾙ 用　27　13　0　14
C0605-BL　LEM ﾛｰ 支柱 ・ﾍﾞｰｽ ﾛｰｽﾜｰﾙ 用　207　33　0　174
C0605-BX　LEM 支柱・ ﾍﾞｰｽ のみ ｵｰﾄﾘﾀｰﾝ 用　28　0　0　28
C0605-D　LEM DW 板座のみ ﾀﾞｰｸｳｫｰﾙﾅｯﾄ　2-　0　0　2-
C0605-DP　LEMﾌﾚｰﾑ DW 板座付 ﾀﾞｰｸｳｫｰﾙﾅｯﾄ　4　3　0　1
C0605-F　LEMﾌﾚｰﾑ のみ　228　45　0　183
C0605-N　LEM NA 板座のみ ﾅﾁｭﾗﾙ　0　0　0　0
C0605-NP　LEMﾌﾚｰﾑ NA 板座付 ﾅﾁｭﾗﾙ　17　0　0　17
C0605-P　LEM 革 / 布張用 ﾍﾞﾆﾔ　8-　44　0　52-
F3: 終了　F12: 前画面　PAGE-DN: 次頁　PAGE-UP: 前頁
英数 半角
05/002


在庫照会 [ADQ071]
品目ｺｰﾄﾞ：
品目名　：ﾚﾑ
在庫数：
販売単価：
売り止めのみ
検索
品目ｺｰﾄﾞ | 品目名 | 品目名（ｶﾅ） | 販売単価 | 検索回数
C0605-9DP | レム ハイスワール ダークウォールナット | レム ハイスワール ダークウォールナット | 75,000 | 2,327
C0605-9DX | レム ハイスワール ダークウォールナット 固定 | レム ハイスワール ダークウォールナット | 94,000 | 195
C0605-9NP | レム ハイスワール ナチュラル | レム ハイスワール ナチュラル | 75,000 | 811
C0605-9NX | レム ハイスワール ナチュラル 床固定 | レム ハイスワール ナチュラル ユカコテイ | 94,000 | 110
C0605-9XP | レム ハイスワール 布張込 | レム ハイスワール ヌノハリコミ | 83,000 | 356
C0605-9XX | レム ハイスワール 布張込　床固定 | レム ハイスワール ヌノハリコミ ユカコテイ | 104,000 | 117
C0605-9KP | レム ハイスワール ブラック革 | レム ハイスワール ブラック カワ | 93,000 | 513
C0605-9KX | レム ハイスワール ブラック革　床固定 | レム ハイスワール ブラック カワ ユカ | 114,000 | 100
C0605-9WP | レム ハイスワール ホワイト革 | レム ハイスワール ホワイト カワ | 93,000 | 431
受注内訳
部品在庫照会
部品ｺｰﾄﾞ | 部品名 | 摘要 | 必要数 | 現在在庫 | 受注 | 売止 | 出荷可能 | 入荷予定日 | 入荷数量 | 次回入荷予定情報
111C058SU6 | フレーム・レム・ダークウォールナット用 | | 1 | 5 | 6 | 2 | -3 | 99/99/99 | 210 |
111C058NS3 | 支柱・レム・ハイスワール用 | | 1 | 140 | 29 | 4 | 107 | 99/99/99 | 150 |
111C058OS4 | ベース・レム用 | | 1 | 436 | 41 | 7 | 388 | 99/99/99 | 511 |

などのファイルも結合し、画面に表示する顧客マスター情報を構築している。【図6 関連ファイルからのデータ取得】（iSeries ナビゲーターの Visual Explain 画面）

評価

開発に関しては、既存プログラムに一切修正を行うことなく、ユーザーの求める顧客マスター情報を追加し、短期間にシステム対応するという目標を達成できた。

また、今回の追加で登録可能とした情報は、オンライン画面だけでなく、「評価表」という一覧帳票でも結果を確認でき、取引先選定の効率化に役立っている。

今回のDelphi/400開発を通して学んだこと

社内の各部門からの要求に応える際に最も重要なことは、レスポンスのよさではないかと考える。それを実現するためのツールとして、Delphi/400 は今回も大きな助けとなってくれた。

今後も既存の RPG 資産を活かしつつ、DB2/SQL をメイン言語として、Delphi/400 で 100% GUI 化を目指して開発に取り組んでいきたい。

M

図2　顧客マスター登録(5250画面)-1

図3　顧客マスター登録(5250画面)-2

図4　顧客マスター登録(Delphi400)-1

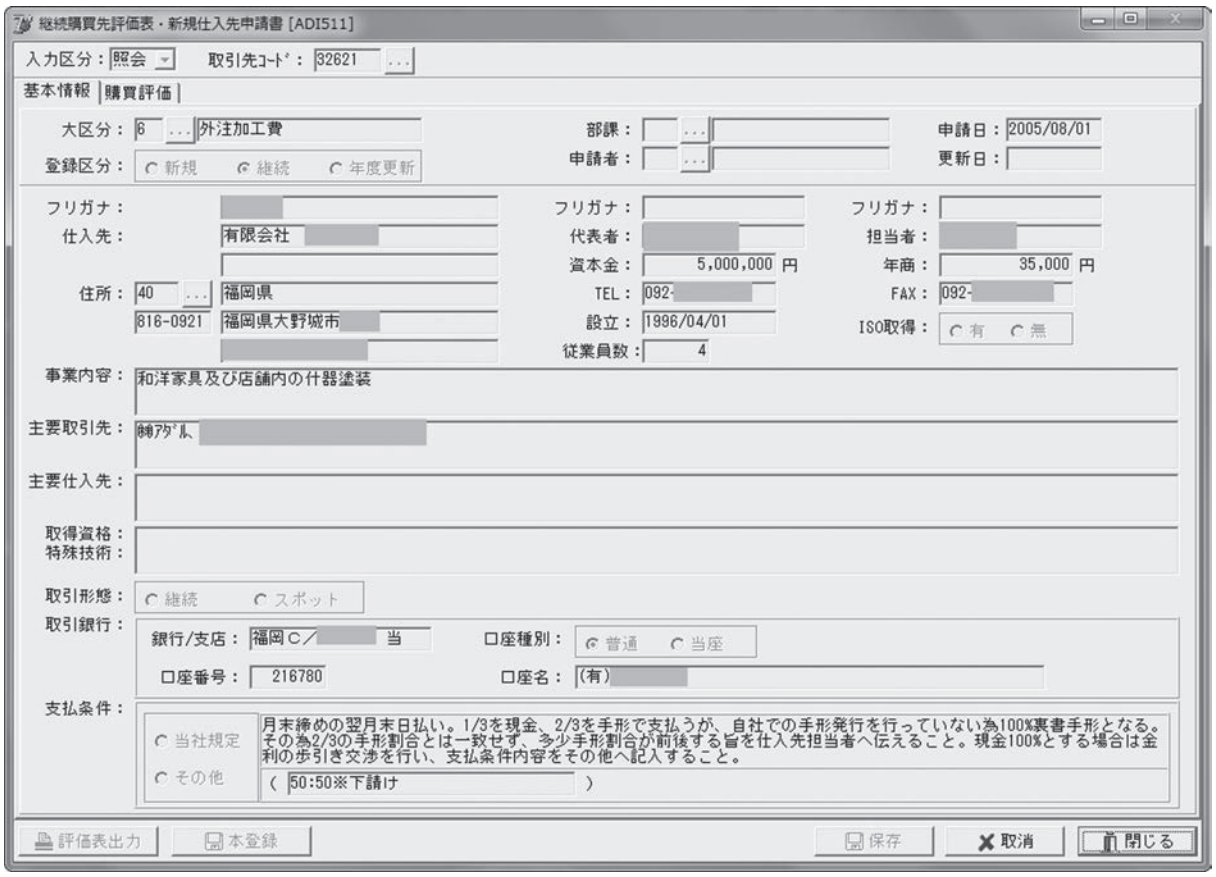


図5　顧客マスター登録(Delphi400)-2

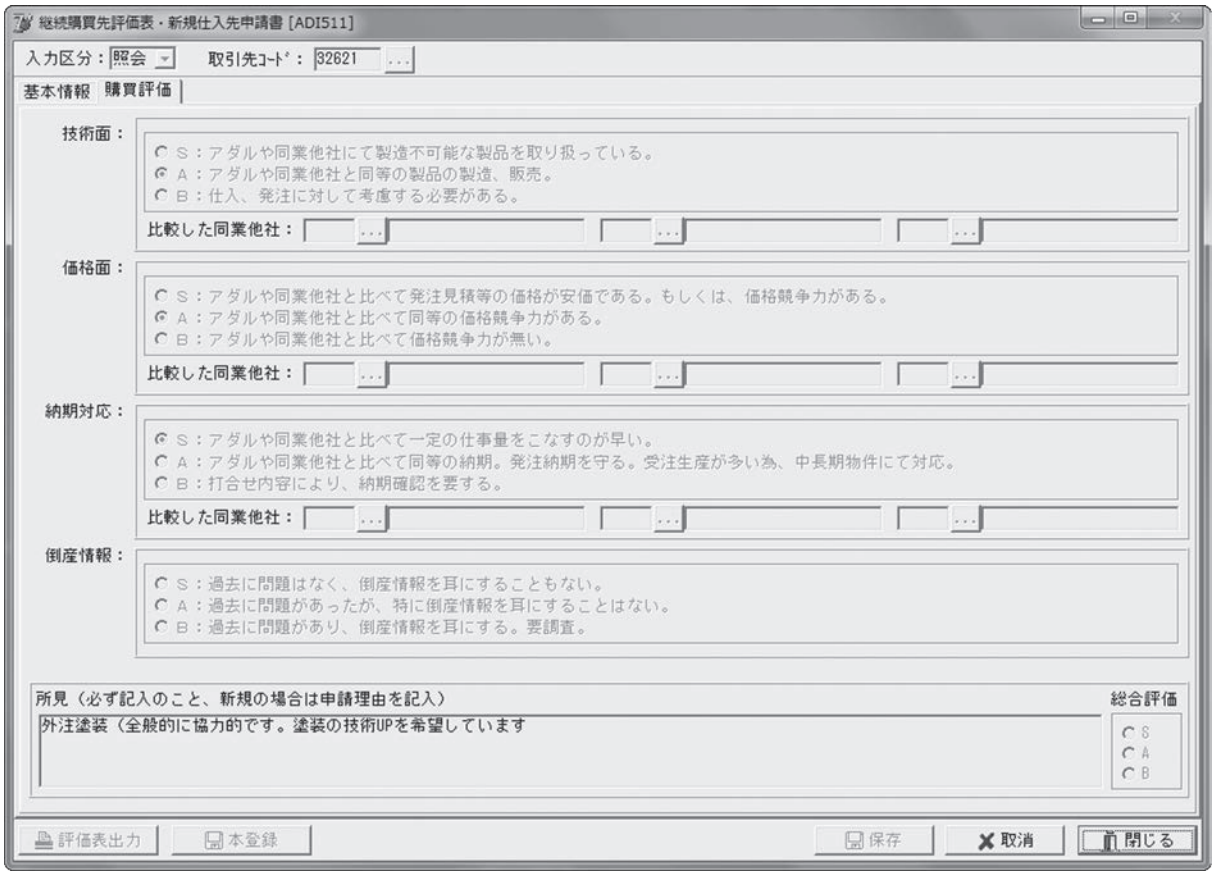
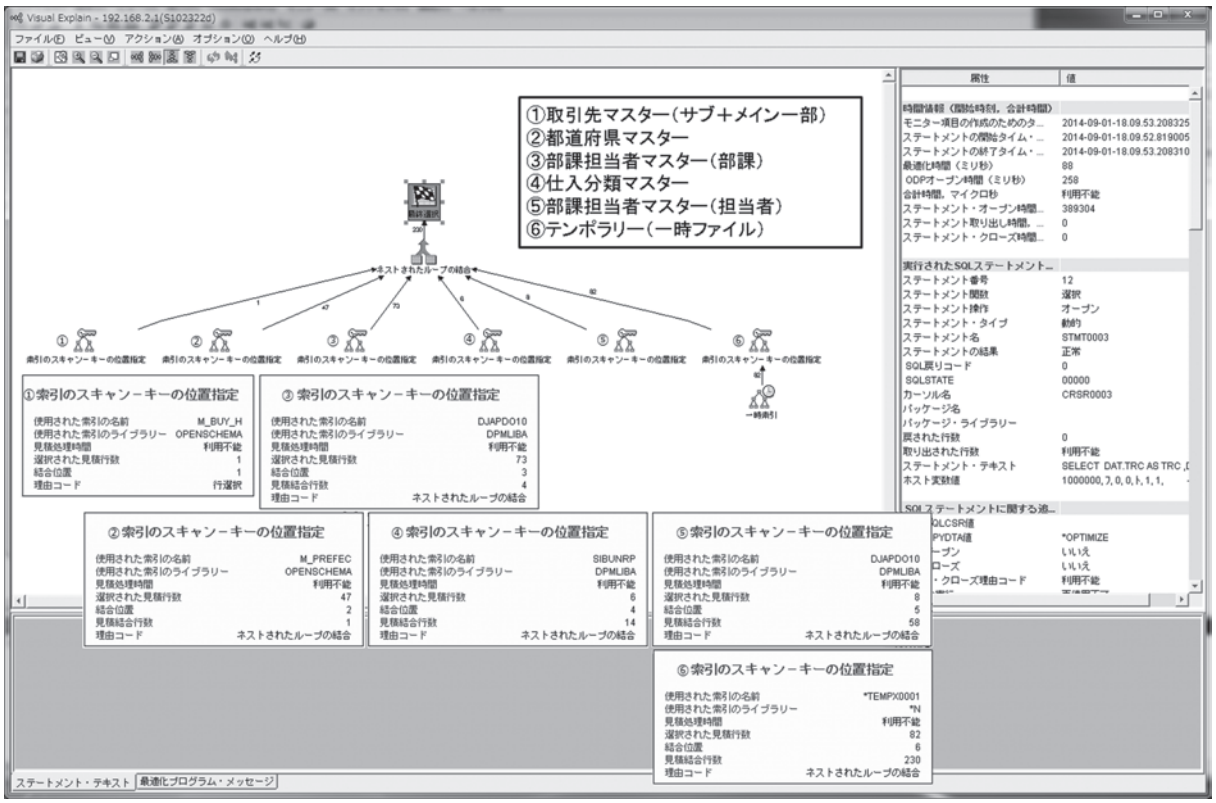


図6　関連ファイルからのデータ取得



シルバー賞

荷札発行システムリプレースについて

仲井 学 様
西川リビング株式会社
経営システム室
課長代理

西川リビング株式会社
http：//www.nishikawa-living.co.jp/

眠りから健康を創造し、より快適な暮らしを提案する西川リビング株式会社。時代のニーズに合わせた健康機能商品や新商品の開発を行っている。創業 1566 年の寝具・寝装品の製造卸業。

はじめに

西川リビング株式会社は、寝具・寝装品の取り扱いを主力とし、インテリア用品や生活雑貨など、暮らしに関わる幅広い商品の提供を通して、快適な暮らしをサポートしている。

本稿では、先般実施した「荷札発行機能」のシステムリプレースについて紹介する。

その中でも特に「量販得意先様向けの荷札発行」を取り上げる。【図 1、図 2】

荷札とは

まず、「荷札」の仕組みについて説明する。「荷札」とは出荷時に倉庫内作業で必要となるシールで、具体的には、「荷札」と「内容明細」の 2 枚を 1 対として構成されている。このシールが商品を梱包した外装ケースに貼り付けられ、搬送ラインを流れていく。

量販出荷の場合は、納品時に「SCM ラベル」も必要となる。これは、先方に納品する際に必要となるシールである。得意先によって レイアウトはさまざまだが、多くの場合、店名、店コード、荷番、そして、バーコードが印字されている。

出荷作業をスムーズに行うため、この 3 種類のシールを 1 セットとして発行する仕組みとしている。

荷札発行機能リプレースの理由と次期システムへの要求スペック

リプレース前は、AS/400 で稼働する AFP ユーティリティというソフトウェアから INFOPRINT250 というレーザープリンターで荷札を発行していたが、メーカーサポートの期限切れが迫っており、プリンター選定を含め半年での対応が必要となっていた。

次期システムのスペックとして一番に求められたのは、「印刷速度」であった。出荷作業にダイレクトに影響するので現行を下回る速度になるのは避けてほしいと、現場からの強い要望があった。

また、「継続性」（高い稼働率）が重要である。「荷札」が発行できないと出荷はできなくなる。これは当然避けなければならない。

「コスト」も重要な条件である。現行機種シリーズは大型レーザープリンターで非常に高額であるため、これが現行の仕組みの継続導入を躊躇する最大の理由となった。

また、当初問題にはなっていなかったが、今回のリプレースを通してコストを見直したところ、トナー代が予想以上にかかっていたことが判明した。

最後に「導入期間」が期限内であるかどうか。メーカーサポート期間中にリプレースが完了することが必須であった。短期間ならサポート切れのまま運用することも考えたが、週 1 ～ 2 回は、紙詰まり、印字のかすれなどでメンテナンスに来ていただいていたことがわかり、「サポート終了＝使用不可能」という逃げ場のない状況であった。

図1　倉庫社名板

図2　倉庫外観

図3　まるち監視くん

機器の選定

次期プリンターは、InfoPrint250 の後継機種か、株式会社サトー（以下、サトー社）のプリンターかという選択肢になった。

InfoPrint 系であれば、ソフトウェア資産はそのまま使用できるが、前述した通りプリンターのコスト面がネックとなっていた。また、現行機がメンテナンスに依存していたことも気になっていた。2 台設置していたが、同時に 2 台とも不調、もしくは故障してしまうことがないとは言い切れない。

一方、サトー社の機器での問題点は、「印刷速度」と「導入期間」であった。単体の印刷速度では現行機種にまったくかなわない。しかし、「印刷速度」については、サトー社「まるち監視くん」を使用し、複数台で同時発行することで対応可能である見通しが立った。

印刷内容のテキストデータにプリンター番号を頭に付けて「まるち監視くん」に渡すと、データごとに指定したプリンターから自動的に発行される。複数台での発行で代わりが効くようになり「継続性」も確保される。

残る課題は「導入期間」だった。これについては、印刷内容データを作成する現行の RPG を再利用できれば「導入期間」もクリアできると考えられた。【図 3】

システム設計

RPG 資産の再利用となれば「Delphi/400」の出番である。

SCM ラベルが量販の得意先様別に違うので、その種類に応じてプログラムが約 50 本ほど存在する。これらの RPG プログラムをほぼそのまま流用した。従来の「AFP ユーティリティ」に代わり、Delphi/400 アプリで荷札関連の帳票プログラムを実行し、結果を「まるち監視くん」経由で新しいプリンターに連携して荷札発行を行う仕組みである。この開発方法は、既存帳票プログラムの品質も受け継がれるため非常に有効であった。

プリンターは検討の結果、5 台導入することとした。複数台の制御で煩雑になってはユーザーは使ってくれないので、できるだけ使いやすい画面を目指して設計した。

画面上で、発行する得意先の設定と使用するプリンターをチェックすれば、分散印刷される枚数が表示される。また、使用するプリンターを減らしたい時や増やしたい時などにプリンター指定のチェックを変えると枚数の表示が更新される。プリンター指定のデフォルト設定も得意先別に可能とした。【図 4、図 5、図 6、図 7、図 8】

導入後の評価

印刷速度の改善

従来の InfoPrint250 では、発行指示をかけてから実際にプリンターが動き出すまで 2、3 分かかっていたが、新システムで「まるち監視くん」にデータを渡すと、ほぼタイムラグなく印刷が開始される。印刷にかかる時間はトータルでは短縮されるようになった。

コストの改善

導入コストは、機器の費用がレーザープリンターに比べ非常に安価になり、機器保守料金もそれに応じて安価になった。用紙関連も感熱紙になったことでトナー代が不要となり、用紙代自体も従来より安価になった。

開発効率のアップ

AFP ユーティリティでの新しいラベルの作成は GUI 画面もなく非常に手間がかかったが、新プリンターの場合は、サトー社の環境を使用することで作成効率が大幅にアップした。

出荷作業効率のアップ

発行順序の変更機能など、Delphi/400 による開発で小回りがきき現場の要望にきめ細かく対応することができた。それが現場での作業効率のアップにもつながった。

以上、本稿では Delphi/400 を活用したシステムリプレースをご紹介した。世の中に存在する安価で便利なツールを組み合わせて非常に効率的な仕組みにすることができたと思っている。

Delphi/400 で開発したプログラムの使用は社内がメインとなっているので、さらに範囲を広げて社外でも活用していくため、今後は Web システムにも利用していきたい。

M

図4　システム構成(新旧比較)

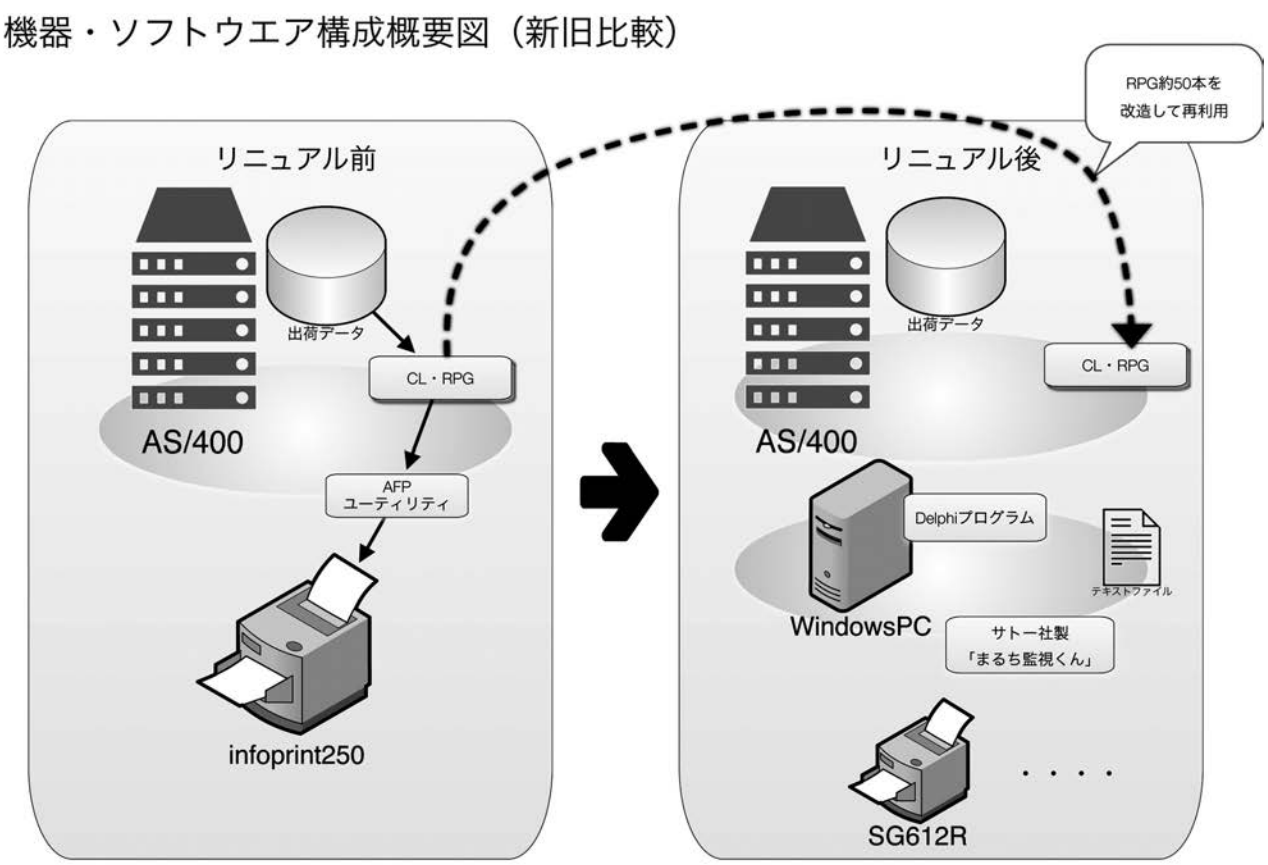


図5　プログラム機能概要図

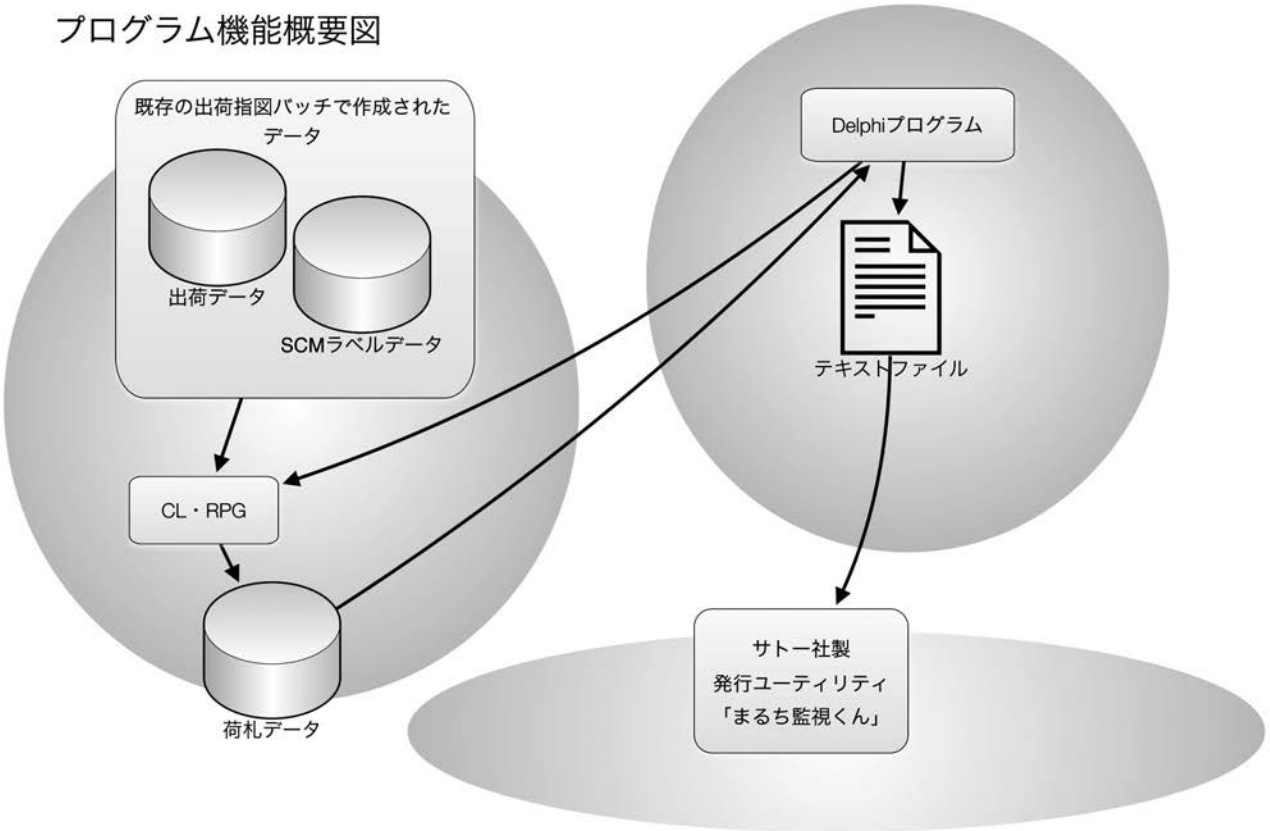


図6　ラベル発行指示画面

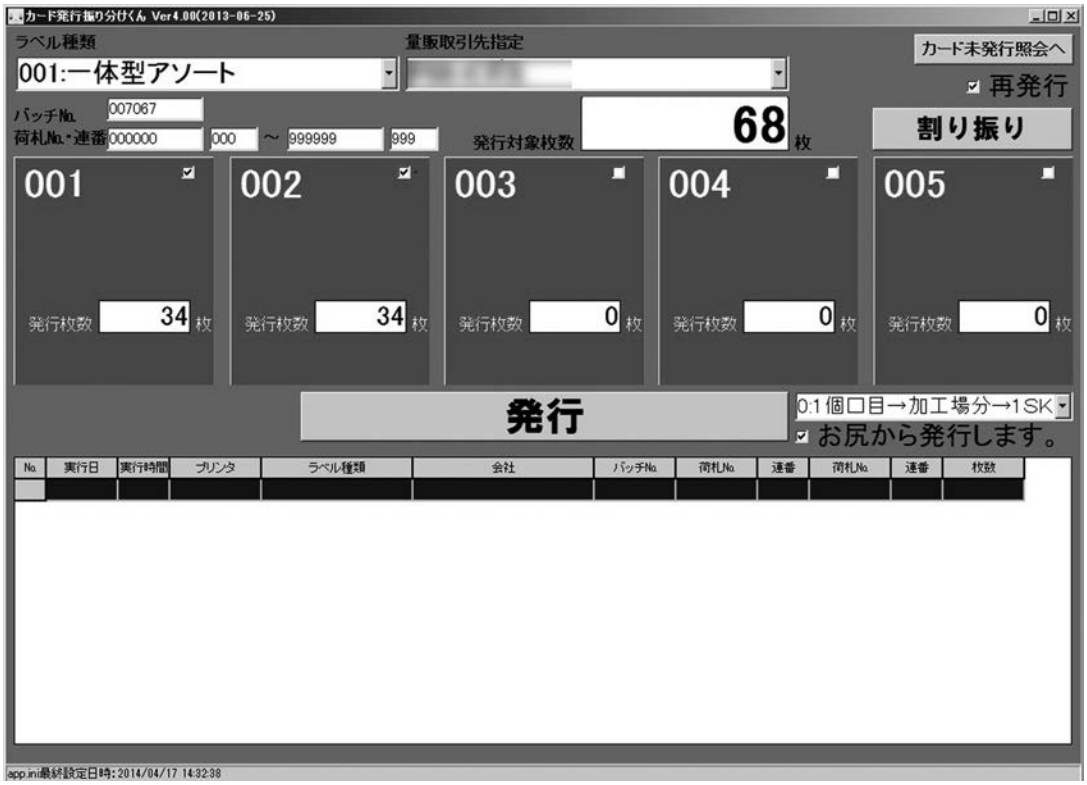


図7　新規プリンター5台（SG612R）

図8　制御用PC

優秀賞

Delphi/400バージョンアップのためのクライアント環境構築

普入 弘 様
株式会社エイエステクノロジー
シニアマネージャー

株式会社エイエステクノロジー
http：//www.as-t.co.jp/

AS/400 向けソフトウェアの開発、運用をメインとする技術者集団として 1988 年に設立。現在、RPG、Delphi/400 による IBM i 向けシステム構築や、Web 開発にも取り組む。豊富なノウハウと優れた技術力をもとに、企業が抱える課題を解決する高品質のソリューションを提供し続けている。

1.業務課題

クライアント / サーバー型の Delphi/400 の端末環境をバージョンアップする際には、クライアントモジュールの旧バージョン・アンインストール、新バージョン・インストール、Delphi/400 Configration 設定情報の更新が必要となる。この作業をユーザーでも簡単に実行できるよう簡易化する必要があった。バージョンアップは、以下のバージョンを対象として行った。

・旧バージョン　Delphi/400 Version 7　利用ドライバ：BDE
・新バージョン　Delphi/400 Version XE3　利用ドライバ：DB Express

2.技術課題

バージョンアップに必要な実施対象項目は、以下の通りである。

旧バージョンのアンインストール対象
①ユーザーアプリケーション
② BDE
③ Delphi/400 運用ライセンスクライアントモジュール

新バージョンインストール対象
①ユーザーアプリケーション
② Delphi/400 運用ライセンスクライアントモジュール

これらを実現するにあたり、次の（1）～（4）を検討した。

3.技術課題の解決策

(1)ユーザーで容易に実行できるよう「インストーラメニュー」の作成

インストーラメニュー作成時の考慮点は、以下の通りである。【ソース 1】
・管理者権限要求アプリケーションとして作成（「第 11 回 ミガロ .テクニカルセミナー」参照）
・インストーラの組み換えを自由にできるよう ini ファイルを利用

(2)「InstallShield Express」が付属しなくなったため、代替のインストーラ作成方法の検討

インストーラ作成にフリーのツール「Windows Installer XML（WiX）」を利用。設定情報を直接テキストエディターでの記述するのは大変なため、GUI ツール「WiX Edit」にて作成。

(3) Delphi/400 運用ライセンスクライアントモジュール「サイレントインストール」の作成

サイレントインストールにより、Configuration の設定も不要で実行だけでインストールが完了する。課題は、「サイレントインストール」の仕様として、iss ファイルを絶対パスで指定する必要があり、インストーラを CD で提供する場合に不便なため、bat ファイルでカレントを指定（「サイレントインストール」の作成については「第 5 回ミガロ .テクニカルセミナー」参照）。【ソース 2】

ソース1　汎用的なインストーラーメニューの作成

＜考慮点＞
・管理者権限要求アプリケーションとして作成（第11回ミガロテクニカルセミナー参照）
・インストーラの組み換えを自由にできるようiniファイルを利用

【delphiソースの一部】

```
procedure TForm1.FormCreate(Sender: TObject);
var
  IniFile: TInifile;
begin
  CURRENT_PATH:=ExtractFilePath(Application.Exename);
  //iniファイル読み取り
  Inifile := TInifile.Create(CURRENT_PATH+'SETUP.ini');
  try
    LblPROGRAM1.Caption:= Inifile.ReadString('PROGRAM1','Name','');
    PROGRAM1:=            Inifile.ReadString('PROGRAM1','Value','');
    LblPROGRAM2.Caption:= Inifile.ReadString('PROGRAM2','Name','');
    PROGRAM2:=            Inifile.ReadString('PROGRAM2','Value','');
  finally
    IniFile.Free;
  end;
end;

procedure TForm1.LblPROGRAM1Click(Sender: TObject);
begin
  ShellExecute(Handle,nil,PChar(CURRENT_PATH + PROGRAM1),nil,nil,SW_NORMAL);
end;

procedure TForm1.LblPROGRAM2Click(Sender: TObject);
begin
  ShellExecute(Handle,nil,PChar(CURRENT_PATH + PROGRAM2),nil,nil,SW_NORMAL);
end;
```

【iniファイル例】

```
[PROGRAM1]
Name=■１．ユーザーアプリケーションのセットアップ
Value=install.msi

[PROGRAM2]
Name=■２．Delphi400インストール
Value=D400install.bat
```

ソース2　Delphi400のサイレントインストール

＜考慮点＞
・issファイルを絶対パスでの指定が必要であり、
インストーラをCDで提供する場合不便なため、batファイルでカレントを指定
（サイレントインストールについては第5回ミガロテクニカルセミナー参照）

【batファイル例】

```
rem %~dp0 は、実行されているファイルが置かれているカレントディレクトリを表す
start /WAIT /MIN XE3_Silent.exe SETUPFILE="%~dp0setuppcF.iss"
```

ソース3　アンインストールの単純化

＜考慮点＞
・レジストリから
「HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVwrsion¥Uninstall」
からアンインストールするアプリケーションの「UninstallString」の値を取得し記述

【batファイル例】

```
rem ユーザーアプリ
start /WAIT MsiExec.exe /X{3F10A288-0623-42DF-B6DA-77A2FE0C80BB}
echo y | rmdir /S c:¥hogehoge

rem BDE
start /WAIT MsiExec.exe /X{958F3040-0A1B-4D98-8081-1C6EA753EB51}

rem D400
start /WAIT MsiExec.exe /X{FFCAC3AE-0430-4028-A104-F2B026D4C50A}
echo y | rmdir /S c:¥co407
```

(4) アンインストールの単純化

1つのバッチファイルで、前述の対象①～③のアンインストールを実行可能とする。考慮点は、レジストリからアンインストールするアプリケーションの「UninstallString」の値を取得し記述。【ソース3】

4.業務課題解決と効果

インストーラメニューの提供によりクライアント環境のバージョンアップの手順が容易にわかり、各機能も実行するだけで設定値の入力が不要なため、ユーザーでクライアント環境の入れ替えが可能となった。

これにより、遠隔地拠点も含めた大量の端末のバージョンアップ作業の工数削減を実現できた。

M

優秀賞

外出先からメールでリアルタイム在庫を問い合せ

島根 英行 様
シルフ


シルフ

業務内容は IBM i を中心としたシステム開発。販売管理をメインに、オフコンや PC サーバーからのリプレース提案を得意としている。

1.業務課題

営業担当者が外出先から商品在庫を確認したい、という要望はよくあるが、Web サーバーなどの追加投資なしで、営業担当者が既に持ち歩いている携帯電話だけで実現できないか、という課題があった。そこで、IBM i 基幹システムに保有するリアルタイムの商品在庫を、携帯メールで手軽に照会できる仕組みを検討することとした。

2.技術課題

Web システムを構築することなく、メールだけで在庫照会を実現することが技術的な課題であった。その解決のための基本的な仕組みは、以下の通りである。

[1]ユーザーは、照会したい商品の商品コードを記載したメールを所定のアドレス宛に送信

[2]Delphi/400 プログラムは以下の処理を行う

(1) メールの待ち受け:一定時間 (5 分) 間隔でメールサーバーをチェック
(2) 受信メールの解析：在庫照会メールが存在した場合、メール本文から商品コードを抽出
(3) 在庫データの取得：IBM i にログインし、指定された商品コードの在庫 DB 照会などを行う
(4) 結果の自動返信：結果をメールに記載（または添付）して差出人のユーザー宛に返信。また、なんらかのエラー発生時は、システム管理者にメールで通知

[3]ユーザーは返信メールにより、在庫などを確認

本稿では、特に（a）メール送受信機能の実現、(b) 本来不定型のメールをシステムへの入力データとして扱う方法を技術課題として紹介する。

3.技術課題の解決策

(a) メール送受信機能について

(1) メール受信：コンポーネント（Indy コンポーネント POP3）を使用して実現。【ソース 1】
考慮点：照会メールの差出人が正当なユーザーであるかチェックを行う
(2) メール送信：コンポーネント（Indy コンポーネント SMTP）を使用して実現。【ソース 2】

(b) 本来不定型のメールをシステムへの入力データとして扱う方法

在庫問い合わせメールの本文はシンプルに「No.（通番）＋商品コード」のみで送信することとした。

入力テキスト例　0：4520179484556
（コロンの前半が No.、後半が商品コード）

No. と商品コードの間は「:」（コロン）

【ソース1】

ソース１　　受信メールの確認

```
procedure TfrmSendMail.chkmaile;
var
  //中略
begin

  try
  //中略

    //各パラメータを設定し、メールサーバーからメールを取得
    IdPOP31.Host          := Edit1.Text;
    IdPOP31.Port          := 110;
    IdPOP31.Username    := Edit2.Text;
    IdPOP31.Password    := Edit3.Text;
    IdPOP31.Connect;
    TStest:= TStringList.Create;
    Msg :=    TIdMessage.Create(Self);
    if IdPOP31.CheckMessages>0 then begin
      i := 1;
      IdPOP31.Retrieve(i, Msg);          //受信メールを取得
      Edit4.Text := Msg.Subject;         //件名取得
      Edit5.Text := Msg.From.Address; //差出人
      //差出人の正当性を確認する（多くの人を管理する時は、データベース化する）
      if Edit5.Text = 'XXXXX@AAAA.ne.jp' then chkA := True;
      TStest.AddStrings(Msg.Body);       //メール本文
      Memo1.Lines.Text:= (jconvert.ConvertJCode(TStest.Text, SJIS_OUT));
      IdPOP31.Delete(i);     //サーバーの受信メールを削除
    end;
  finally
    IdPOP31.Disconnect;
    Msg.Free;
    TStest.Free;

    //以下省略       IBM i へのログイン、在庫等データの取得へ続く
```

で区切るルールとし、プログラムで読み取っている。【ソース 3】

4.業務課題解決と効果

実際には、在庫数に加えて、出荷データ、入庫データ、棚卸しデータなどの問い合わせが可能な仕組みとし、問い合わせ件数が多い場合は、結果を csv ファイルにして添付ファイルで返信するなどの工夫を行っている。これにより、当初の目的通り、外出先からリアルタイム在庫残高などを照会する仕組みを、追加投資なしで実現することができた。

M

【ソース2】

ソース 2　　ユーザーへのメールの送信

```
function TfrmSendMail.chkZaiko(chkZA002 :String): Boolean;
var
        //中略
begin
  Result := False;

  try
    //SQL で在庫ファイル取得　（中略）
      chkkazu := DM.Query1.FieldByName('ZA014').AsInteger;      //在庫数セット
      chkMei := DM.Query1.FieldByName('IM011').AsString;        //商品名セット
    end;
        //中略

  try
    SMTP := TIdSMTP.Create(nil);          // 以下、送信メールを定義
    msg := TIdMessage.Create(SMTP);
    msg.From.Name := edtFromName.Text;
    msg.ContentType := 'text/plain';
    msg.CharSet := 'ISO-2022-JP';
    msg.ContentTransferEncoding := '7bit';
    msg.Recipients.EMailAddresses := edtToMail.Text;  //宛先
    msg.From.Text :=  'aaaa@ccc.co.jp';       //差出人
    edtSubject.Text := 'ZAIKO';      //在庫
    memBody.Lines.Add(chkMei+'の在庫数は、'+IntToStr(chkkazu)+'  です'+#13);

        //中略
    // メッセージを送信
    try
      SMTP.Host       := edtHost.Text;        //SMTP サーバー名
      SMTP.Port       := 0;                   //ポート番号
      SMTP.Username := edtUserName.Text;  //SMTP ユーザー名
      SMTP.Password := '';                    //SMTP パスワード
      SMTP.Connect;
      SMTP.Authenticate;
      SMTP.Send(Msg);
      Memo2.Lines.add(edtToMail.Text + ' 送信しました。');
      //以下略
```

【ソース3】

ソース 3　　メール本文から「商品コード」と「項目 No.」を取得

```
{************************************************************
  関数名　　：ComBoGet_Text
  機能　　　：コンボボックス内の項目テキストを取得
  引数[I/O]：コンボボックスのテキスト
  戻り値　　：コンボボックスのテキスト内の「商品コード」部分
  備考　　　：入力文字列　0:4520179484556（No.＋商品コード）
************************************************************}
Function ComBoGet_Text(Comb_Text:String):String;
Var
  i:Integer;
begin
  i := Pos(':', Comb_Text)+1;    //コロン(:)より後の文字列を抽出
  result:=Copy(Comb_Text,i, Length(Comb_Text));  //長さを多くとる
end;

{************************************************************
  関数名　　：ComBoGet_No
  機能　　　：コンボボックス内の項目№を取得
  引数[I/O]：コンボボックスのテキスト
  戻り値　　：コンボボックスのテキスト内の「項目 No.」部分
  備考　　　：入力文字列　0:4520179484556　（No.＋商品コード）
************************************************************}
Function ComBoGet_No(Comb_Text:String):String;
Var
  i:Integer;
begin
  i := Pos(':', Comb_Text)-1;     //コロン(:)より前の文字列を抽出
  result:=Copy(Comb_Text,1,i)
end;
```

Migaro.
Technical Report
2014

ミガロ.SE 論文／ミガロ. テクニカルレポート

吉原 泰介

株式会社ミガロ.

RAD事業部 技術支援課

[Delphi/400] iOS/Androidネイティブアプリケーション入門 ―マルチデバイス開発手法から社内配布

略歴
1978年3月26日生
2001年　龍谷大学 法学部卒
2005年7月 株式会社ミガロ. 入社
2005年7月 システム事業部配属
2007年4月 RAD事業部配属

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

1.はじめに

この数年でスマートデバイスの導入が個人だけではなく、企業でも急速に進んできている。実際に2014年に総務省が行ったアンケート調査では、国内企業のスマートデバイス導入率は28.4%と推計されており、3割近い企業でスマートデバイスが使用されていることになる（総務省・経済産業省「平成24年経済センサス－活動調査」）。

2011年の同調査での企業導入率が10%未満だったことを考えると、飛躍的に導入数が伸びていることがわかる。

こうしたスマートデバイスを導入した企業の多くが「業務効率化」を目的としている。これはメールやWebの利用だけではなく、社内システムの活用が強く期待されていることである。

Delphi/400のテクニカルサポートにもスマートデバイスのお問い合わせをいただくことが多くなってきており、これからの社内システムにはスマートデバイスアプリケーションが必要とされる機会が増えてくると実感している。

では、実際にスマートデバイスを導入している企業では、どういった機種が使用されているのかというと、その大半がiOS（iPhone、iPad）やAndroidとなっている。それ以外のWindows Phoneなどの機種は、まだ企業導入されていないのが現状である。

つまり、これから企業のスマートデバイスでアプリケーションを活用するには、iOS、Android機種へ、いかに対応できるかが重要となってくる。

こうした大きな環境変化の背景もあり、従来のクライアント/サーバー（以下、C/S）、Web型開発に加えて、iOS、Androidのネイティブアプリケーション開発にも対応したDelphi/400 Version XE5の新機能を紹介したいと考えた。

本稿では、Delphi/400を使ったスマートデバイス向けのネイティブアプリケーション開発について、開発環境や開発手順、配布・運用のポイントを説明していきたい。

2.スマートデバイスアプリケーションの種類

この章では、スマートデバイス上で動作するアプリケーションの種類や特徴について説明する。

スマートデバイスアプリケーションは、大きく2つの種類に分類できる。アプリケーションの種類としては、Webアプリケーションとネイティブアプリケーションがある。Delphi/400ではどちらのアプリケーションも開発できるが、それぞれのアプリケーションの特徴を把握してみる。

Webアプリケーション

Webアプリケーションは、PCでの使用と同様にWebブラウザを使って使用するアプリケーションである。Webアプリケーションは Webサーバー上で動作しており、スマートデバイス端末ではWebブラウザを使って利用することができる。【図1】

図1　Webアプリケーション構成

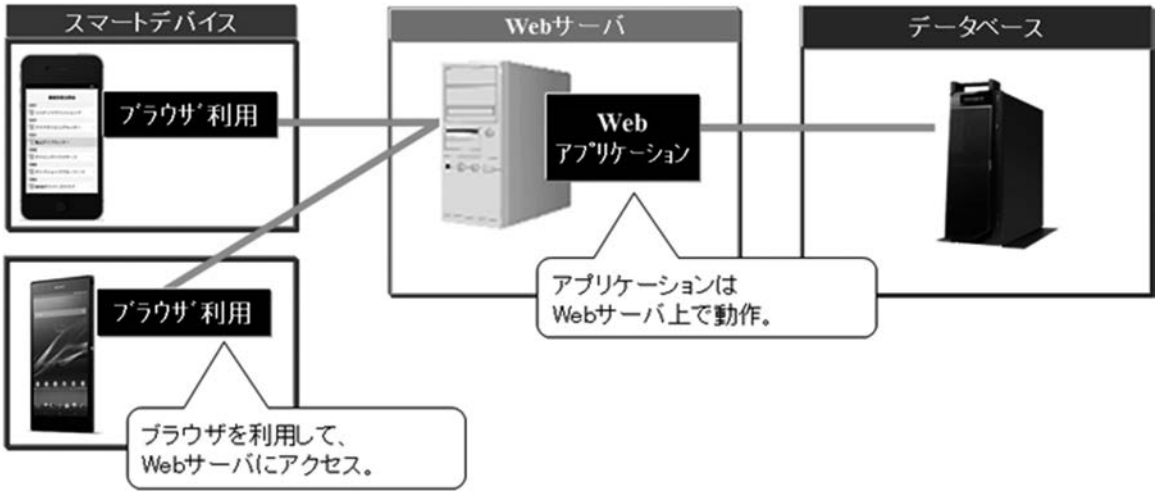


図2　ネイティブアプリケーション構成

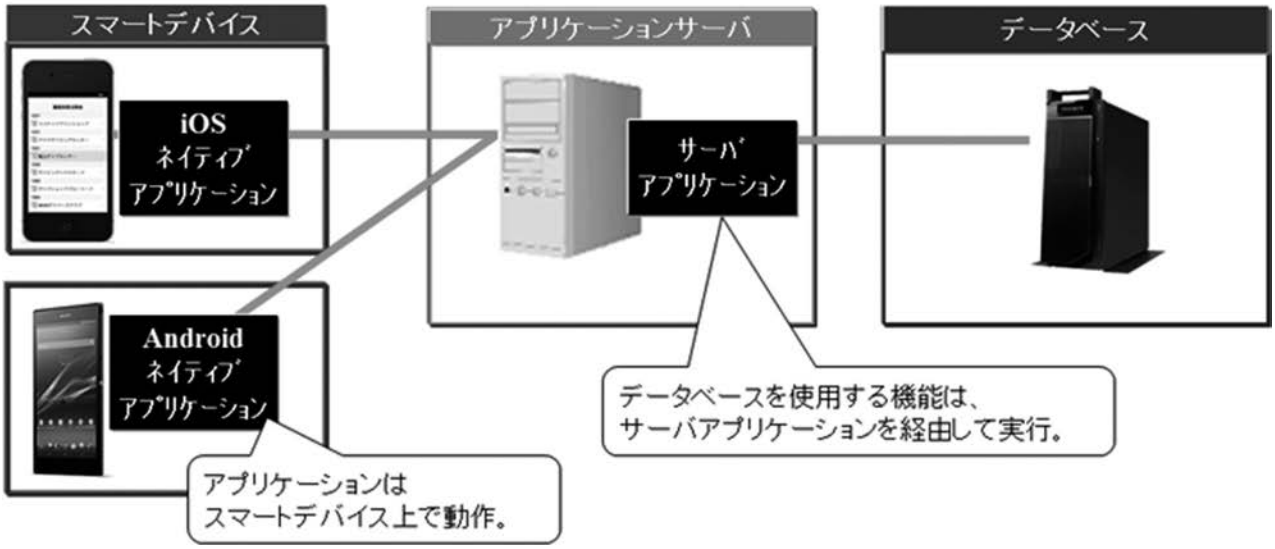


図3　ネイティブ・Webアプリケーションの特徴

	ネイティブ	Web
開発言語	Delphi	Delphi
開発生産性	◎	◎
デバイス機能	◎	△
パフォーマンス	◎	○
オフライン動作	◎	×
配布	△	◎

そのため、スマートデバイス端末にはアプリケーションがインストールされることはない。

この特徴のメリットは、「アプリケーションの配布が不要」という点と、「ブラウザに対応していれば機種の制限がない」という点が挙げられる。逆にデメリットとしては、「スマートデバイスのネイティブ機能が十分に活用できない」「ネットワークに接続していない環境では使用できない」という点がある。

ネイティブアプリケーション

ネイティブアプリケーションは、スマートデバイス端末上にインストールして使用するアプリケーションである。もちろんネイティブアプリケーションは、スマートデバイス端末上で動作する。【図2】

この特徴のメリットとしては「スマートデバイスのネイティブ機能を100%活用できる」「ネットワークに接続していない環境でも使用できる」という点である。またスマートデバイス端末上で直接実行するため、アプリケーションの動作レスポンスは、一般的にWebアプリケーションより優れている。

デメリットとしては、「iOS、Androidごとに別言語の習得・開発が必要となる」という点が挙げられる。しかしDelphi/400では、iOS、Androidのネイティブアプリケーションを Delphi言語のみで開発することができる。そのため、ネイティブアプリケーションの一般的なデメリットも、Delphi/400では逆に長所となっている。【図3】

これは「マルチデバイス開発」と呼ばれる画期的な開発手法によるものである。「マルチデバイス開発」については、後述の開発環境で詳しく説明する。

それぞれアプリケーションの種類によって、得手・不得手の部分があるが、Delphi/400ではどちらのアプリケーションも開発でき、用途によって選択することができる。

例えば、機能面が優先であればネイティブアプリケーション、運用管理の軽減が優先であればWebアプリケーションといった選択も可能である。

3.ネイティブアプリケーションの開発環境

この章では、Delphi/400のネイティブアプリケーション開発環境を詳しく説明していく。

iOS/Androidのネイティブアプリケーションは Delphi/400で開発できるが、対象とするスマートデバイスによって必要となる開発環境が違ってくる。それぞれの開発で必要となる環境や設定ポイントを簡単にまとめてみた。対象とする開発環境部分をご一読いただきたい。

3-1. iOS向け開発環境

Delphi/400でiOS向けのアプリケーションを開発する場合、Windowsだけではコンパイルやアプリケーションの配布が行えないため、OSX（Mac）が必要となる。【図4】

もちろんMac上に仮想環境(Windows)を構築すれば、1台のマシンで開発環境を用意することも可能である。

必要となる環境

- Windows端末（Delphi/400 Version XE5）
- Mac端末（OSX 10.7 ～ 10.9）
- iOS Developer Program（Xcode、配布）
- iOS実機（iPhone、iPadなどiOS 6.0 ～ 7.1）

Mac環境の構築

iOSの開発環境では、Delphi/400をインストールしているWindows端末とは別にMac端末が必要になる。Mac端末はOSX 10.7Lion以降をサポートしている。

Xcodeのインストール

Mac端末にはiOS6.0以降に対応したXcodeのインストールが必要になる。XcodeはApp StoreまたはAppleのDeveloperページからダウンロードしてインストールすることができる。【図5】

iOS Developer Program

iOSアプリケーションはApple社の規約により、iOSへ配布するためには「iOS Developer Program」に加入する必要がある。「iOS Developer Program」は1年間の有償プログラムとなっており、用途別に2種類用意されている。「iOS Developer Program」と「iOS Developer Enterprise Program」である。

① iOS Developer Program
https://developer.apple.com/jp/programs/ios/

「iOS Developer Program」は、主にApp Store向けのアプリケーションを配信するためのプログラムになる。Ad Hocと呼ばれる機能で社内向けにアプリケーションも配布することはできるが、台数は上限が100台に設定されており、端末の事前登録も必要になる。

② iOS Developer Enterprise Program
https://developer.apple.com/jp/programs/ios/enterprise/

「iOS Developer Enterprise Program」は社内専用向けアプリケーションを配布するためのプログラムになる。「iOS Developer Program」と違い、社内向けに配布できる台数に制限がなく、端末の事前登録も必要ない。ただしApp Store向けにアプリケーションを配信することはできない。

この2種類のプログラムは用途別に用意されているが、社内用アプリケーションを開発・運用する場合には、「iOS Developer Enterprise Program」のプログラムが目的に合っている。注意点としては、プログラムの購入から手続きの完了までApple社の処理に数日かかるため、開発前に事前に設定しておく必要がある（本稿執筆時2014年8月時点の情報。今後Apple社によって変更されることもあるため、最新情報はApple社サイトなどでご確認いただきたい）。

実機の登録

iOSアプリケーションの開発は、Mac上のiOSシミュレータでも実行できるが、実機のiPhoneやiPadとは違う部分も多い。そのため、実際の開発では実機でのテストが必須といえる。

実機をテストで使用するiOSは、Mac上でキーチェーンアクセスを行って登録しておく必要がある。キーチェーンアクセスはMac上の「アプリケーショ

図4　iOSアプリケーション開発環境

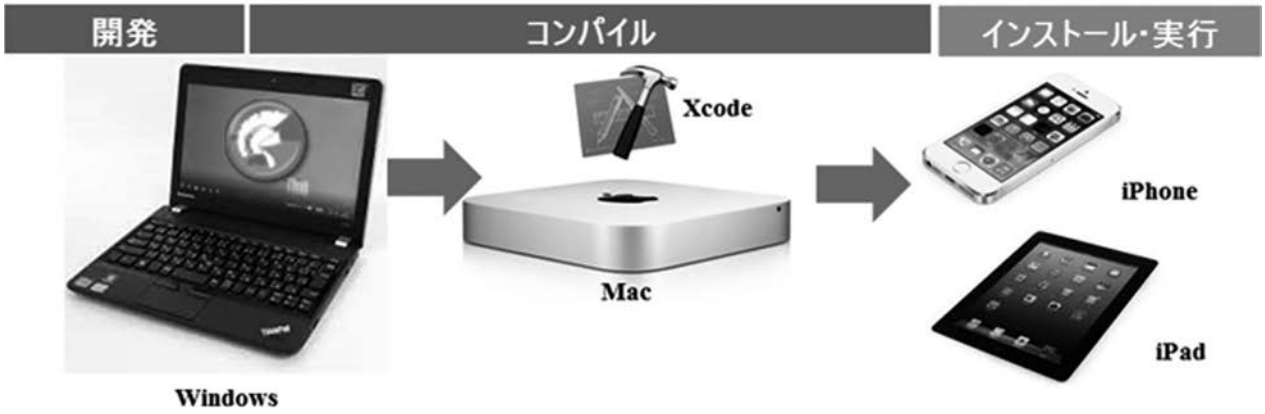


図5　Xcodeの入手

図6　Mac用　PA Server

ン | ユーティリティ」メニューから作業できるので、Apple 社のマニュアルを参考に登録作業を行う。

PAServer のインストール

Delphi/400 の Windows 開発 PC から、コンパイルしたアプリケーションを転送したり、デバッグを行うための PAServer（Platform Assistant Server）を Mac にインストールする。Delphi/400 Version XE5 では、開発 PC の下記フォルダに Mac 向けのインストーラが用意されている。

フォルダパス：
C:\Program Files\Embarcadero\RAD Studio\12.0\PAServer

このフォルダにある「RADPAServer XE5.pkg」を Mac 端末にコピーして、Mac 上でダブルクリックするとインストールが行える。【図 6】

インストールが完了すると、メニューの「アプリケーション」に「RAD PAServer XE5」として登録されているので、PAServer をダブルクリックして起動する。【図 7】

PAServer が起動するとコンソール画面で「Enter キーを押す」と表示されるので、[Enter] キーを押してサービスの開始が完了する【図 8】（iOS7.1 使用の場合は、Hotfix_6（29795）を適用する必要がある）。

接続プロファイルの作成

Delphi/400 開発 PC から、Mac に接続する設定を作成する。Delphi 開発画面の［ツール¦オプション］からオプション画面を開き、「接続プロファイルマネージャ」を選択する。【図 9】

追加ボタンを押すと【図 10】のようなダイアログが表示されるのでプロファイル名を設定して「次へ」を押す。プロファイル名は任意で設定できるので、分かりやすい名前（Mac など）にしておくと使用時に便利である。

次に表示される設定画面で Mac 端末の接続情報を設定し、接続テストが成功すれば完了である。接続テストには「接続テスト」ボタンが用意されている。【図 11】

SDK の取得

Delphi/400 上で対象のデバイス OS に合わせた開発を行うために、SDK の取り込みが必要になる。接続プロファイル同様に Delphi 開発画面の［ツール | オプション］からオプション画面を開き、「SDK マネージャ」を選択する。【図 12】

追加ボタンを押すと【図 13】のダイアログ画面が表示される。プラットフォームに「iOS デバイス」を選択して、接続するプロファイルには作成済のプロファイルを選択して設定する。最後に接続先から対象の SDK バージョンが自動表示されるので、選択して「OK」ボタンを押下する。これだけで、自動的に SDK がダウンロードされて組み込みが完了となる。

3-2. Android向け開発環境

Delphi/400 で Android 向けのアプリケーションを開発する場合、Windows 内に全ての開発環境を構築することができる。【図 14】

必要となる環境

- Windows 端末（Delphi/400 Version XE5）
- Android 実機
  （Android 2.3.3 以降の ARM7 + NEON 対応デバイス）

開発環境の構築

Android の開発環境では、iOS と異なり Windows 端末に全て環境を構築できる。ただし、開発の対象となる Android 実機の PC 接続用ドライバは事前にインストールが必要となる。Android の機種によってインストール方法が異なるため、機種の製造元が提供する方法を確認してインストールを行う。

SDK の取得

Delphi/400 上で対象のデバイス OS に合わせた開発を行うために、Android でも SDK の取り込みが必要になる。Delphi/400 では、Android SDK 専用の管理ツールとして「AndroidTools」が用意されている。

スタートメニューから［Embarcadero RADStudio XE5|Android SDKs］より「Android Tools」でツールを起動することができる。【図 15】

起動すると AndroidSDKManager の画面が表示されるので、開発対象の AndroidOS バージョンにチェックをして、「Install」ボタンを押下する。これだけで必要な SDK を取り込むことができる。Android は iOS に比べて OS のバージョンも非常に多いが、全ての SDK を取り込むとかなりのディスク容量を使用するので、必要なバージョンだけを取り込んだほうがよい。【図 16】

3-3. マルチデバイス開発

ここまでで iOS、Android それぞれの開発環境のポイントを説明したが、実際の開発では、iOS も Android も同じようにネイティブアプリケーションのプログラムを開発することができる。Delphi/400 のネイティブアプリケーションの開発画面は、従来の Windows フォーム設計部にスマートデバイス画面を表示して開発することができ、このスマートデバイス画面に対して、コンポーネント配置、コーディングといった従来の C/S 型、Web 型と同じ手法で開発を行う。【図 17】

スマートデバイス画面は、右上のプロジェクトマネージャにおいて、開発対象となるデバイスが選択できるようになっており、対象のデバイス（iOS、Android、Mac）を指定するだけで、1 つのプログラムからそれぞれのネイティブアプリケーションを生成できる。

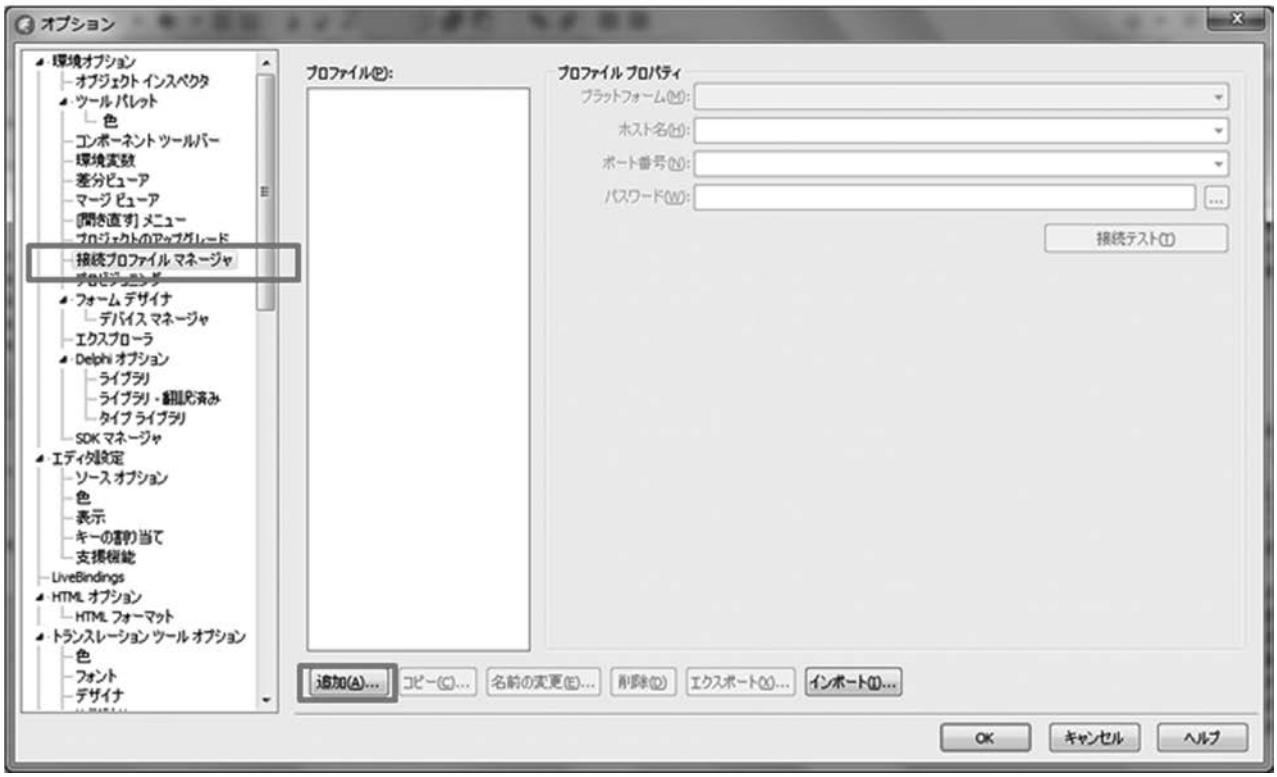
これは FireMonkey と呼ばれる Delphi の新しいフレームワークを使用しており、コンパイル先に指定したデバイス向けのアプリケーションに自動で対応できる。デバイスの違いは FireMonkey フレームワークが吸収してくれるため、開発者は Delphi 言語で開発するだけで、Windows のみならず、スマートデバイスにも対応できる。【図 18】

この開発手法は「マルチデバイス開発」と呼ばれ、Windows や iOS、Android など複数デバイスのアプリケーション開発が必要となる場合に、Delphi/400 ならではの高い生産性を発揮することができる。

図7　PAServerの起動

図8　PAServerのコンソール

図9　接続プロファイルマネージャ



4.ネイティブアプリケーションの開発手順

この章では、Delphi/400 のネイティブアプリケーション開発の流れを、簡単なアプリケーション開発例を題材に詳しく説明していく。

4-1. 基本的な開発手順

ネイティブアプリケーションの開発は先の章でも述べた通り、従来の C/S 型、Web 型のアプリケーションと同様の手順で開発できるのでご安心いただきたい。

基本的な開発の流れは次のようになる。【図 19】

①画面でコンポーネントを配置する
②イベントにプログラミングを行う
③コンパイルして実行（実行時にデバイスにインストールされる）

今回はスマートデバイスのカメラ機能を組み込んだ簡単なアプリケーションを例として、開発の手順を説明していく。【図 20】

ネイティブアプリケーションを新規に作成する場合には、［ファイル｜新規作成］より「FireMonkey モバイルアプリケーション -Delphi」を選択する。

選択時にダイアログでテンプレート選択画面が表示される。【図 21】今回は基本となる「空のアプリケーション」を選択する。他にもリスト形式画面などのテンプレートが用意されているので、新規作成時には便利である。

新規作成されたスマートデバイス画面では、右上のコンボボックスで、デバイスに合わせた設計画面イメージを選択することができる。【図 22】

選択肢としては iPhone や iPad、Android の主要機種（Nexus、Galaxy）が用意されている。また Android の場合、市場の機種が多いため解像度ベースでの画面イメージも選択できる。これは画面表示サイズなどの設計イメージを容易にするための機能なので、開発したプログラムには直接影響しない（Android の見た目で設計して iOS にコンパイルすることも可能だが、表示デザインなどが調整しにくい）。

ここからはアプリケーションの開発内容を説明する。

①画面でコンポーネントを配置する

今回はフォームに ToolBar、Button、Image、ActionList のコンポーネントを【図 23】のように配置する。

Button は StyleLookup プロパティで、選択しているデバイスのイメージに合わせた表示スタイルが設定できるようになっているので、カメラのアイコンになるスタイルを設定しておく。他のコンポーネントも同様に StyleLookup プロパティで表示スタイルの設定が可能である。別のデバイスを切り替えた場合には、そのデバイスに用意された同様の表示スタイルが自動で適用される。

②イベントにプログラミングを行う

アプリケーションの処理は、コンポーネントのイベントにコーディングすることができる。今回は Button にカメラ機能を使うイベントを設定してコーディングを行う。

Button の Action プロパティで「標準アクションの新規追加」から「メディアライブラリ「TTakePhotoFromCameraAction」を設定する。【図 24】

これだけで、スマートデバイスのカメラ撮影機能を Button で利用することができる。

イベントタブには OnDidFinishTaking というイベントがあるので、このイベントをダブルクリックしてコーディング処理部分を作成する。OnDidFinishTaking イベントはカメラ撮影が終わったあとに実行されるイベント処理である。【図 25】

コーディングする内容は【ソース 1】のように 1 行だけ記述する。このデバイスで撮影された画像を画面の Image コンポーネントにセットするというプログラムコードである。

③コンパイルして実行

ここまでの作業でネイティブアプリケーションのプログラム自体は完成している。最後にコンパイルを行ってアプリケーションの動作を確認する。

プロジェクトマネージャ画面にコンパイル先のデバイスが選択できるようになっている。【図 26】

選択できるのは Android、iOS シミュレータ、iOS デバイスである。それぞれ「ターゲット」という部分に接続しているデバイスの端末名が表示されるので、端末名をダブルクリックして選択する。

今回は iOS のデバイスを選択してコンパイル実行（メニューの実行、または F9）でアプリケーションを生成して実行してみる。実行するとコンパイル完了あとに、iOS の実機上でアプリケーションがインストールされ、作成したカメラアプリケーションが起動する。アプリケーションのボタンを押すとカメラ機能が起動し、撮影を行うことができる。撮影した画像はアプリケーションの画面にセットされる。

これだけでカメラ機能を連携した iOS ネイティブアプリケーションが完成したことになる。【図 27】

それでは Android ネイティブアプリケーションでは、どのように開発するかというと、実は今作成したプログラムのコンパイル先を変更するだけでよいのである。

プロジェクトマネージャの「ターゲット」に表示される Android 端末を選択して、コンパイルしてみる。すると、プログラムは 1 行も変えていないので、同じアプリケーションが Android 上にインストールされて実行される。【図 28】

これが先に説明した Delphi/400 のマルチデバイス開発である。

1 つのプログラムからコンパイル先の指定だけで、複数のデバイスに対応できる。この開発手順を試していただくと、簡単にスマートデバイス向けのアプリケーションが開発できることを実感していただける。

4-2. IBM i 活用手順

先の例では、ネイティブアプリケーションの基本的な開発手順を説明してきた。ここからはネイティブアプリケーションから、Delphi/400 の機能を使って IBM i へ接続する方法を説明する。

通常、PC から社内のデータベースに接続する場合は、PC にデータベース接続用のドライバをインストールしている。

しかし、スマートデバイスでは、社内のデータベースに直接接続することはできない。これは IBM i に限らず、Oracle や SQLServer など、どのデー

図10　接続プロファイルの追加

図11　接続プロファイルの設定

タベースでも同じである。その理由は、iOS や Android などのデバイス上には、データベースに接続するためのドライバがインストールできないからである。そのため、スマートデバイスのネイティブアプリケーションから IBM i に接続する場合には、【図 29】のようにアプリケーションサーバーを経由した 3 階層方式の接続となる。

アプリケーションサーバーには、C/S アプリケーション同様に IBM i に接続するサーバーアプリケーションが必要になる。Delphi/400 では、このサーバーアプリケーションにも「DataSnap」と呼ばれる専用開発機能が用意されており、容易に開発が可能である。

サーバーアプリケーションには、SQLConnection や SQLQuery などの DB コンポーネントを設定したり、処理関数をプログラミングすることで機能を実装する。

「DataSnap」を使ったサーバーアプリケーションの開発方法は、本稿では割愛させていただくが、『Migaro. Technical Report No.5』に「DataSnap を使用した 3 階層アプリケーション構築技法」というレポートで詳しくまとめているので、こちらを参考にしていただきたい。

それでは、この「DataSnap」のサーバーアプリケーションに接続するネイティブアプリケーションの開発手順を説明する。

今回は、IBM i 上の得意先マスタ（CUSTOMER ファイル）の一覧を表示するアプリケーションを例として、開発手順を確認していく。【図 30】

①画面でコンポーネントを配置する

まずフォームに ToolBar、Label、Switch、ListView を【図 31】のように配置する。Label には“得意先一覧”とタイトル名を設定しておく。

スマートデバイス上に配置する Label では、Label の表示文字が部品以上に長い場合、“得意先…”というように画面上で自動省略されてしまう。そのため、Label の AutoSize プロパティを True に設定しておくことをお勧めする。この設定をしておくと、表示文字の長さに合わせて部品のサイズ側が自動調整してくれる。【図 31】

次に IBM i の接続コンポーネントを配置する。フォームに SQLConnection、DSProviderConnection、ClientDataSet を【図 32】のように配置する。

それぞれのコンポーネント設定内容を説明する。SQLConnection コンポーネントは ConnectionName プロパティに“DataSnapConnection”を設定する。【図 33】

そして Params プロパティに、アプリケーションサーバーの IP アドレスとポート番号を設定しておく。これにより「DataSnap」のサーバーアプリケーションに接続することができる。また、スマートデバイス上ではダイアログが出ないので、LoginPrompt プロパティは False に設定しておく必要がある。

次に、DSProviderConnection コンポーネントを設定する。【図 34】

SQLConnection プロパティには、先ほど設定した SQLConnection を指定する。ServerClassName プロパティには、サーバーアプリケーションで作成しているクラス名を設定するが、デフォルト名では“TServerMethods1”を指定する。

最後に ClientDataSet コンポーネントの設定を行う。【図 35】

RemoteServer プロパティに、先ほど設定した DSProviderConnection を指定する。この設定を行うと、ProviderName プロパティにサーバーアプリケーションの Provider が自動で表示されるので選択する。最後に CommandText プロパティに IBM i にアクセスしたい SQL 内容をセットする。

今回は得意先マスタ（CUSTOMER ファイル）にアクセスするため、次のような SQL を記述する。

“SELECT * FROM CUSTOMER”

ここまでの設定で IBM i へ接続して、得意先マスタのデータにアクセスすることができる。ClientDataSet をダブルクリックして、リストを右クリックから「すべてのフィールドの追加」を選択する。これで IBM i から得意先マスタの項目を取り込むことができる。

次に、アクセスしたデータをアプリケーションの画面上に表示する部分を作成する。C/S 型、Web 型アプリケーションの場合、DBGrid などのコンポーネントが便利だが、残念ながら FireMonkey のフレームワークには同じコンポーネントが存在しない。そのため、今回は LiveBinding というビジュアルリンク機能を使用する。LiveBinding は取得したデータを画面上のコンポーネントに自動でリンクしてくれる便利な機能である。これは簡易作成機能なので、もちろんプログラミングでデータをセットしても問題ない。

画面に配置した ListView にデータを表示するには、フォームを右クリックから「ビジュアルにバインド」を選択する。【図 36】

開発画面下部にビジュアルバインディングの設計画面が起動されるので、この画面で項目のリンク設定を行う。

リンクの方法は簡単である。データ項目と表示したいコンポーネントの項目をドラッグ＆ドロップするだけで、感覚的にリンクを設定できる。【図 37】

設定ができたら ClientDataSet の Active プロパティを True に設定すると、表示結果を確認できる。【図 38】

このように、実際に開発設計画面上に IBM i のデータがリンク表示されるので、プログラムをコンパイルしなくとも、画面を細かく調整することができる。

②イベントにプログラミングを行う

アプリケーションで得意先マスタのデータを表示 / 非表示ができるように Switch コンポーネントの OnSwitch イベントにコーディングを行う。プログラムを【ソース 2】のように 1 行だけ記述する。

これでスィッチの ON/OFF によって、データの表示 / 非表示を操作できる。データの表示制御は LiveBinding が自動で行ってくれるので細かいプログラミング制御は必要ない。

③コンパイルして実行

ここまでの作業でプログラムは完成である。プロジェクトマネージャで対象のデバイスを選択して、コンパイルを実行する。マルチデバイス開発なので、iOS でも Android でも可能である。【図 39】

このように IBM i のデータを活用するネイティブアプリケーションも、ほとんどプロパティの設定だけで簡単に開発

図12　SDKの追加

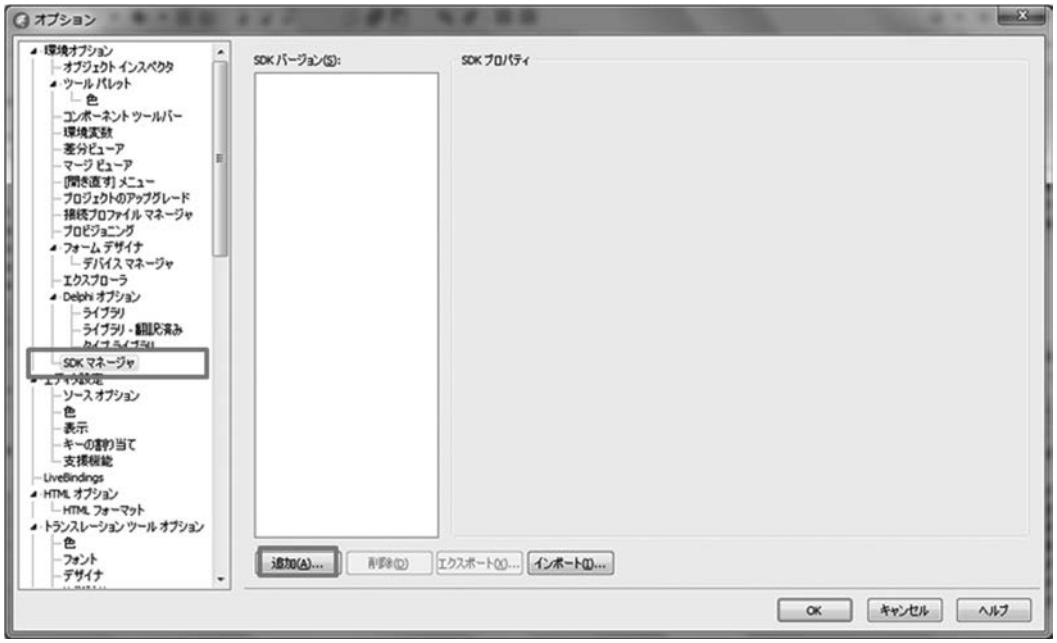


図13　SDKの選択

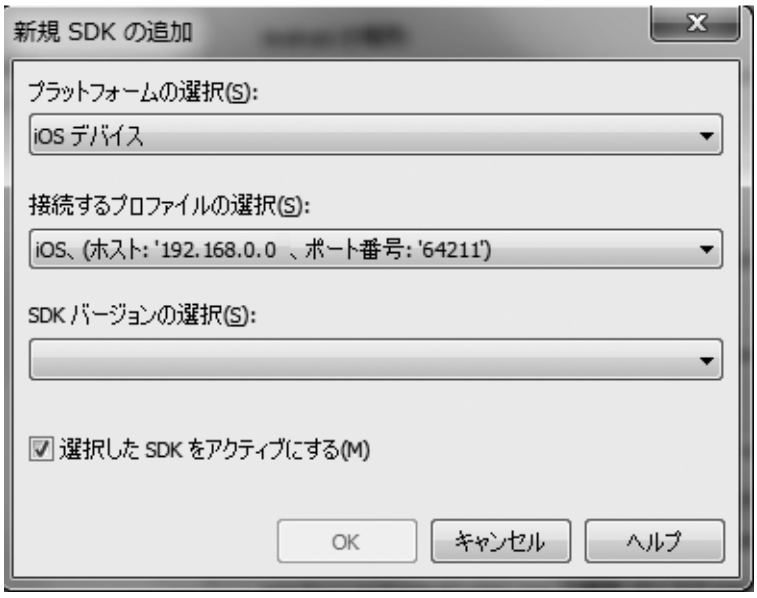


図14　Androidアプリケーション開発環境

できる。例えば、この得意先マスタの一覧のデータをタッチすることで、得意先の受注一覧を表示するといった機能カスタマイズも、同じような手順で作成できる。【図 40】

こうしたアプリケーションであれば、コンポーネントの設定と数行のコーディングで開発できてしまう。

5.ネイティブアプリケーション開発のポイント

前章までは Delphi/400 のネイティブアプリケーションの開発手順について説明してきた。この章では、ネイティブアプリケーション開発時にヒントになりそうなポイントをいくつか補足したい。

5-1. iOSとAndroidの違い

Delphi/400 では、iOS でも Android でも 1 つのプログラムで開発できるが、iOS と Android ではデバイスの違いがあるため、設計上でいくつか考慮しておく点がある。

1 つはハードウェアキーの違いである。Android には「ホームボタン」や「戻るボタン」「メニューボタン」が物理的に存在するが、iOS には「ホームボタン」しか用意されていない。【図 41】

例えば、iOS で「戻るボタン」が前提のアプリを作成してしまうと意図した画面遷移操作が行えなくなってしまう。そのため、OS・ハードの違いを把握した画面設計は非常に重要となってくる。

また、デバイスの構造が違うので、アプリケーション内でファイルを扱う場合にも考慮が必要である。

例えば、アプリケーションで音声や動画を流したりする場合には、オーディオファイルなどをアプリケーションと一緒に配布する必要がある。

しかし、当然ながらデバイス上の構造が違うので、ファイルを保存するためのパスも違ってくる。従って、ファイルのパスなどは、iOS/Android ごとに設定をしておく必要がある。

ファイルの配置は［プロジェクト｜配置］からデバイスごとに設定できるので、iOS であれば“.¥StartUp¥Documents¥”、Android であれば“assets¥internal¥”に設定する。【図 42】（アプリケーションの外に配置する場合はパスも異なる）

こうしたアプリケーション固有の配置ファイルパスをプログラムで取得する場合には、コーディングも違ってくる。配置した Alerm.mp3 というオーディオファイルを TMediaPlayer コンポーネントに設定する場合であれば、【ソース 3】のように記述することができる。

デバイスごとに異なるプログラムコードは、iOS であれば｛$IFDEF IOS｝、Android であれば｛$IFDEF ANDROID｝というタグを記述しておけば、特定のデバイス実行時のみ有効なコーディングも可能である。

もちろん配布したファイルを読み込むだけでなく、C/S 型アプリケーションのように設定ファイルをデバイス上に作成・保持することもできる。またネットワークに接続されていない場合に、ローカル環境で動作するように CSV などのファイルデータをデバイス内で保存し、ネットワークにつながってから IBM i にデータを登録するといったことも実現できる。

5-2. アプリケーションのカスタマイズ設定

プログラミングとは別に、ネイティブアプリケーションで設定しておけるカスタマイズ設定も補足しておく。

アイコンのカスタマイズ

例えば、スマートデバイスにインストールしたアプリケーションのアイコンも設定が可能である。［プロジェクト｜オプション］から「アプリケーション」を選択するとデバイスごとにアイコンを設定できるようになっている。【図 43】

ここで png などの画像ファイルで作成した任意のアイコンを設定すると、インストール時に自社用のアイコンで登録することも可能である。アイコンはデバイスによって解像度がさまざまなので、対象のデバイスに合わせたサイズのアイコンを用意する必要がある。

デバイス向きのカスタマイズ

また、同じ［プロジェクト｜オプション］から「アプリケーション」の設定画面で「向き」というタブを選択すると、デバイス固有の向き設定を固定化することもできる。【図 44】

標準では縦横の画面変更時に表示調整が行われるが、例えば縦専用で設計した画面の場合、この設定を行えば横表示にならないようにアプリケーション画面を縦固定にすることができる。

セキュリティ権限のカスタマイズ

他にも Android アプリケーションの機能で、特別な権限が必要な場合は、［プロジェクト｜オプション］から「使用する権限」の設定画面で、権限を付与したり、あるいはセキュリティ上で制限したりすることも可能である。設定はチェックの ON/OFF だけで、かなり細かい設定まで行うことができる。【図 45】

5-3. ネイティブ機能の連携例

先に説明した開発手順では、ネイティブ機能連携の一例としてカメラ機能の連携開発を説明したが、他にもさまざまなネイティブ機能を連携することができるので、一例を紹介しておきたい。

例えば、カメラ機能を応用して、バーコードを読み取ったり【図 46】、GPS の位置情報を利用して GoogleMap を利用することもできる。【図 47】

また、加速度センサーなどを使用すれば、デバイスの傾きなどを利用した画面操作を行うことも可能である。さらに、音声データを録音・再生したり、先にも紹介したアイコンに通知を表示することもできる。【図 48、図 49】

この章で紹介した設定機能やネイティブ連携機能は、Web アプリケーションでは実現できない内容も多く、ネイティブアプリケーションならではの機能性、拡張性の高さといえる。

6.ネイティブアプリケーションの配布・運用

この章では開発したアプリケーションをユーザーのスマートデバイスに、どのように配布して運用するかについて、説明する。

6-1.社内公開と一般公開

ネイティブアプリケーションのスマートデバイスの配布には大きく、社内公開と一般公開の配布方法がある。それぞれの特徴は次の通りである。【図 50、図 51】

図15　Android Toolsの起動

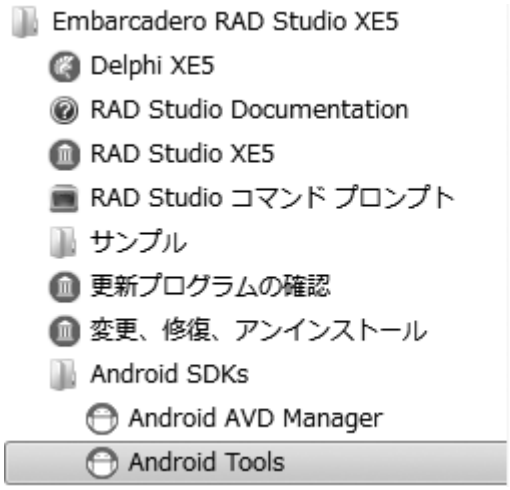


図16　Android SDK Manager

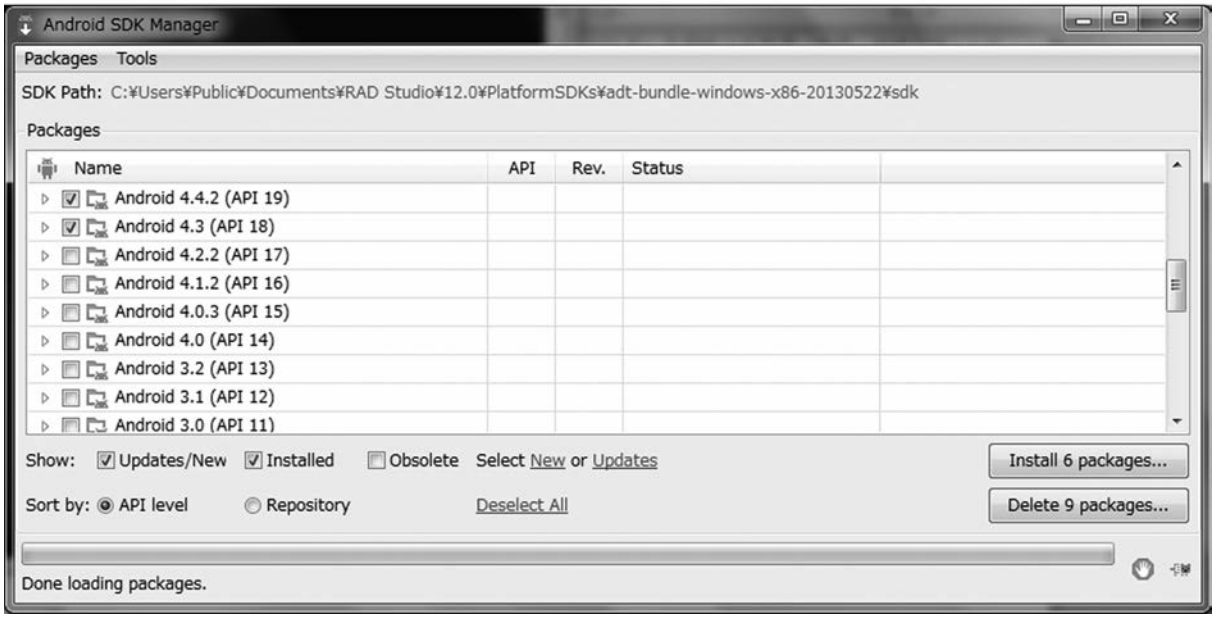


図17　スマートデバイスアプリケーション開発画面

社内公開

社内公開の場合、開発したネイティブアプリケーションを社内用のWebサーバーに配置して、各スマートデバイスからインストールする。社内専用のアプリケーションであれば、自由に開発、運用することができるため、ストアなどの審査も必要としない。アプリケーションの配布タイミングも制約はないので、自由にリリースすることができる。

ただしiOSの場合には、開発環境の章でも説明した通り、iOS Developer Programに加入しなければ配信することができず、また配信台数もプログラムの種類によって制限されるので考慮が必要である。

一般公開

一般公開の場合、開発したネイティブアプリケーションを専用ストアから配信して、各スマートデバイスからインストールする。専用ストアはiOSであればApp Store、Androidであれば GooglePlayStoreを利用することになる。

ストアを経由して配信するため、インストールも容易となるが、誰でもインストールすることができてしまうため、ストア配信時には審査がある。そのため、社内利用限定のアプリケーションは配信することは難しく、またリリース後に変更があっても、すぐにリリースはできない。

6-2. ネイティブアプリケーションの配布ファイル

社内公開でWebサーバーから配信する場合、アプリケーションファイルを配置してHTMLからリンクすることになる。

アプリケーションファイルはコンパイルの際に生成することができる。iOSであれば、ipa、plistファイル、Androidであればapkファイルが配布の対象となるので、これらをWebサーバー上に配置する必要がある。

HTMLのリンクの記述内容は難しくはないが、例を記載しておくので参考にしていただきたい。【iOS：ソース4, Android：ソース5】

またiOSの場合、Webサーバー上に配置するplistファイルはipaファイルへのURLリンクパスを含めて作成する必要がある。メモ帳などで編集できるので、以下のように作成する。【ソース6】

6-3. Webサーバーの設定・考慮点

Webサーバーは IISや Apacheなど使用することができ、htmlは先に説明した内容を公開すればアプリケーションを配布できる。

ここでもiOSでは、1つ注意点がある。

iOS7.1からはApple社での仕様が大きく変わり、SSLでの配信でなければ、インストールを行うことができなくなっている。つまり、通常のhttpでの配信を行えないので、httpsに対応したSSL配信ができるWebサーバー環境が必要になってくるので注意していただきたい（iOS7.0まではhttpで配信が行える）。

また、「6-2」で説明した配布アプリケーションファイルの拡張子は、Webサーバーに MIMEタイプとして設定を登録しておく必要がある。IISの場合は、IISマネージャを使用して、サーバーの「プロパティ」ページで次の MIMEタイプを追加する。Apacheの場合は、mime.typesに追加する。各拡張子の設定内容は次の通りである。

ipa ファイル
　application/octet-stream
plist ファイル
　text/xml
apk ファイル
　application/vnd.android.package-archive

ここまで設定が一度完了すれば、アプリケーションを開発するごとにファイルを配置して、リンクの追加だけで配布・運用することができる。

7.まとめ

本稿ではDelphi/400の新機能であるiOS/Android向けの開発手順や運用ポイントを説明してきた。スマートデバイス開発と聞くと、敷居が高く感じられる開発者の方も多いだろうが、Delphi/400では、従来のC/S、Web型と同様の手法でスマートデバイスのネイティブアプリケーショも開発できる。

冒頭でふれた通り、これからスマートデバイス向けに社内アプリケーションが必要とされる機会が増えてくる。

しかし、忘れてはならない重要なポイントがある。それは、これまで使用しているWindowsの社内システムがスマートデバイスの社内システムに置き変わることは少ないということである。

実際にWindows開発者に対して実施されたアンケート調査結果では、モバイルアプリケーションはデスクトップアプリケーションの置き換えにはならないという意見が9割を超えていた。【図52】

これはスマートデバイスとPCは使い勝手や用途が異なるので、Windowsの社内システムは今後も必要とされるということを意味している。つまり今後は、両システムの併用が一般的になるかもしれない。

そうしたとき、今回紹介したDelphi/400のマルチデバイス開発は、これまでのWindows向けの開発スキルをスマートデバイスでも活かせる技術であり、社内開発を拡張していくための最適解の1つといえる。Delphi/400を使って、今後の社内システムでスマートデバイスに対応される場合には、本稿を開発の足がかりにしていただければ幸いである。

M

図18　FireMonkeyフレームワーク

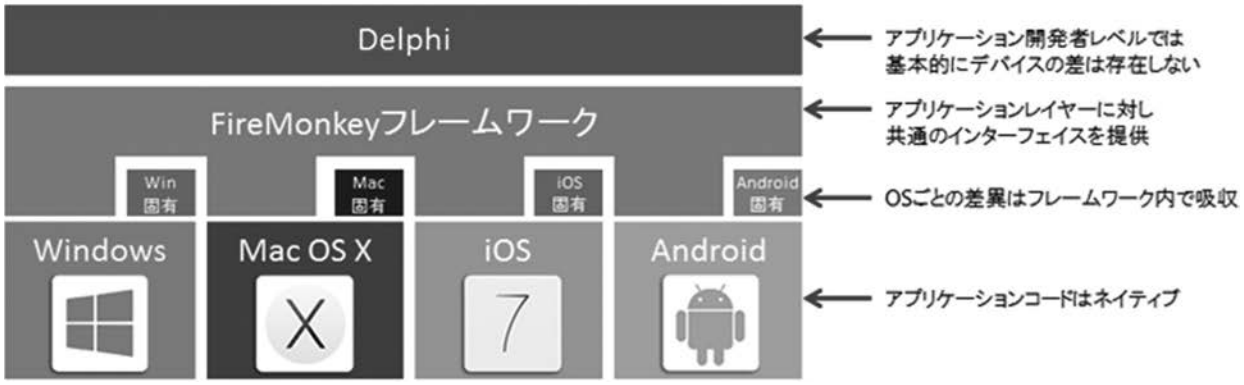


図19　開発の流れ

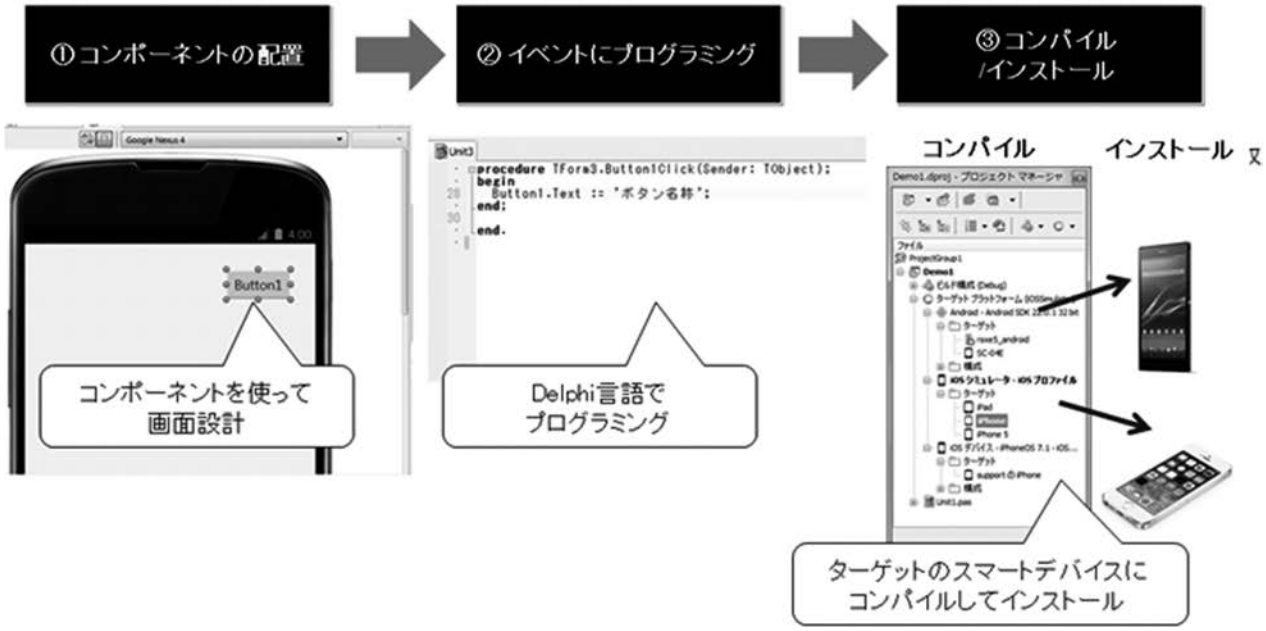


図20　カメラアプリ

図21　テンプレートの選択

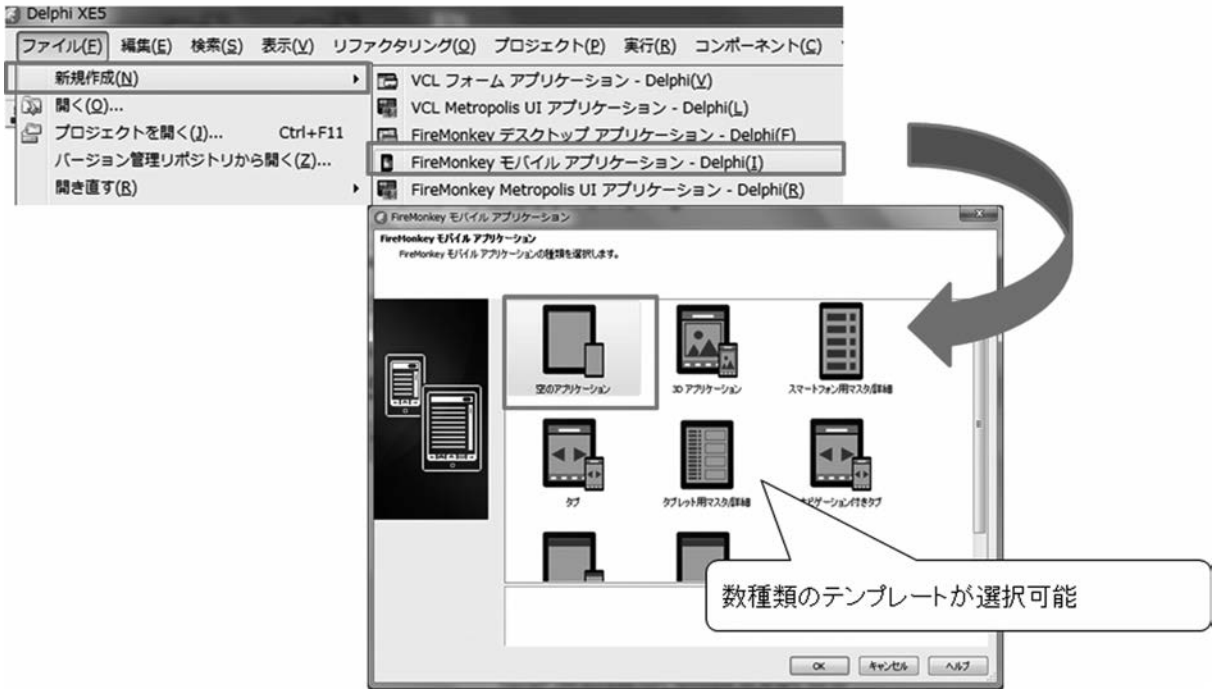


図22　設計デバイスイメージ

図23　コンポーネントの配置

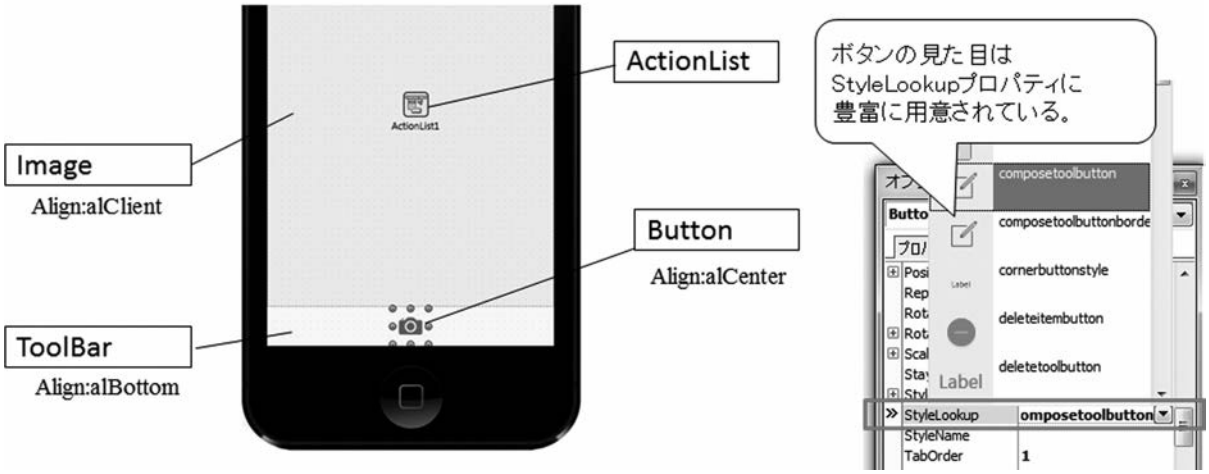


図24　Actionの設定

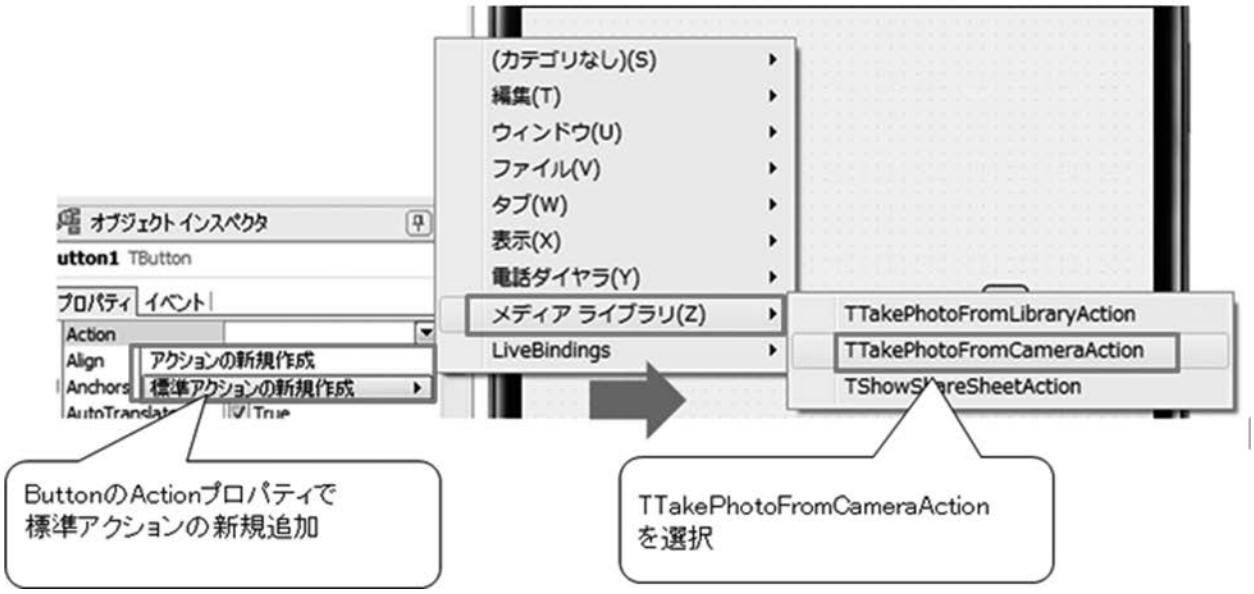


図25　カメライベントの設定

ソース1

OnDidFinishTaking処理（カメラ撮影終了処理）

```
procedure TForm1.TakePhotoFromCameraAction1DidFinishTaking(Image: TBitmap);
begin
  Image1.Bitmap.Assign(Image);
end;
```

図26　コンパイルターゲット

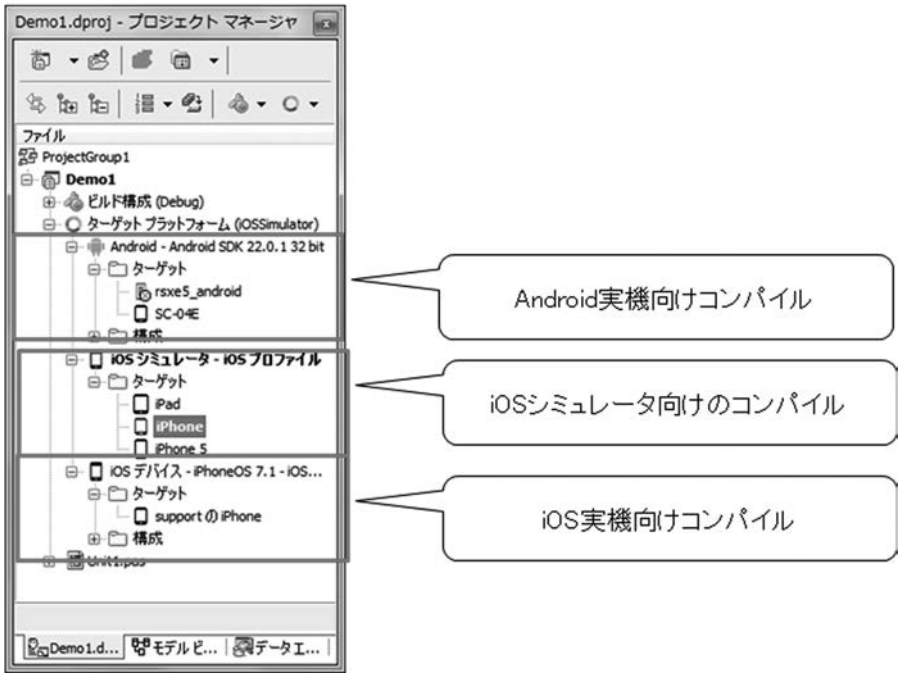


図27　iOSアプリケーションの実行

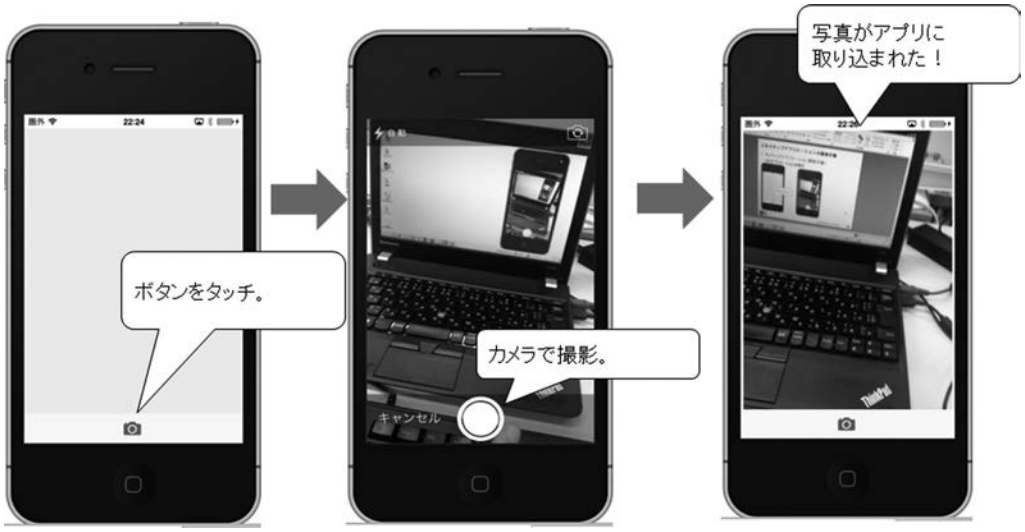


図28　Androidのアプリケーションの実行

図29　IBM iへの3層接続

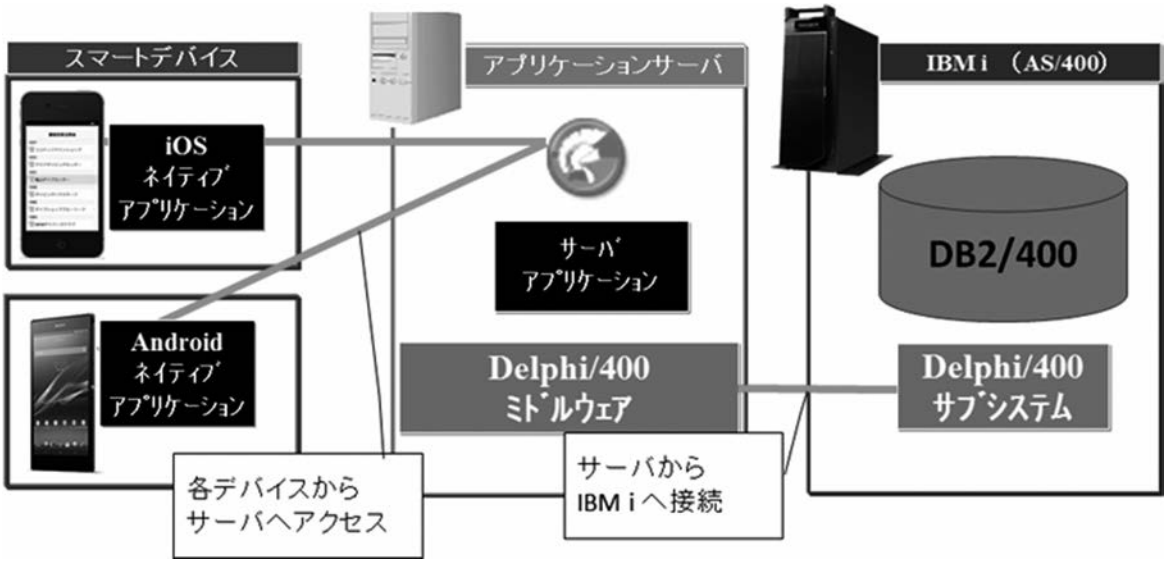


図30　得意先一覧照会アプリ

図31　コンポーネントの配置

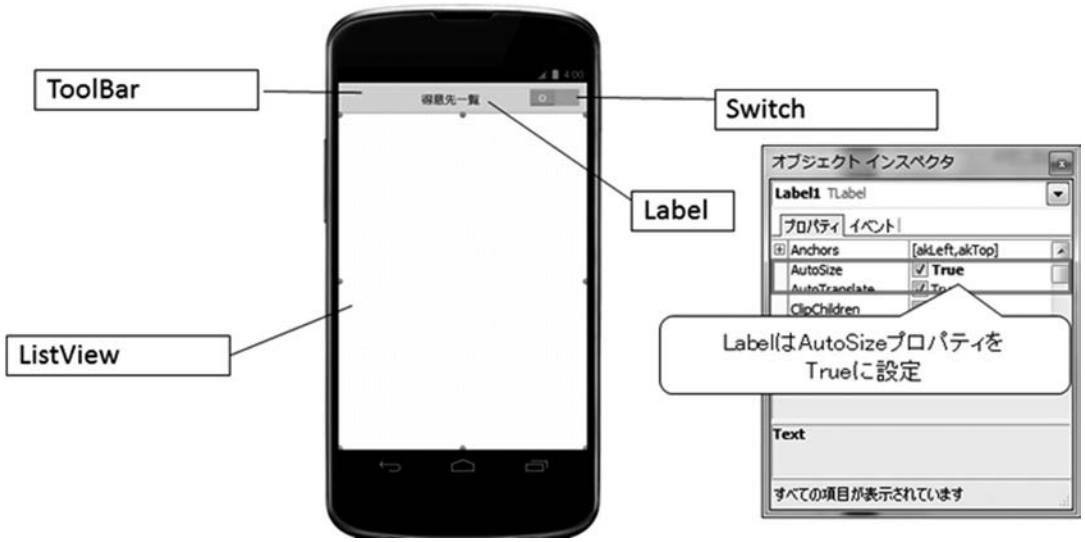


図32　DBコンポーネントの配置

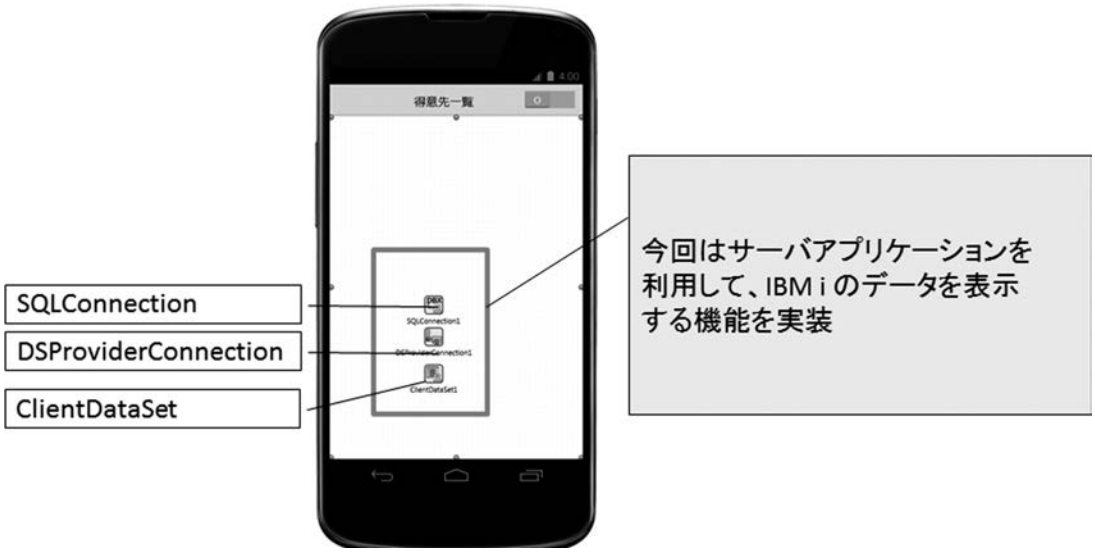


図33　SQLConnectionコンポーネント設定

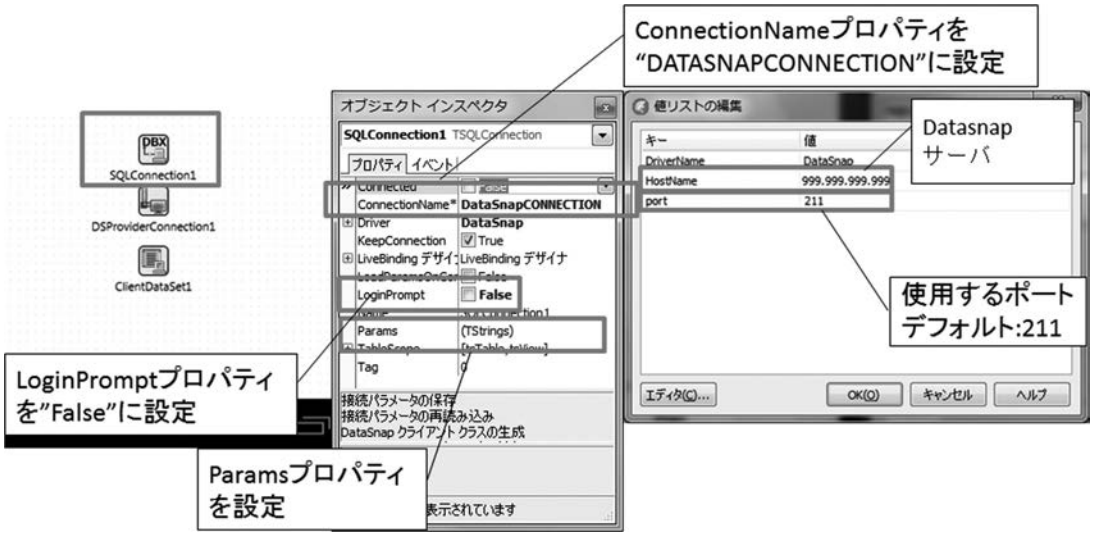


図34　DSProviderConnectionコンポーネント設定

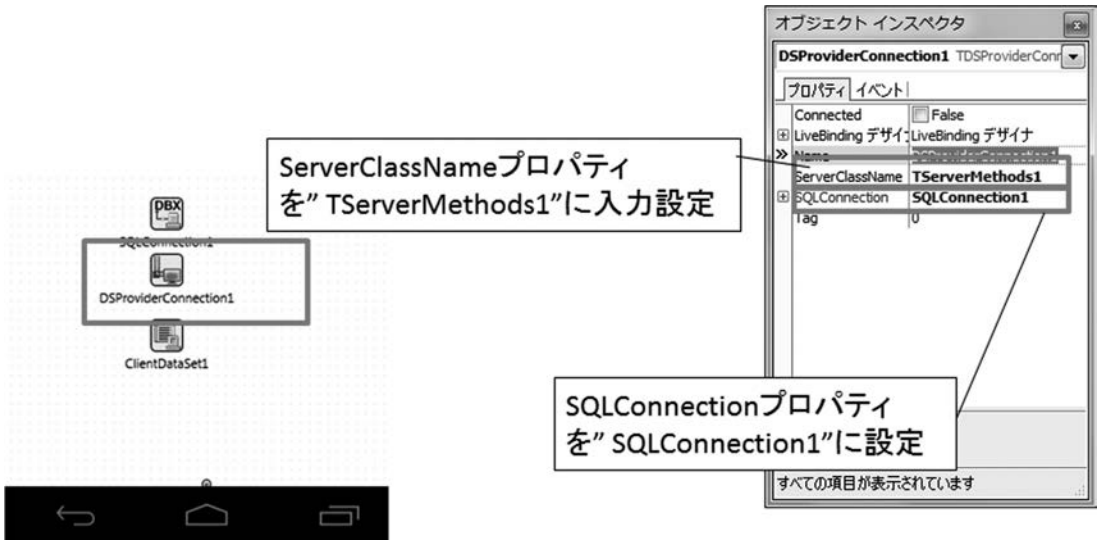


図35　ClientDataSetコンポーネント設定

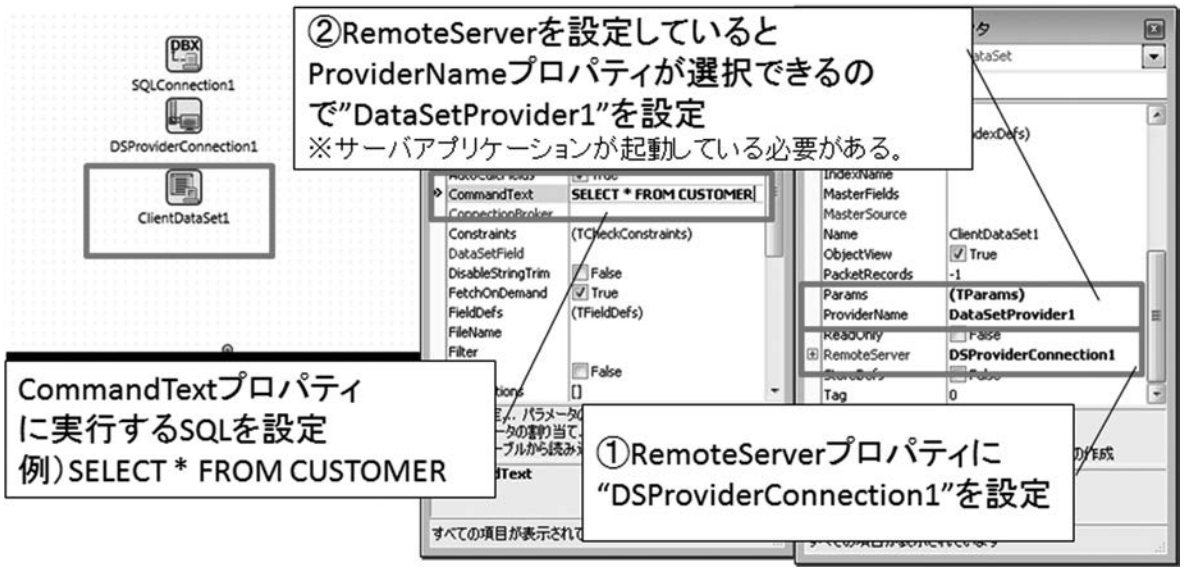


図36　LiveBindingの設定

図37　LiveBindingのリンク機能

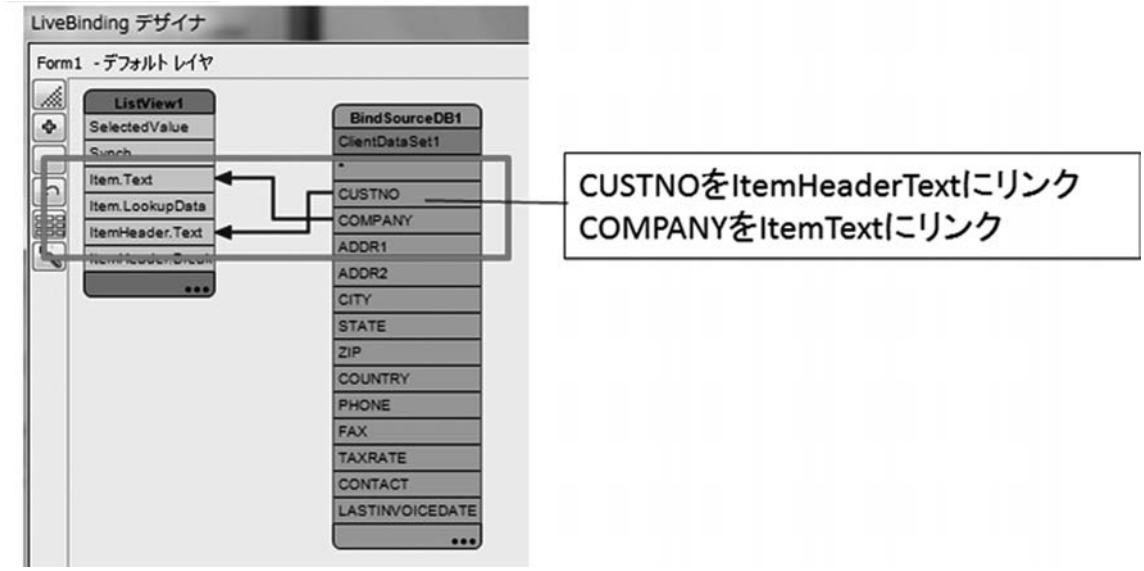
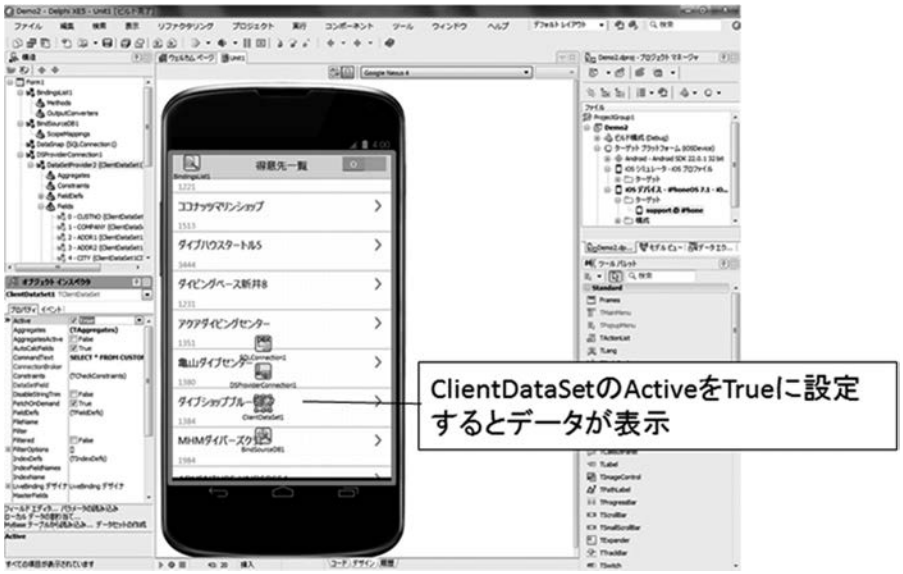


図38　IBM iデータの表示



ソース2

スイッチ切り替え処理

```
procedure TForm1.Switch1Switch(Sender: TObject);
begin
  ClientDataSet1.Active := Switch1.IsChecked;
end;
```

図39　iOS/Androidアプリの実行

図40　カスタマイズ例

図41　デバイスの違い

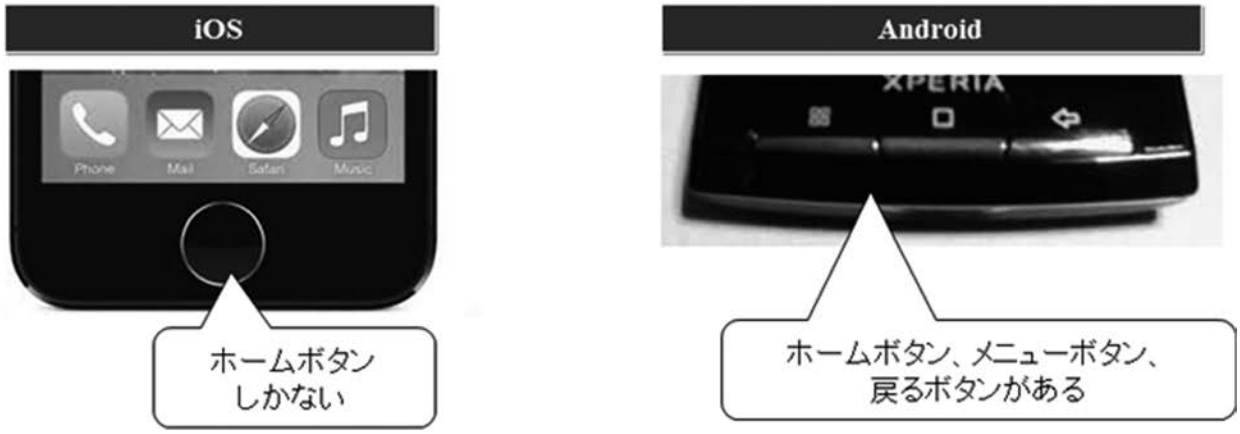


図42　ファイル配置の設定

ソース3

ファイルパスの取得例（iOS)

```
MediaPlayer1.FileName := GetHomePath +PathDelim + 'Documents' + PathDelim + 'Alerm.mp3';
```

ファイルパスの取得例（Android)

```
MediaPlayer1.FileName := TPath.Combine(TPath.GetDocumentsPath, 'Alerm.mp3');
```

図43　アイコンの設定

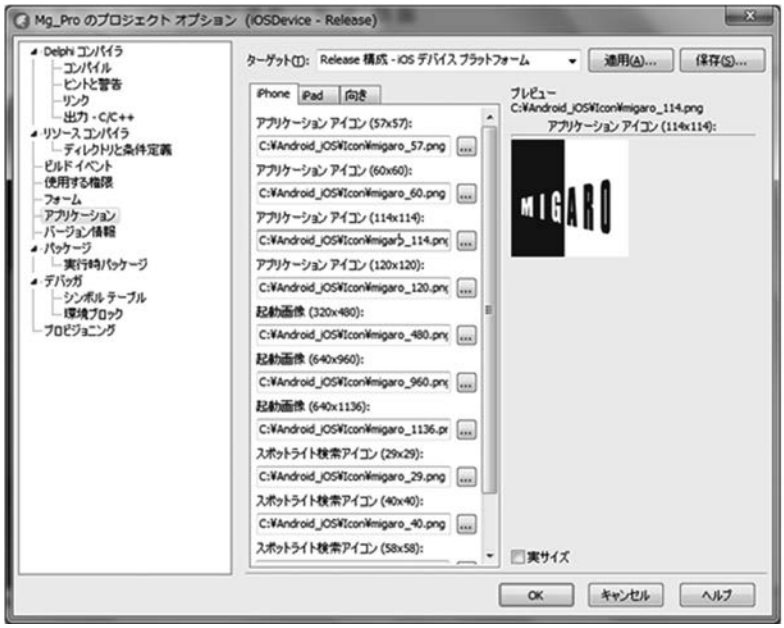


図44　デバイス向きの設定

図45　セキュリティ権限の設定

図46　バーコードの連携

図47　位置情報GPSの連携

図48　音声レコーディング連携

図49　通知機能連携

図50　配布の種類

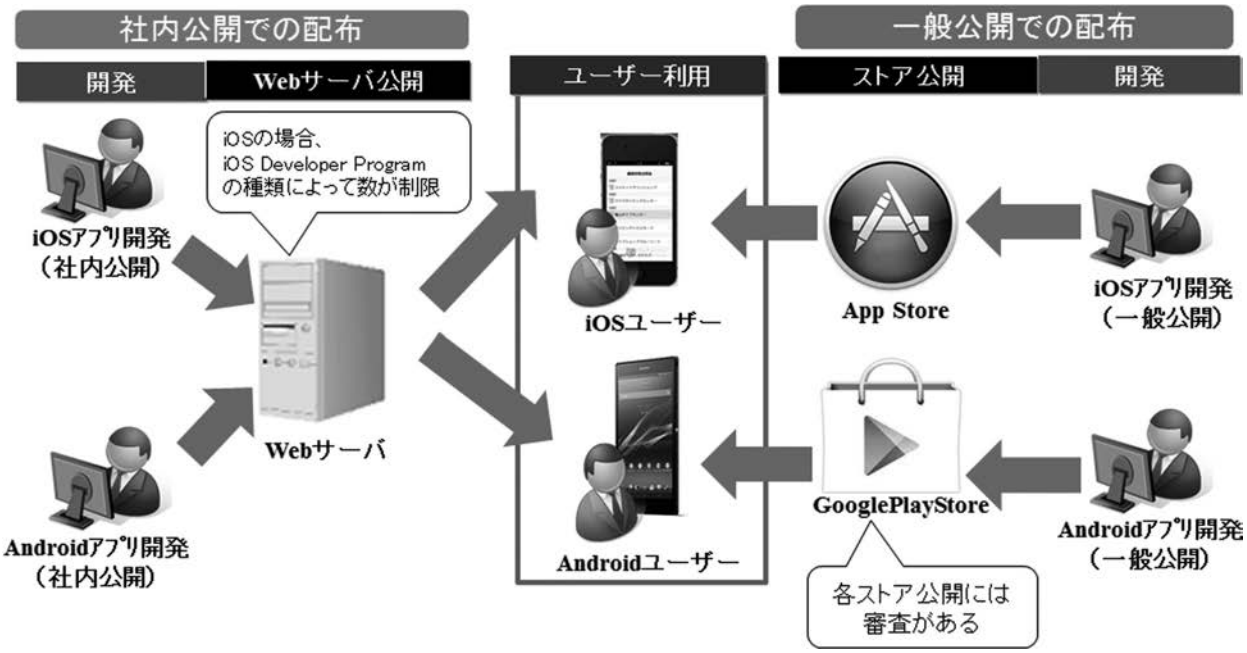


図51　メリット・デメリット

	社内公開	一般公開
メリット	・社内だけで配布・利用できる。 ・審査がないため、社内専用のアプリケーションが開発できる。	・ストアで公開するため、どこからでもすぐにインストールして利用できる。
デメリット	・Webサーバ等を用意して、配布環境の構築・運用が必要。	・誰でも利用できてしまう。 ・公開には審査が必要。 （自社用アプリの公開は難しい）

ソース4

ダウンロード用HTML例(iOS)

```
<h1>iOSダウンロードサイトサンプル</h1>
<form>
  <a href="itms-services://?action=download-manifest&url=https://Webサーバ/Sample.plist">
   アプリケーションダウンロード</a><br>
</form>
```

ソース5

ダウンロード用HTML例(Android)

```
<h1>Androidダウンロードサイトサンプル</h1>
<form>
 <a href="./Sample.apk" type="application/vnd.android.package-archive">アプリケーションダウンロード</a><br>
</form>
```

ソース6

ダウンロード用plist例

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>items</key>
        <array>
                <dict>
                        <key>assets</key>
                        <array>
                                <dict>
                                        <key>kind</key>
                                        <string>software-package</string>
                                        <key>url</key>
                                        <string>https://Webサーバ/Sample.ipa</string>
                                </dict>
                        </array>
                        <key>metadata</key>
                        <dict>
                                <key>bundle-identifier</key>
                                <string>com.hogehoge.hogehoge</string>
                                <key>kind</key>
                                <string>software</string>
                                <key>title</key>
                                <string>Sample</string>
                        </dict>
                </dict>
        </array>
</dict>
</plist>
```

図52　今後の開発方針

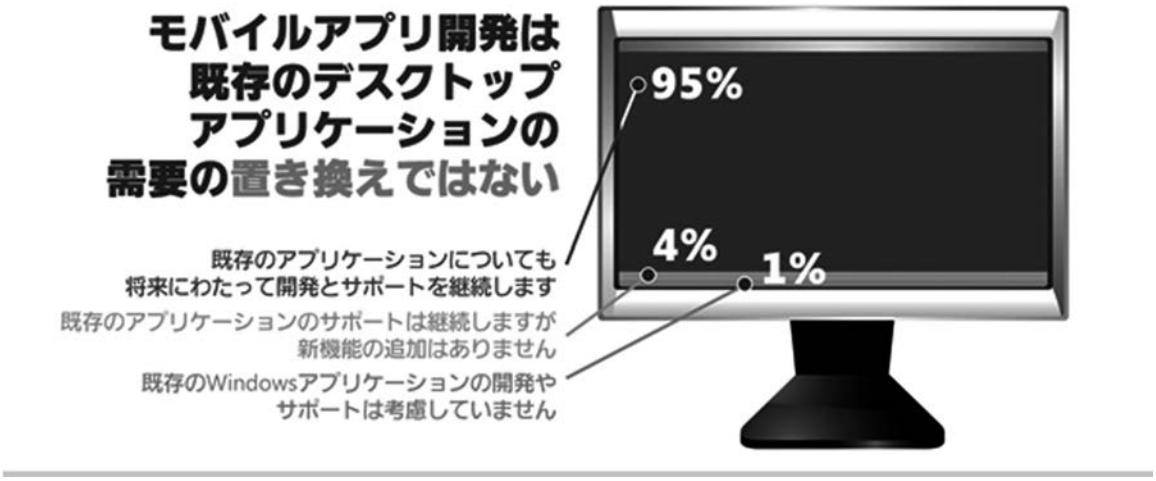


図53　PCアプリケーションとスマートデバイスアプリケーションの特徴の違い

PCアプリケーションの特徴	スマートデバイスアプリケーションの特徴
細かいキーボード入力ができる	どこでもすぐに使用できる
画面が大きく見やすい	タッチで感覚的に操作できる
ネットワークが安定している	カメラ、GPS、センサー等が利用できる
携帯性が低い	細かいキーボード入力は不向き

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

[Delphi/400] ファイル加工プログラミングテクニック —ファイルの圧縮・展開

略歴
1973 年 5 月 26 日
1996 年 関西大学 工学部卒
2002 年 3 月 株式会社ミガロ. 入社
2002 年 3 月 RAD 事業部配属
2007 年 4 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

最近はメール 1 つ取ってもプレーンテキストで書かれることは減り、文字装飾や画像・添付ファイルが加えられるといったように、何かにつけデータ容量が増大しつつある。

その半面、固定回線のような安定した高速・大容量回線から通信速度や転送量に制約のあるモバイル環境へのシフトが進み、データの圧縮・展開技術の重要度が増している。

上記を受け、本レポートでは代表的なファイルの圧縮･展開形式を確認し､ファイルの圧縮・展開といった加工を行うためのポイントをご紹介する。

2.代表的な圧縮･展開ファイル形式

“データ圧縮”とすると、アナログ技術を使った通信における帯域圧縮や画像・音声などに使われる非可逆圧縮も含まれるため、ここでは Windows でよく使われるアーカイブ機能（※ 1）を持ち、可逆圧縮（※ 2）を行うファイル形式に限定して取り扱うこととする。

（※ 1）アーカイブ機能

アーカイブとは書庫の意味であり、複数のファイルを 1 つのファイルにまとめることを指す。通常、アーカイブと同時にデータ圧縮が行われるが、厳密な話をするならアーカイブとデータ圧縮は別物である。

（※ 2）可逆圧縮

元のデータを損なうことなく、圧縮前のデータを完全に再現可能な圧縮方法のことを指す。画像や音声などは再現性よりも圧縮率を取ることが多く、人間があまり強く意識しない成分を無視したり、数式などで近似したりすることでデータをより圧縮する非可逆圧縮になっていることが多い。

Windows の世界で代表的なファイル圧縮・展開形式と言えば、ZIP 形式がまず挙げられる。次いで、Windows Installer などで用いられる CAB 形式が有名である。日本国内限定であれば、圧縮アルゴリズムを含め純国産の LZH 形式も非常に有名であったが、こちらは現在、開発者サイドから使用中止が呼びかけられている。これらのファイル形式について詳細を確認していく。

ZIP形式

Windows だけでなく、コンピュータの世界で広く利用されているファイル形式で、主な拡張子は「zip」である。必要に応じて LZ77 や Deflated のような各種ある圧縮アルゴリズムを選択・使用することができるため、圧縮率重視や速度重視など、柔軟な使い分けが可能である。Windows 上での利用については、Windows98 の Plus ！パックで登場して以来、エクスプローラで標準サポートしている。

ZIP 形式にはよく似た名前の形式が多く存在しており、その代表的なものとしては、圧縮アルゴリズムが ZIP 形式と

同じ Deflate を使用している gzip や zlib である。また、7z や bzip2、rzip といったものもあるが、これらは必ずしも ZIP 形式と互換性があるわけではない。

逆に Java で使用される jar ファイルや war ファイルのように、一見すると拡張子が異なり全く違う形式のファイルに見えるものの、中身は ZIP 書庫ファイルと同等といったものも存在する。

Delphi では、以前から zlib 用のライブラリがインストールメディアに添付されていたが、これらを使って ZIP 形式のファイルとして作成するには、ファイルヘッダーやフッターといった部分を自作する必要があり、非常に手間がかかるものであった。しかし、Delphi/400 Version XE3 以降では、System.Zip ユニットが追加され、開発環境をインストールした直後から簡単に圧縮・展開プログラムを作ることが可能になった。

CAB形式

Win32API でサポートされており、Windows Installer や ActiveX の自動ダウンロードで標準に用いられるファイル形式で、主な拡張子は「cab」である。圧縮アルゴリズムとしてはマイクロソフトが独自改良した MSZip か LZX を使用でき、ある程度の速度を保ったまま圧縮率を上げることが可能である。

この形式はマイクロソフトが開発した形式で、構造がソフトウェアの配布に向いているのが特徴でもある。過去（1998 年 11 月）に株式会社インプレスが運営するオンラインソフトウェアを紹介する Web サイト「窓の杜」で行われた 10 種類のファイル圧縮形式によるファイル圧縮対決でも優秀な成績を残している。

参考:LZH形式

圧縮アルゴリズムやファイル仕様など、一式全てが日本人によって開発された純国産のファイル形式で、主な拡張子は「lzh」である。圧縮アルゴリズムは LZSS 法で圧縮したデータをさらにハフマン法を用いて圧縮する LZHUF であり、lh0 から始まり lh7 方式まで公開されているが、開発途中のまま停止している。

当初、ZIP 形式の圧縮ツールが有料だったこともあり、日本国内はもとより海外でも広く使われており、国内における事実上の標準形式にまでなっていた。しかし、アンチウイルスソフトの多くが LZH 形式のファイルに対応しておらず、悪意を持って改竄された LZH 形式のファイルを検疫できない点が 2006 年にベンダーや情報処理推進機構などへ報告されたにもかかわらず、2010 年になってもベンダーの対応が行われず、いっこうに問題が解決されていない点が引き金となり、利用を控えるようにとの呼びかけが起きた。

なお、日本で圧縮ファイルを展開することを「解凍」と呼ぶのは、LZH 形式の標準ユーティリティであった LHA のマニュアルに由来する。

3.ファイル圧縮・展開の具体例（基礎）

まずは基礎ということで、Delphi/400 Version XE3 以降に追加された System.Zip ユニットを使ったファイルを圧縮・展開する例をご紹介する。この例では新設のユニットを利用してコード記述を行うが、難解な記述を必要としない。文章で表現すると、uses 節に System.Zip を追加し、TZipFile クラスを利用するだけである。

TZipFileクラスのクラスメソッドを使った圧縮・展開

一番簡単な方法は、TZipFile クラスが持っている圧縮・展開を行うための専用の命令を使用することである。

ファイルを圧縮する場合には、ZipDirectoryContents メソッドを使用する。このメソッドでは、パラメータに圧縮後のファイル名と圧縮対象となるフォルダパスを指定するだけで処理できる。【コード例 3-①】

ファイルを展開する場合には、ExtractZipFile メソッドを使用する。このメソッドでは、パラメータに展開対象のファイル名と展開先のフォルダ先を指定するだけで処理できる。【コード例 3-②】

どちらのメソッドも非常にシンプルで使用方法も似ているので、簡単に圧縮・展開機能の処理を実装することができる。

TZipFileクラスのオブジェクトを生成しての圧縮・展開

次に、TZipFile クラスからオブジェクトを生成し、操作を行う例を示す。この方法は、異なるフォルダの複数のファイルを 1 つの書庫として圧縮したり、書庫ファイル内の特定のファイルだけ展開したりするといった柔軟な処理が可能である。【コード例 3-③】

TZipFile クラスからオブジェクトを生成して利用する場合、書庫ファイルの内容にアクセスしたり、書庫にファイルを追加したり、展開したりといった作業を細かく制御できることがわかる。

上記のように TZipFile クラスを使用することで、非常に簡単に Zip ファイルを取り扱えるが、このクラスは簡易であるので、例えば暗号化ができなかったり、日本語ファイル名の対応が UTF8 のみであったりと若干使いにくい箇所もある。これらの考慮点に対応するために、オープンソースで開発されているコンポーネントを使用する方法を次章で紹介する。

4.ファイル圧縮・展開の具体例（応用）

基礎編では、Delphi/400 Version XE3 以降で追加された System.Zip ユニットの TZipFile クラスを使用する例を紹介したが、同時に考慮すべきこともあると述べた。記述しなかった考慮点も含めて、以下に整理する。

(1) Delphi XE 以前のバージョンに TZipFile クラスが存在しない
(2) 日本語ファイル名が UTF8 形式でしか対応していない
(3) 暗号化に対応していない（パスワード付 ZIP 書庫に対応していない）
(4) ZIP 以外のファイル形式に対応していない

これらを解決する 1 つの方法として、TurboPower Abbrevia を使う例を紹介する。TurboPower Abbrevia はオープンソースで開発されているコンポーネントで、MPL ライセンスに基づいて提供されており、以下の URL から入手可能である。

TurboPower Abbrevia
http://sourceforge.net/projects/tpabbrevia/

ここでは、2014年8月時点で最新のTurboPower Abbrevia 5.2の使用を前提とする。

上記URLからDownloadリンクを辿って入手したAbbrevia 5.2.zipファイルを展開し、IDEに組み込むと、TAbZipperやTAbUnZipper、TAbZipKitなどのコンポーネントが組み込まれる。画面上に一覧を表示したりするならTAbZipKitを、プログラム内で圧縮や展開するだけならTAbZipperやTAbUnZipperを利用するとよいだろう。

AbbreviaをつかったZIP形式の圧縮・展開

基礎編でTZipFileクラスの専用命令を使用したのと同様の操作を、Abbreviaコンポーネントの TAbZipperとTAbUnZipperを使って行ってみる。

ファイルを圧縮する場合には、TAbZipperコンポーネントのAddFilesメソッドとCloseArchiveメソッドを使用する。このコンポーネントでは、プロパティに圧縮後のファイル名を指定し、AddFilesメソッドで圧縮対象となるフォルダパスを指定するだけで処理できる。【コード例4-①】

ファイルを展開する場合には、TAbUnZipperコンポーネントのExtractFilesメソッドを使用する。このメソッドでは、プロパティに展開対象のファイル名と展開先のフォルダ先を指定するだけで処理することができる。【コード例4-②】コード例4-①、4-②のFileName・BaseDirectoryなどのプロパティはオブジェクトインスペクタで設定可能である。

基礎編のTZipFileクラスのように簡単にZIP書庫を取り扱うことができる上に、このコンポーネントはDelphi 6～Delphi Version XE6までと非常に幅広いバージョンに対応している。また、UTF8形式以外の日本語ファイル名も対応しており、このコンポーネントを利用するだけで、前述した考慮点の(1)と(2)が解決されたことになる。(3)の暗号化も対応しており、その利用方法はTAbZipperやTAbUnZipperのPasswordプロパティをセットするだけという簡単なものになっている。コード例4－①、4－②にパスワード指定を追加した例を示す。【コード例4-①-1、コード例4-②-1】

コード例4-①-1、4-②-1のFileName・Password・BaseDirectoryなどのプロパティは、オブジェクトインスペクタで設定可能である。

残った考慮点は(4)のZIP形式以外の対応となるが、CAB形式であればTAbCabKitやTAbMakeCab、TAbCabExtractorといったコンポーネントも準備されており、圧縮・展開が可能である。

AbbreviaをつかったCAB形式の圧縮・展開

Abbreviaコンポーネントの TAbMakeCabとTAbCabExtractorを使って、CAB形式で圧縮・展開する例を示す。基本的には使用するコンポーネントが異なるだけで、使い方はZipの圧縮・展開と同じようなコーディングで簡単に実装することができる。【コード例4-③、コード例4-④】

コード例4-③、4-④のFileName・BaseDirectoryなどのプロパティはオブジェクトインスペクタで設定可能である。

CAB形式は仕様上、暗号化に対応していないため、パスワードは指定できない。その点を除けばコンポーネントの違いはあるものの、CAB形式の書庫をZIP形式とほぼ同じ手順で取り扱うことができる。

ここまでで Abbreviaのコンポーネントの基礎的な使い方を紹介した。実は、Abbreviaのコンポーネントのうち、Zipper系のコンポーネントは、ZIP/CAB形式以外にもTAR/GZIP/BZIP2などのファイル形式にも対応しており、コンポーネントに渡すFileNameの拡張子を変更するだけで自動判別するような仕組みも持っているので、いろいろと試していただきたい。

この後は、圧縮・展開の意味からは少し外れてしまうが、自身を展開するためのプログラムが付加された実行形式の圧縮ファイルである「自己展開書庫」の作成について紹介する。

AbbreviaではTAbMakeSelfExeコンポーネントを利用することで、ZIP形式の書庫ファイルから自己展開形式のファイルを作成することが可能である。その際、事前に自己展開プログラムを準備しておく必要があるが、Abbreviaを展開した際にexamplesフォルダ内にSelfStub.dprという自己展開プログラム用のプロジェクトが用意されているので、これをコンパイルすると自己展開プログラム用の実行ファイルができ上がるようになっている。

参考:Abbreviaを使った自己展開書庫の作成

Abbreviaコンポーネントの TAbMakeSelfExeを使って、自己展開書庫を作成する例を示す。自己展開書庫の元となるZIP形式の書庫ファイルと、前述したexamplesフォルダ内のSelfStub.dprをコンパイルしてできるSelfStub.exeを事前に準備しておく必要がある点にご注意いただきたい。このコンポーネントも使い方は通常の圧縮・展開と同じなので、作成する自己展開書庫と対象の圧縮ファイルを指定するだけで実装できる。【コード例4-⑤】

コード例4-⑤のStubExe・ZipFileのプロパティはオブジェクトインスペクタで設定可能である。

自己展開プログラムであるSelfStub.exeを工夫することで、自己展開前にバージョン情報や注意事項を画面表示するといったことも可能である。また、出力先フォルダの指定方法などに工夫を凝らすこともできる。しかし、自己展開プログラムのサイズが大きくなると、必然的に自己展開書庫のサイズも大きくなってしまうため、必要最小限にとどめる必要がある。自己展開書庫はユーティリティプログラムではなく、自分自身の展開に特化した単機能プログラムであることを忘れてはならない。

5.まとめ

今回はファイルの圧縮・展開、特にZIP形式に重点を置き、その操作方法を紹介した。先に述べたようにデータの大容量化、モバイル環境へのシフトといった流れは、圧縮・展開といった技術への

コード例3-①　TZipFileクラスのクラスメソッドを使った圧縮例

```
1  procedure TfrmZIPSample.btnZipClick(Sender: TObject);
2  begin
3    // C:¥Tempフォルダの内容をC:¥test.zipに圧縮する
4    TZipFile.ZipDirectoryContents('C:¥test.zip', 'C:¥Temp¥');
5  end;
```

コード例3-②　TZipFileクラスのクラスメソッドを使った展開例

```
1  procedure TfrmZIPSample.btnUnZipClick(Sender: TObject);
2  begin
3    // C:¥test.zipの内容をC:¥Tempフォルダに展開する
4    TZipFile.ExtractZipFile('C:¥test.zip', 'C:¥Temp¥');
5  end;
```

コード例3-③　TZipFileクラスを利用する例

```
1  procedure TfrmZIPSample.btnZipClassClick(Sender: TObject);
2  var
3    zip: TZipFile;
4    i: Integer;
5  begin
6    zip := TZipFile.Create;
7    try
8      // C:¥test.zipを読み書き可能な形で開く
9      zip.Open('C:¥test.zip', zmReadWrite);
10
11     // 書庫内のファイル一覧をメモコンポーネントに列挙する
12     for i := 0 to zip.FileCount - 1 do
13       Memo1.Lines.Add(zip.FileName[i]);
14
15     // C:¥Temp¥aaa.txtを書庫に追加する
16     zip.Add('C:¥Temp¥aaa.txt');
17
18     // 書庫からbbb.txtを展開する
19     zip.Extract('bbb.txt');
20
21     // 書庫を閉じる
22     zip.Close;
23   finally
24     zip.Free;
25   end;
26 end;
```

重要度をますます強めていくことであろう。

また、通常のZIP書庫ではなく暗号化ZIP書庫を使うことで、単純にデータ容量を小さくする以外に、セキュリティを高めることも可能である点にも注目したい。

例えば、メールの送信プログラムで添付するファイルを暗号化ZIP書庫にして、メール送信後に同一宛先に対して展開パスワードを記載したメールを自動送信するようにすれば、ユーザーに手間をかけさせることなくセキュリティを高めることが可能であろう。この機会にぜひファイルの圧縮・展開といった操作を試していただければ幸いである。

なお、本レポートを作成するにあたり、zlibライブラリ、統合アーカイバコンポーネント、ZipMaster、ZipForgeといったソフトウェアを確認したが、それぞれに考慮を必要とする点があったため、参考までに以下に記しておく。

・zlibライブラリ（Delphiのメディアに同梱または同時インストール）

TZipFileクラス同様、新しいバージョンのDelphiでは標準でSystem.ZLibユニットとして組み込まれるようになっているため、追加インストールは不要である。また、古いバージョンのDelphiでは、インストールメディア内にライブラリ一式が同梱されており、それを使用することも、インターネットから最新ファイルを入手することも可能である。

圧縮アルゴリズムはZIP形式と互換性があるものの、ファイルヘッダーやフッターといった項目を自作する必要があり、これを使ってZIP書庫を作成したり、一般的なZIP書庫を展開したりするという目的に利用するのは難しい。しかし、ヘッダーやフッターのない専用形式として取り扱うなら、圧縮・展開機能は十分に使用可能である。

・統合アーカイバコンポーネント

参考：http://www.geocities.jp/norg1964/cmarc/

NIFTYサーブにて進められていた「統合アーカイバAPI仕様」に準拠した各種ファイル仕様向けのライブラリを利用するためのコンポーネントである。残念なことに開発が停止しており、対応しているDelphiのバージョンは2～7までである。他のバージョンで利用する場合は、一部修正が必要である。

・ZipMaster

参考：http://www.delphizip.org/

同梱されているDLLを使ってZIP書庫を作成する一風変わったコンポーネントである。開発も継続されており、XE以前の対応やXE3以降の64bit対応なども行われている。インストールが若干面倒な点と、DLLが必要となる点に注意。

・ZipForge

参考：http://www.componentace.com/zip_component_zip_delphi_zipforge.htm

ZIP形式の仕様をほぼ完全に準拠した高機能なコンポーネントである。公共向けや企業での利用は有償のコマーシャル版を利用する必要がある。

M

コード例4-①　TAbZipperを使った圧縮例

```
1  procedure TfrmAbZipSample.btnZipClick(Sender: TObject);
2  begin
3    // C:¥Tempフォルダの内容をC:¥test.zipに圧縮する
4    AbZipper1.FileName := 'C:¥test.zip';
5    AbZipper1.AddFiles('C:¥Temp¥*.*', 0);
6    AbZipper1.CloseArchive;
7  end;
```

コード例4-②　TAbUnZipperを使った展開例

```
1  procedure TfrmAbZipSample.btnUnZipClick(Sender: TObject);
2  begin
3    // C:¥test.zipの内容をC:¥Tempフォルダに展開する
4    AbUnZipper1.FileName := 'C:¥test.zip';
5    AbUnZipper1.BaseDirectory := 'C:¥Temp';
6    AbUnZipper1.ExtractFiles('*.*');
7  end;
```

コード例4-①-1　TAbZipperでパスワード付圧縮する例

```
1  procedure TfrmAbZipSample.btnZipClick(Sender: TObject);
2  begin
3    // C:¥Tempフォルダの内容をC:¥test.zipにパスワード付で圧縮する
4    AbZipper1.FileName := 'C:¥test.zip';
5    AbZipper1.Password := 'Password';
6    AbZipper1.AddFiles('C:¥Temp¥*.*', 0);
7    AbZipper1.CloseArchive;
8  end;
```

コード例4-②-1　TAbUnZipperでパスワードを指定して展開する例

```
1  procedure TfrmAbZipSample.btnUnZipClick(Sender: TObject);
2  begin
3    // C:¥test.zipの内容にパスワードを指定してC:¥Tempフォルダに展開する
4    AbUnZipper1.FileName := 'C:¥test.zip';
5    AbUnZipper1.Password := 'Password';
6    AbUnZipper1.BaseDirectory := 'C:¥Temp';
7    AbUnZipper1.ExtractFiles('*.*');
8  end;
```

コード例4-③　TAbMakeCabを使った圧縮例

```
1  procedure TfrmAbCabSample.btnMakeCabClick(Sender: TObject);
2  begin
3    // C:¥Tempフォルダの内容をC:¥test.cabに圧縮する
4    AbMakeCab1.FileName := 'C:¥test.zip';
5    AbMakeCab1.AddFiles('C:¥Temp¥*.*', 0);
6    AbMakeCab1.CloseArchive;
7  end;
```

コード例4-④　TAbCabExtractorを使った展開例

```
1  procedure TfrmAbCabSample.btnExtCabClick(Sender: TObject);
2  begin
3    // C:¥test.cabの内容をC:¥Tempフォルダに展開する
4    AbCabExtractor1.FileName := 'C:¥test.cab';
5    AbCabExtractor1.BaseDirectory := 'C:¥Temp';
6    AbCabExtractor1.ExtractFiles('*.*');
7  end;
```

コード例4-⑤　TAbMakeSelfExeを使った自己展開書庫の作成例

```
1  procedure TfrmAbSelfExeSample.btnMakeSelfExeClick(Sender: TObject);
2  begin
3    // C:¥SelfStub.exeをC:¥test.zipに付加して自己展開書庫を作成する
4    AbMakeSelfExe1.StubExe := 'C:¥SelfStub.exe';
5    AbMakeSelfExe1.ZipFile := 'C:¥test.zip';
6    AbMakeSelfExe1.Execute;
7  end;
```

前坂 誠二

株式会社ミガロ.

システム事業部

[Delphi/400] FastReportを使用した帳票作成テクニック —FastReport応用

略歴
1989年3月21日生
2011年3月 関西大学 文学部卒
2011年4月 株式会社ミガロ. 入社
2011年4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

FastReportは、Delphi/400 Version XE3で新たにバンドルされた帳票作成ツールである。FastReportを用いると、誰でも容易に帳票出力プログラムを作成できる。基本的な帳票作成の方法については『Migaro.Technical Report 2013』の「FastReportを使用した帳票作成入門」に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。本稿は、FastReportを使用し、グラフや画像ファイルを用いた帳票出力の方法やFastReportの実践的なテクニックについて紹介する。

本稿ではまず、第2章でレポートウィザードを使用したデータベース帳票の作成方法を紹介する。次に、第3章・第4章では、第2章で作成した帳票デザインを基に、グラフや画像ファイルを挿入する方法を紹介する。第5章では、FastReportで帳票出力プログラムを作成する際に活用できる帳票作成プログラムの実践的なテクニックを紹介する。

なお、本稿で使用しているプログラム例はDelphi/400 Version XE3を使用し、FastReportはバンドル版を使用している。また、FastReportのバージョンは4.12.13である。

2.レポートウィザードを使用したデザイン作成

FastReportを使用し、データベース帳票を新規作成する場合、一般的な方法としては、デザイン画面上にバンドオブジェクトを配置し、TfrxMemoViewコンポーネントなどのコンポーネントを1つずつ貼り付けて、項目設定を行っていく。もちろんこの方法でレポートデザインは作成できるが、項目ごとに1つずつコンポーネントを貼り付ける作業を手間に感じることがある。今回紹介するレポートウィザードを使用した作成方法はそういった手間を省くことができ、非常に簡単な手順でデータベース帳票の基盤が作成できる。また、用紙レイアウトや出力させたい項目も手順にそって設定できるので、FastReportを初めて使用する方でも、悩むことなく、デザインの作成が可能である。

今回は帳票の作成例として、【図1】のような“売上一覧表”を作成する。この例では、営業所名ごとにグループ化を行い、グループごとの小計、そして最終レコードのあとに合計を表示させる。

レポート作成の準備

まず、作成の準備として、印刷フォーム画面にTfrxReportコンポーネントを貼り付ける。今回出力する帳票はデータベースから取得した値を出力するため、TfrxDBDataSetコンポーネントも併せて貼り付け、DataSetプロパティに出力したいデータセットを紐づけておく。

レポートウィザードの使用

レポートウィザードを使用するには、まず、Delphiの開発画面で貼り付けたfrxReport1をダブルクリックし、レポートデザイナ画面を起動させる。レポートデザイナ画面が起動した後、「File」

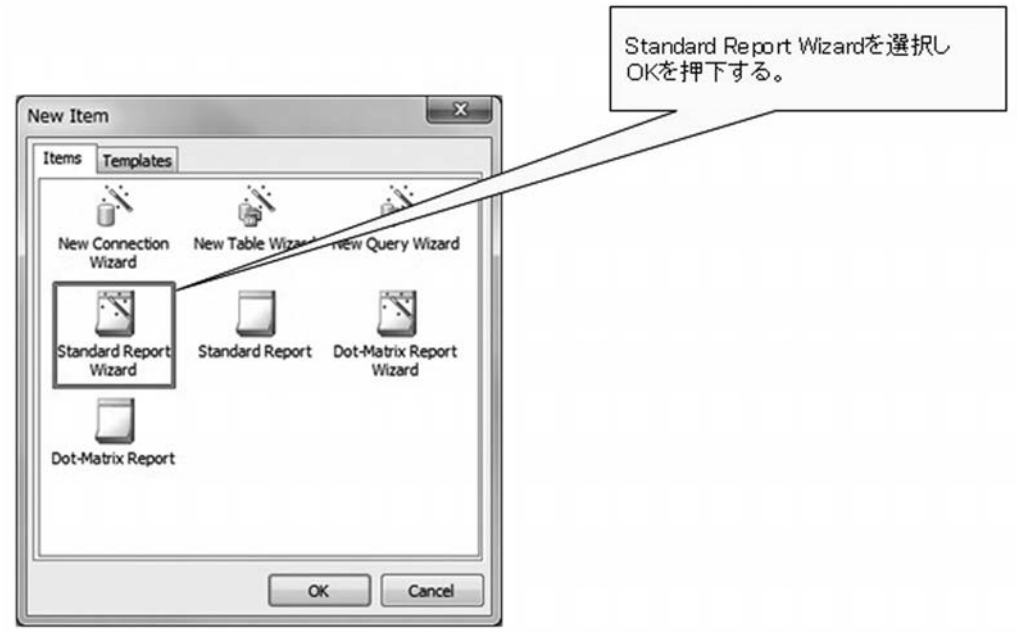
図1

売上一覧表(2014年)

名称	住所1	住所2	電話番号	FAX番号	金額
東京営業所					
株式会社足立商店	東京都足立区	1−1−2	XXX-XXXX-XXX	XXX-XXXX-XXX	1,000
株式会社足立興業	東京都足立区	1−2−3	XXX-XXXX-XXX	XXX-XXXX-XXX	2,000
株式会社荒川商店	東京都荒川区	2−2−3	XXX-XXXX-XXX	XXX-XXXX-XXX	3,000
株式会社荒川工業	東京都荒川区	1−2−1	XXX-XXXX-XXX	XXX-XXXX-XXX	5,000
株式会社板橋商店	東京都板橋区	5−2−3	XXX-XXXX-XXX	XXX-XXXX-XXX	18,000
				小計:	29,000
大阪営業所					
株式会社池田商店	大阪府池田市	1−2−3	XXX-XXXX-XXX	XXX-XXXX-XXX	1,000
株式会社泉商店	大阪府泉大津市	12−2−3	XXX-XXXX-XXX	XXX-XXXX-XXX	2,000
株式会社泉興業	大阪府泉大津市	13−2−3	XXX-XXXX-XXX	XXX-XXXX-XXX	3,000
株式会社佐野商店	大阪府泉佐野市	13−2−3	XXX-XXXX-XXX	XXX-XXXX-XXX	8,000
株式会社泉佐野商店	大阪府泉佐野市	13−2−3	XXX-XXXX-XXX	XXX-XXXX-XXX	15,000
				小計:	29,000
青森営業所					
株式会社津軽商店	青森県青森市	1−2−2	XXX-XXXX-XXX	XXX-XXXX-XXX	100
株式会社北青軽商店	青森県北青森市	1−3−2	XXX-XXXX-XXX	XXX-XXXX-XXX	400
				小計:	500
広島営業所					
株式会社安芸商店	広島県安芸市	2−1−2	XXX-XXXX-XXX	XXX-XXXX-XXX	900
株式会社江田島商店	広島県江田島市	1−4−2	XXX-XXXX-XXX	XXX-XXXX-XXX	800
				小計:	1,700
高知営業所					
株式会社香美商店	高知県香美市	3−1−5	XXX-XXXX-XXX	XXX-XXXX-XXX	2,910
株式会社四万十商店	高知県四万十市	1−1−5	XXX-XXXX-XXX	XXX-XXXX-XXX	10,000
				小計:	12,910
				合計:	73,110

2014/08/20 15:33:38 Page 1

図2



New」を選択する。すると、【図 2】の画面が表示されるので、Standard Report Wizard を選択する。次に以下の Step1 ～ Step5 で帳票の出力設定を選択する。

Step1：使用したいデータセットを選択する。【図 3】
Step2：使用したいデータセットの項目を選択する。【図 4】
Step3：グループ化したい項目を選択する（任意）。【図 5】
Step4：用紙の向き、項目の配置を選択する。【図 6】
Step5：レポートのスタイルを指定する。【図 7】

Step1 ～ Step5 で出力設定を選択後、最後に Finish ボタンを押下する。あとは、レポートタイトルや明細タイトルのキャプションを設定すれば、簡単に帳票デザインの基盤が完成する。【図 8】

データのグループ化について

今回は、営業所名でグループ化した帳票の出力を行う。グループ化を行うには、コンポーネントパレットから Insert Band をクリックし、Group Header Band を選択する。【図 9】

すると、ダイアログ画面が表示されるので Data field または Expression 欄に、グループ化したい項目を設定する。【図 10】

なお、本稿で、このグループ化設定は、レポートウィザード（【図 5】で記載）にて行っている。

Group Header Band を使用すると、データをグループ化するだけでなく、ドリルダウン設定も行うことができる。ドリルダウン設定を行うと、プレビュー画面でグループごとにデータの表示・非表示を切り替えることができる。【図 11】【図 12】

また、このプレビュー画面で行った表示・非表示の操作は印刷時にも影響がある。つまり、ドリルダウン設定を行った状態で、印刷を実行すると、現在プレビュー画面で表示している内容がそのまま印刷されるということである。

ドリルダウン設定を行うには、レポートデザイナ画面の Group Header Band を選択し、DrillDown プロパティを True に変更するだけで設定が可能である。また、ExpandDrillDown プロパティを True にすると、初期表示時に全てのグループが展開された状態で表示する。グループのデータが非表示の状態で、グループの小計など Group Footer Band に配置した項目のみ表示させたい場合は、ShowFooterIfDrillDown プロパティを True にする。【図 13】

合計の表示

データベースの値を使用した合計値を表示するには、TfrxMemoView コンポーネントを使用し、テキストに“[SUM(<データセット名.フィールド名>)]”と記述する。また、計算範囲については、コンポーネントの配置場所によって決定する。例えば、ページごとの合計値を表示させたい場合は、Page Footer Band に配置することでページごとの合計値を表示させることができる。全レコードの合計値を表示させたい場合は Footer Band に配置することで表示させることができる。

ただし、このような計算項目をレポートデザイナ画面で使用し、さらに Group Header Band のドリルダウン機能も併せて利用している場合は、注意が必要である。レポートデザイナ画面の計算項目は、表示しているデータのみを計算結果に含めるので、ドリルダウン機能でデータを非表示にしている場合、意図していない計算結果になる可能性がある。そのため、ドリルダウン機能を併せて利用する場合は、Delphi ソース内で TClientDataSet の内部計算項目などを使用して、計算処理を行い、その計算値をセットさせるといった工夫が必要となる。【図 14】

3.グラフを用いた帳票の作成

本章では、グラフを用いたデータベース帳票の作成手法について説明する。グラフを使用すると、視覚的に出力内容が表現され、文字データばかりの帳票よりも出力内容が把握しやすくなる。FastReport では、グラフを用いた帳票出力プログラムも通常の帳票出力同様、容易に作成することができる。では、その作成方法について紹介しよう。

新規ページの作成

本章では、例として第 2 章で作成した帳票の最終ページに【図 15】のようなグラフを使用した帳票出力を行う。まず準備として、レポートデザイナ画面の「File ¦ NewPage」を選択する。すると、Report Tree に Page2 という新しいページが作成される。今回は、この Page2 にグラフの出力を行う。【図 16】

グラフの出力

グラフを出力するには、TfrxChartView コンポーネントを使用する。レポートデザイナ画面のコンポーネントパレットから ChartObject を選択し、TfrxChartView コンポーネントを画面に貼り付けると、Chart Editor が表示される。【図 17】

TfrxChartView コンポーネントは、Chart Editor でグラフの種類選択や詳細設定を行う。Add Series ボタンを押すと、グラフの種類を選択するダイアログが表示されるので、ここで使用したいグラフの選択を行う。今回は、棒グラフを使用するため、Bar から Normal を選択する。【図 18】

グラフの種類を選択すると、Chart Editor に先ほど追加したグラフがツリー形式で表示される。今回出力するグラフは、全営業所での顧客の売上 Top5 を降順に表示させる。

まずは、グラフのタイトル設定を行うため、Chart を選択し、プロパティの Title を選択する。そして、Text をダブルクリックすると入力画面が表示されるので、そこでタイトル名を設定する。【図 19】

次にグラフの設定を行う。Bar をクリックし、DataSet には、対象である frxDBDataSet1 を選択する。そして、Values の Label には顧客名の項目、Y には、売上金額の項目を選択する。Other Options では、並び順や上位何件まで表示させるかが設定できる。今回は Sort order を、Descending に設定し、TopN Values には 6 を設定する。【図 20】

なお、TopN Values を 6 と設定すると、6 件目のデータには、上位 5 件を除いた項目の合計値が表示され、TopN caption の値が名称として設定される（【図 15】参照）。

図3

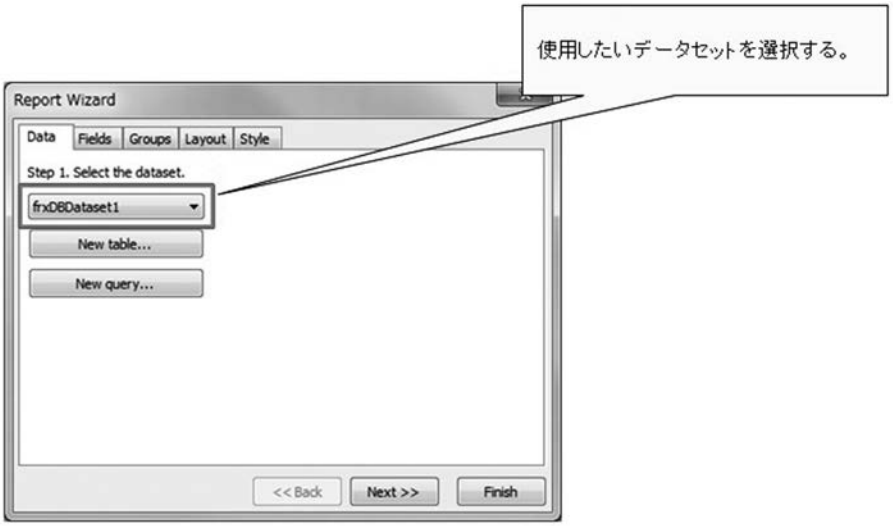

使用したいデータセットを選択する。
Report Wizard
Data
Fields
Groups
Layout
Style
Step 1. Select the dataset.
frxDBDataset1
New table...
New query...
<< Back
Next >>
Finish

図4

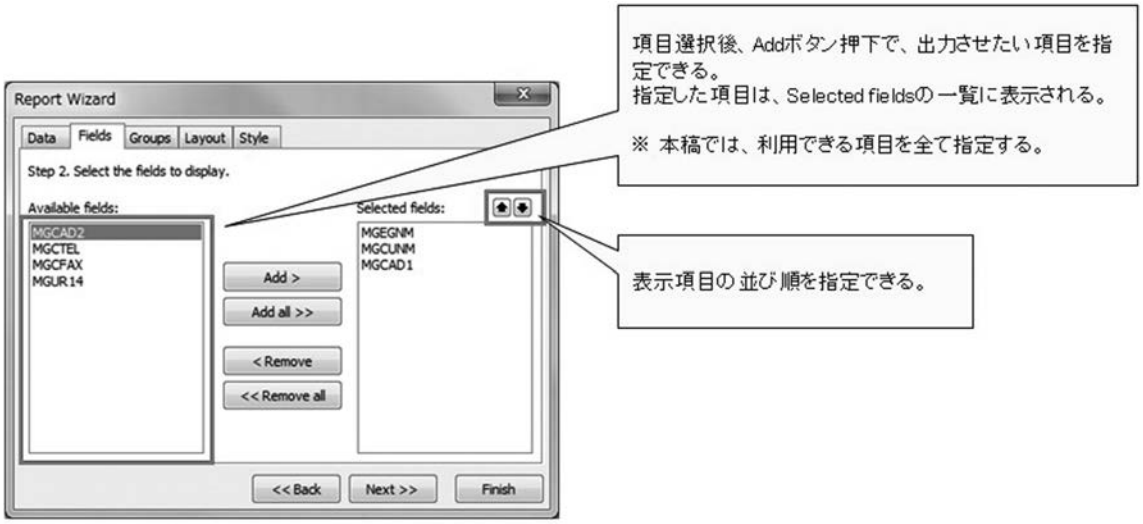
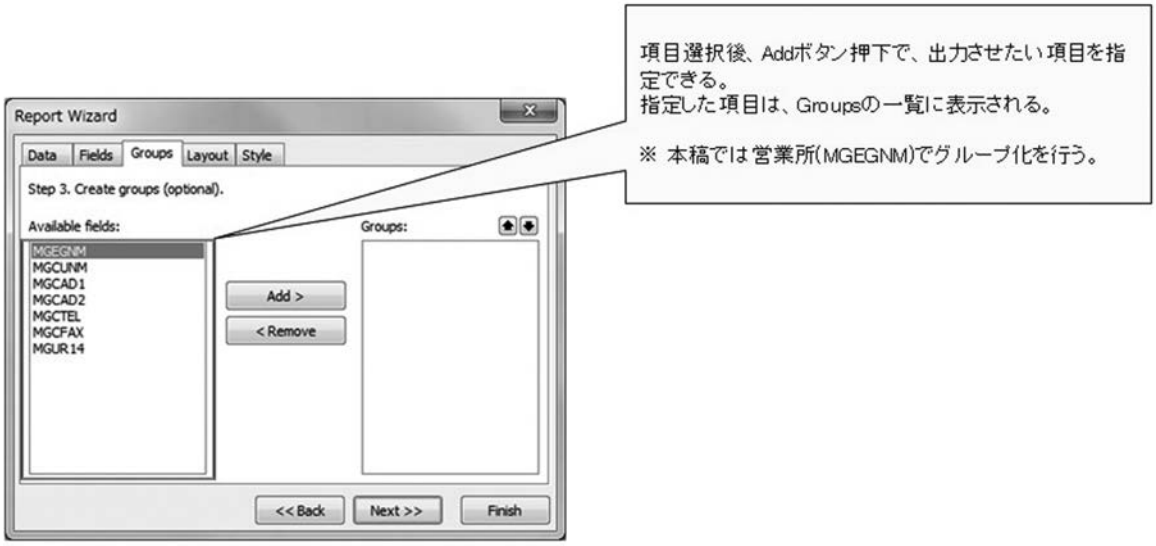

項目選択後、Addボタン押下で、出力させたい項目を指定できる。
指定した項目は、Selected fieldsの一覧に表示される。
※ 本稿では、利用できる項目を全て指定する。
表示項目の並び順を指定できる。
Report Wizard
Data
Fields
Groups
Layout
Style
Step 2. Select the fields to display.
Available fields:
MGCAD2
MGCTEL
MGCFAX
MGUR14
Add >
Add all >>
< Remove
<< Remove all
Selected fields:
MGEGNM
MGCUNM
MGCAD1
<< Back
Next >>
Finish

図5


項目選択後、Addボタン押下で、出力させたい項目を指定できる。
指定した項目は、Groupsの一覧に表示される。
※ 本稿では営業所(MGEGNM)でグループ化を行う。
Report Wizard
Data
Fields
Groups
Layout
Style
Step 3. Create groups (optional).
Available fields:
MGEGNM
MGCUNM
MGCAD1
MGCAD2
MGCTEL
MGCFAX
MGUR14
Add >
< Remove
Groups:
<< Back
Next >>
Finish

以上でグラフを出力するための設定は完了である。あとは、Delphi からプレビュー処理を呼び出すだけであるが、グラフを含む帳票を出力する場合は、必ず Delphi ソースの Uses に frxChart を追加しておく必要がある。

4.画像ファイルを用いた帳票の作成

本章では、画像ファイルを用いた帳票の作成手順について紹介する。画像ファイルの出力もグラフの出力と同様に簡単な手順で行うことができる。今回の例では、ページのフッターごとに会社のロゴを出力するプログラムを作成する。【図 21】

まず、レポートデザイナ画面で、TfrxPictureView コンポーネントを Page Footer Band に貼り付ける。【図 22】

次に、Delphi にてプレビュー処理の前に、次のロジックを記述する。変数 frxPic を TfrxPictureView で宣言し、先ほどデザイン画面で貼り付けた TfrxPictureView コンポーネントを TfrxReport の FindObject 関数で探し出し、セットする。この際、FindObject 関数の引数には、デザイン画面で貼り付けたコンポーネントの Name をセットする点がポイントである。次に、LoadFromFile 関数を使用し、出力したい画像データのパスを引数にセットする。最後に、プレビュー処理を実行すれば、画像データを使用した帳票出力の完成である。【ソース 1】

なお、今回は画像ファイルに JPEG ファイルを用いたが、他にも BMP ファイルや PNG ファイルなども使用可能である。

5.帳票作成プログラムの応用テクニック

本章では、FastReport で帳票を作成する際に利用できるテクニックを紹介する。これらの内容を活用すれば、FastReport を使用した帳票作成プログラムの幅がより広がる。

配列で保持しているレコードデータの出力

FastReport では、データベースで取得したレコードのデータだけでなく、Delphi プログラム上で内部保持している配列データも、Master Data Band を使用して出力することが可能である。【図 23】

Master Data Band を使用するため、Delphi のロジックで for 文などの繰り返し処理を記述せずに非常に少ないロジックで配列データの出力を行える。

まずは Delphi の開発画面で TfrxUserDataSet コンポーネントを配置し、UserName プロパティにレポートデザイナ画面で表示させたいデータセット名、Fields プロパティに出力させたい項目を設定する。【図 24】

次に、レポートデザイナ画面の設定を行う。「Report | Data」から Select Report Datasets ダイアログを開き、先ほど Delphi の開発画面の UserName プロパティで設定したデータセット名（本稿では frxUserDataSet1 と設定）にチェックを入れ、OK を押下する。【図 25】

すると、Data Tree に frxUserDataSet1 が新しく追加される。この後の手順は、TfrxDBDataSet コンポーネントを使用するときと同様に、Master Data Band を配置し、出力したい項目をその上に配置する。以上で、レポートデザイナ画面の設定は完了である。

ただし、TfrxUserDataSet の場合は、TfrxDBDataSet を使用する場合と違い、レポートウィザードでの作成ができないため、その点は注意が必要である。

最後に Delphi でデータ出力処理のロジックを記述する。データベースから値を出力する際は、ShowReport と記述するだけで、帳票出力が可能であったが、配列データを出力する際は、他にもロジックの記述が必要になる。使用するイベントは TfrxUserDataSet コンポーネントの OnCheckEOF イベントと OnGetValue イベントである。OnCheckEOF イベントでは、どのタイミングで処理を終了するかを記述し、OnGetValue イベントでは、frxUserDataSet コンポーネントの Fields プロパティで定義した変数に、どの値をセットするかを記述する。あとは、データ取得処理のあとに、帳票出力処理を呼び出せば、完成である。【ソース 2】

1行ごとに背景色を変更する方法

【図 23】のように、連続した明細データが出力された時、データを見づらく感じたことはないだろうか。そういった場合によく用いられるのが、1 行ごとにレコードの背景色を分け、データを見やすくするという手法である。もちろん FastReport でもその手法を用いることができるのだが、少し工夫が必要となる。

まず、レポートデザイナ画面で TfrxMemoView コンポーネントを選択し、Master Data Band 上に配置する。配置が終わるとダイアログ画面が表示されるので、Highlight タブを選択し、条件式に「<Line#> Mod 2」と記述する。

次に、Align プロパティで baClient を選択し、Master Data Band 全体に配置されるように設定する。最後に、右クリックより Send to Back（最背面へ移動）を選択し、出力項目の背面に配置されるように設定する。【図 26】

あとは、プレビュー処理を呼び出し、出力内容を確認すると、【図 27】のように 1 行ごとに背景色が設定された帳票が出力される。

帳票レイアウトの分離

FastReport は、帳票レイアウトを実行モジュール（Exe ファイル）内に組み込んで出力する方法だけでなく、帳票レイアウトを 1 つのレイアウトファイルとして作成し、実行モジュールから参照することも可能である。メリットとしては、項目の配置移動などレイアウトの修正のみを行った場合に、実行モジュールの再コンパイルが不要となる点である。

ではまず、実行モジュールからレイアウトファイルを作成する方法であるが、レポートデザイナ画面を起動し、名前を付けて保存を行う。すると、帳票レイアウトが .fr3 という拡張子のファイルで保存される。

次に、作成したレイアウトファイルを読み込むには、プレビュー処理の実行前に、LoadFromFile 関数を呼び出し、引数に先ほど保存したレイアウトファイルのパスを指定する。レイアウトファイルの参照は、このわずか 1 行のロジックを追加するだけで可能となる。【ソース 3】

ただし、引数に指定するパスの指定を間違えると、帳票が正しく出力されない

図6

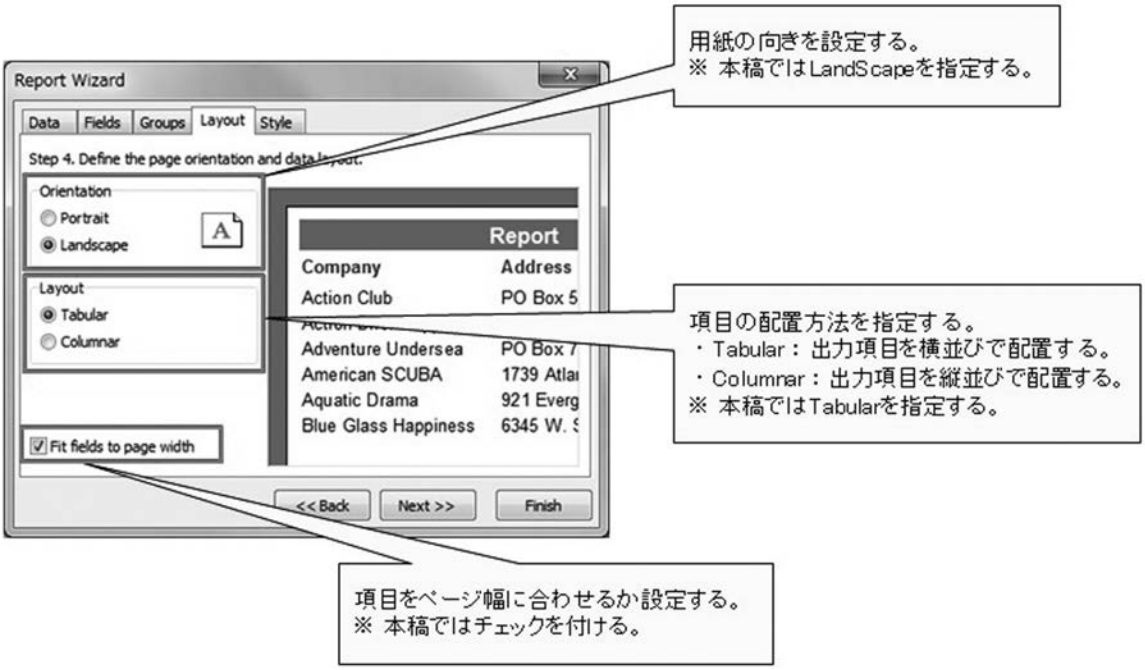


図7

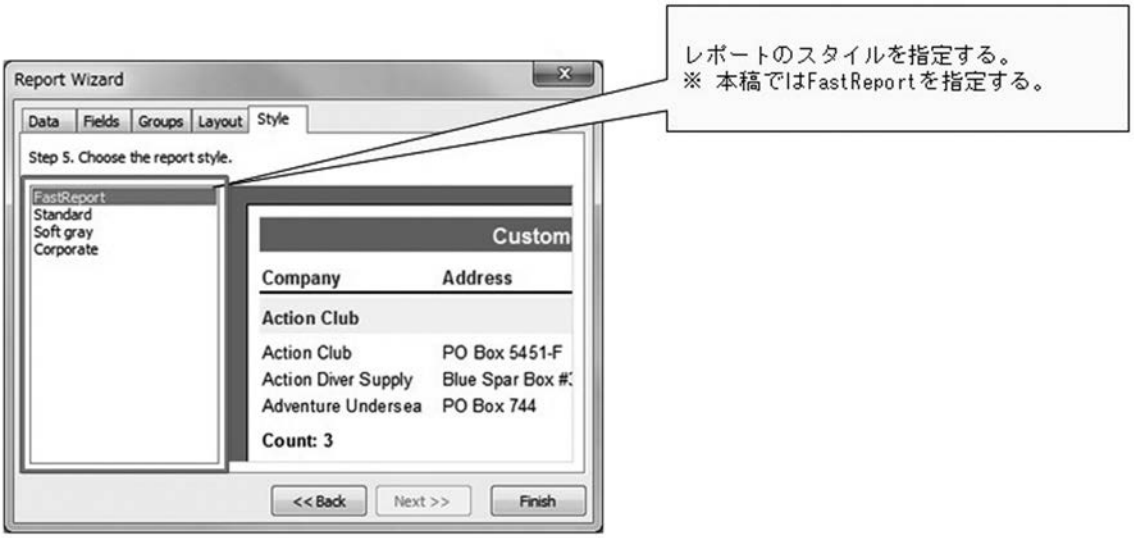
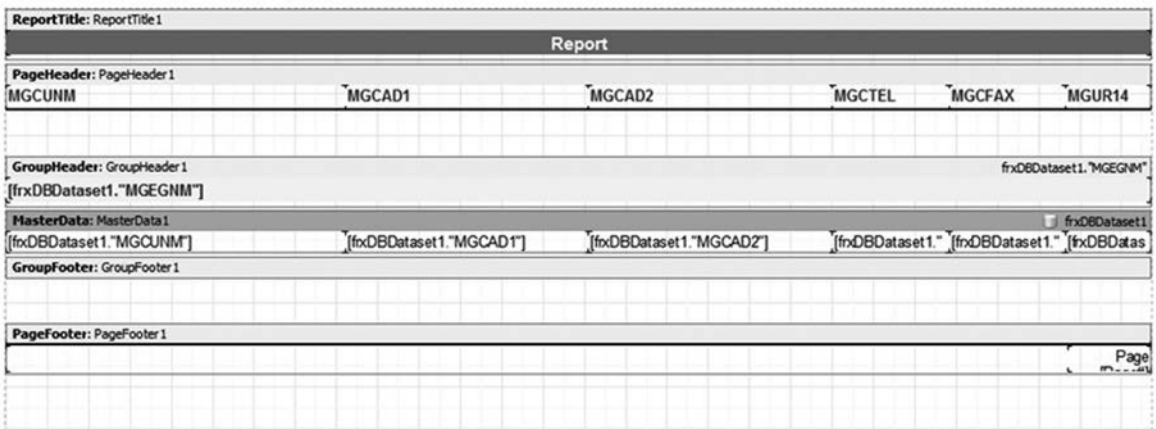


図8



レポートタイトル名や明細タイトル名は
手動で設定する。

ので注意が必要である。

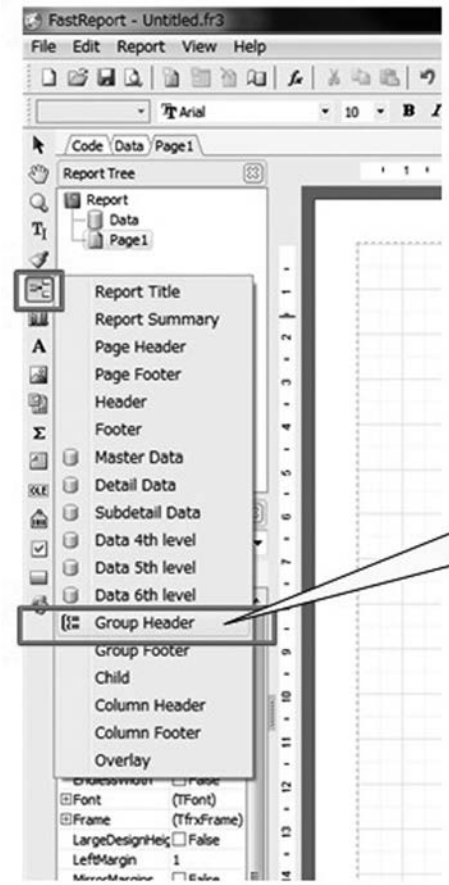
6.最後に

今回は、Delphi/400 の帳票ツールの 1 つである FastReport の帳票作成テクニックを紹介した。FastReport はレポートウィザードの使用で、コンポーネントを貼り付ける手間もなく、容易にデザインの基盤が作成できる。また、グラフや画像ファイルの挿入も FastReport のレポートデザイナ画面でのビジュアル設計が可能であるため、Delphi でのソースをほとんど記述することなく作成できることを本稿でお分かりいただけたであろう。よって、FastReport を利用すれば、誰でも容易に帳票出力を利用したアプリケーション作成が可能である。

FastReport のバンドル版は Delphi/400 Version XE3 以降に付属しており、すぐに使用できるので、今後はさらに利用するユーザーが増えていくと思われる。その際に、本稿の内容をご活用いただければ幸いである。

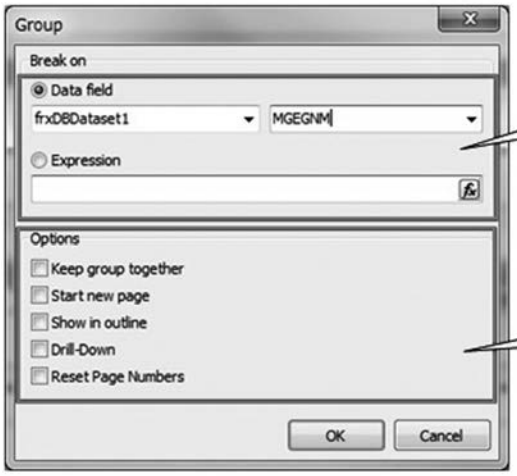
M

図9



コンポーネントパレットのInsert Bandから
Group Header Bandを選択する。

図10



グループ化したい項目を指定する。
Expresssionで、関数等の指定も行える。

Optionsの設定は、グループ決定後に
プロパティからでも設定可能である。

図11

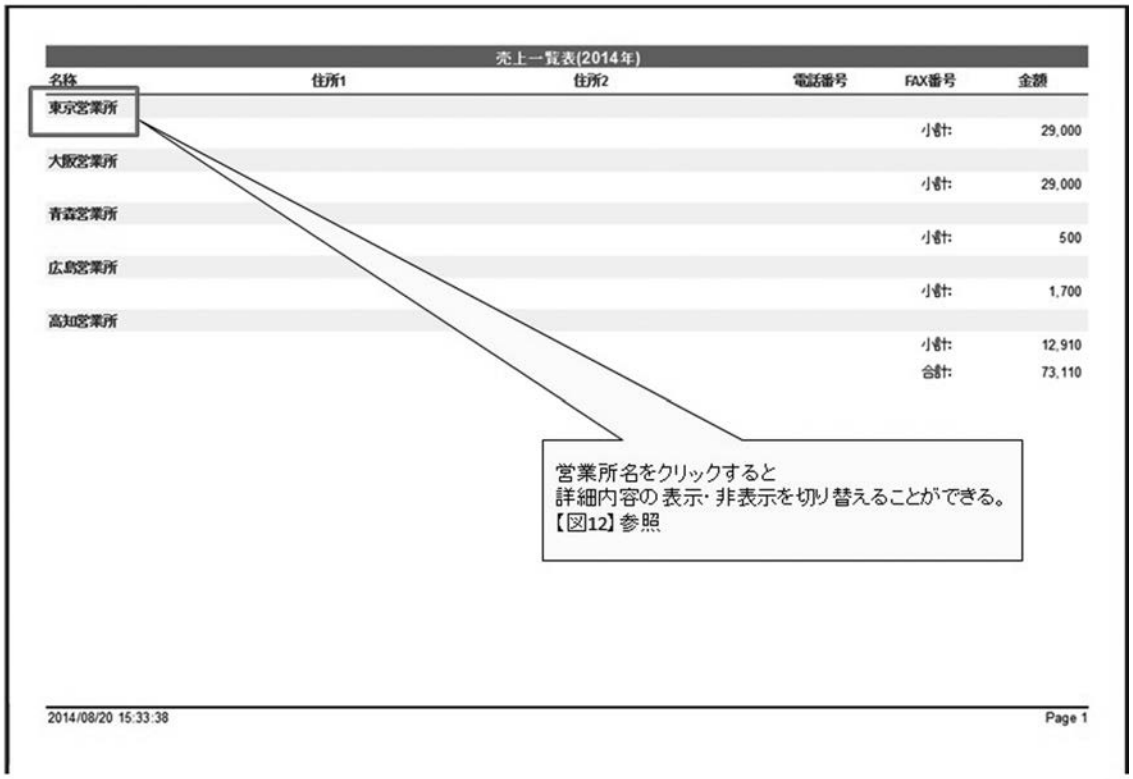

売上一覧表(2014年)
名称
住所1
住所2
電話番号
FAX番号
金額
東京営業所
小計: 29,000
大阪営業所
小計: 29,000
青森営業所
小計: 500
広島営業所
小計: 1,700
高知営業所
小計: 12,910
合計: 73,110
営業所名をクリックすると
詳細内容の表示・非表示を切り替えることができる。
【図12】参照
2014/08/20 15:33:38
Page 1

図12

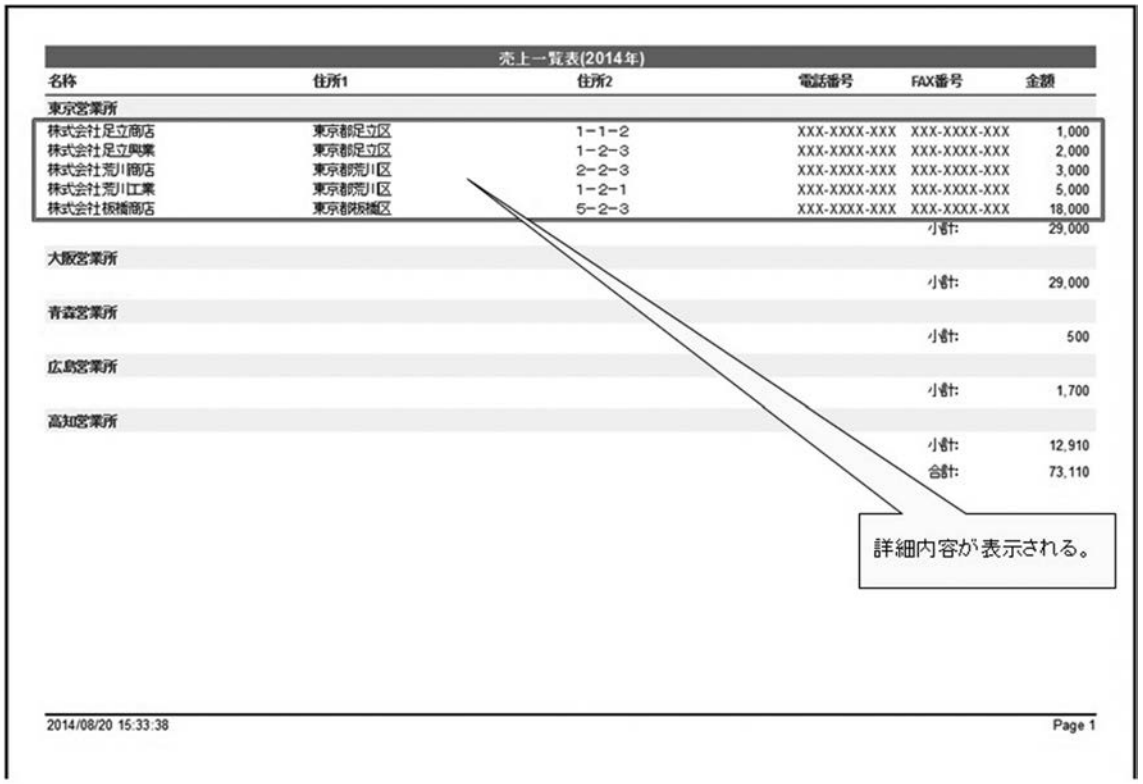
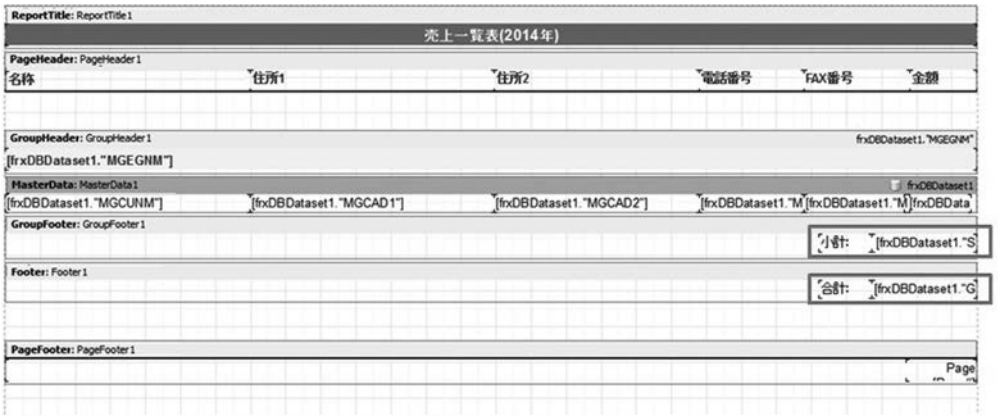

売上一覧表(2014年)
名称
住所1
住所2
電話番号
FAX番号
金額
東京営業所
株式会社足立商店 東京都足立区 1−1−2 XXX-XXXX-XXX XXX-XXXX-XXX 1,000
株式会社足立興業 東京都足立区 1−2−3 XXX-XXXX-XXX XXX-XXXX-XXX 2,000
株式会社荒川商店 東京都荒川区 2−2−3 XXX-XXXX-XXX XXX-XXXX-XXX 3,000
株式会社荒川工業 東京都荒川区 1−2−1 XXX-XXXX-XXX XXX-XXXX-XXX 5,000
株式会社板橋商店 東京都板橋区 5−2−3 XXX-XXXX-XXX XXX-XXXX-XXX 18,000
小計: 29,000
大阪営業所
小計: 29,000
青森営業所
小計: 500
広島営業所
小計: 1,700
高知営業所
小計: 12,910
合計: 73,110
詳細内容が表示される。
2014/08/20 15:33:38
Page 1

図13

図14



GroupFooterに小計を配置し、Footerに総合計を配置する。
DrillDown機能を使用時は、計算項目を使用せず、Delphiで計算処理を行い、セットする。

図15

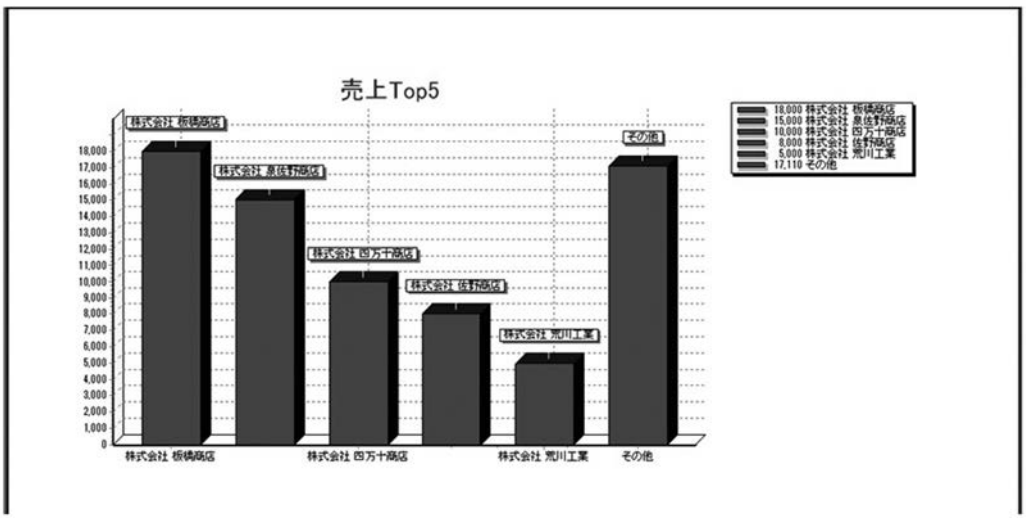


図16

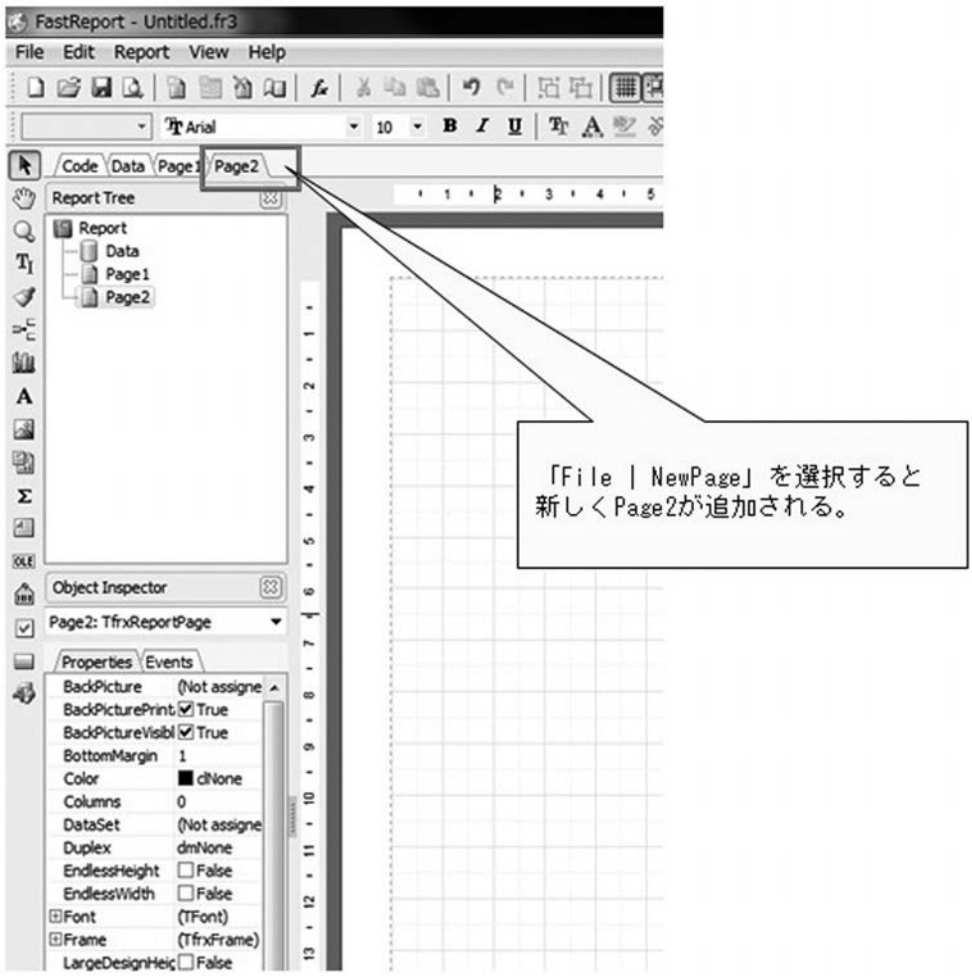

FastReport - Untitled.fr3
File Edit Report View Help
Code Data Page1 Page2
Report Tree
Report
Data
Page1
Page2
Object Inspector
Page2: TfrxReportPage
Properties Events
「File | NewPage」を選択すると
新しくPage2が追加される。

図17

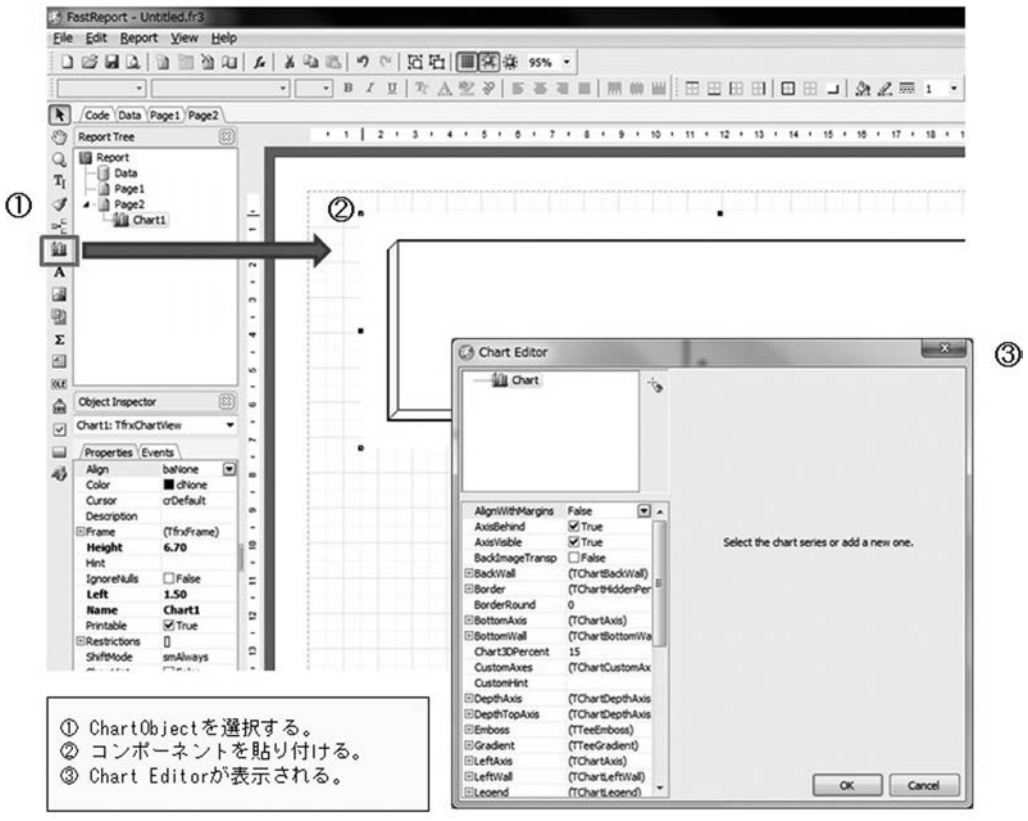

①
②
③
Chart Editor
Chart
Select the chart series or add a new one.
OK
Cancel
① ChartObjectを選択する。
② コンポーネントを貼り付ける。
③ Chart Editorが表示される。

図18

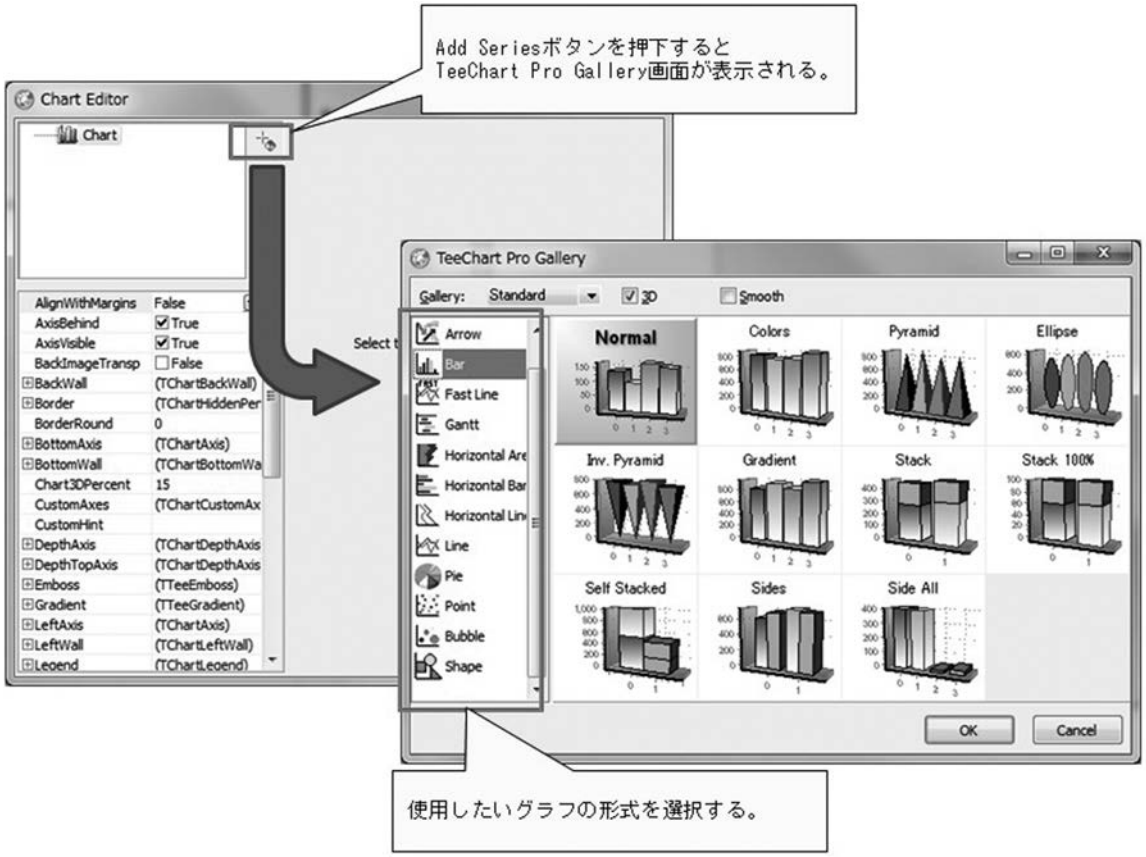

Add Seriesボタンを押下すると
TeeChart Pro Gallery画面が表示される。
Chart Editor
Chart
TeeChart Pro Gallery
Gallery: Standard
3D
Smooth
Arrow
Bar
Fast Line
Gantt
Horizontal Area
Horizontal Bar
Horizontal Line
Line
Pie
Point
Bubble
Shape
Normal
Colors
Pyramid
Ellipse
Inv. Pyramid
Gradient
Stack
Stack 100%
Self Stacked
Sides
Side All
OK
Cancel
使用したいグラフの形式を選択する。

図19

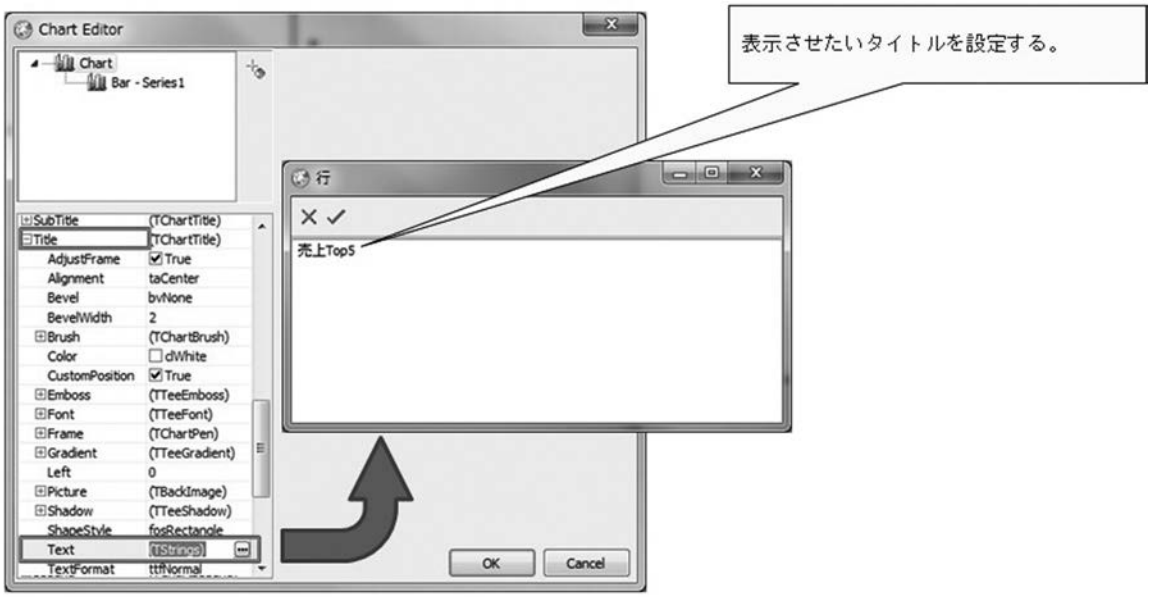

表示させたいタイトルを設定する。
Chart Editor
Chart
Bar - Series1
SubTitle (TChartTitle)
Title (TChartTitle)
AdjustFrame True
Alignment taCenter
Bevel bvNone
BevelWidth 2
Brush (TChartBrush)
Color clWhite
CustomPosition True
Emboss (TTeeEmboss)
Font (TTeeFont)
Frame (TChartPen)
Gradient (TTeeGradient)
Left 0
Picture (TBackImage)
Shadow (TTeeShadow)
ShapeStyle fosRectangle
Text (TStrings)
TextFormat ttfNormal
行
売上Top5
OK
Cancel

図20

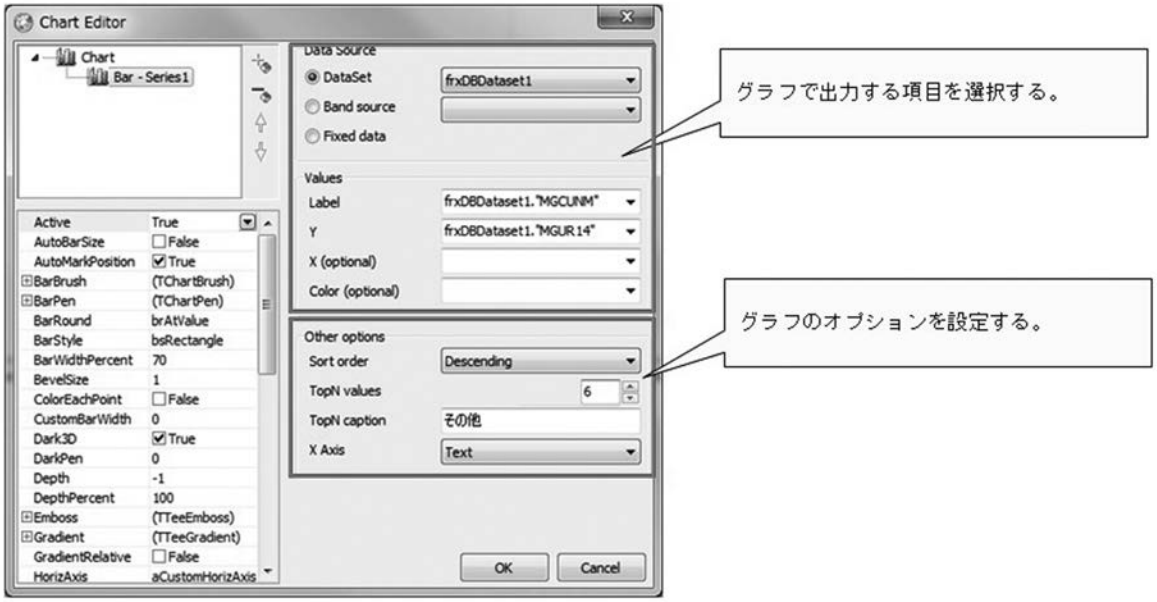
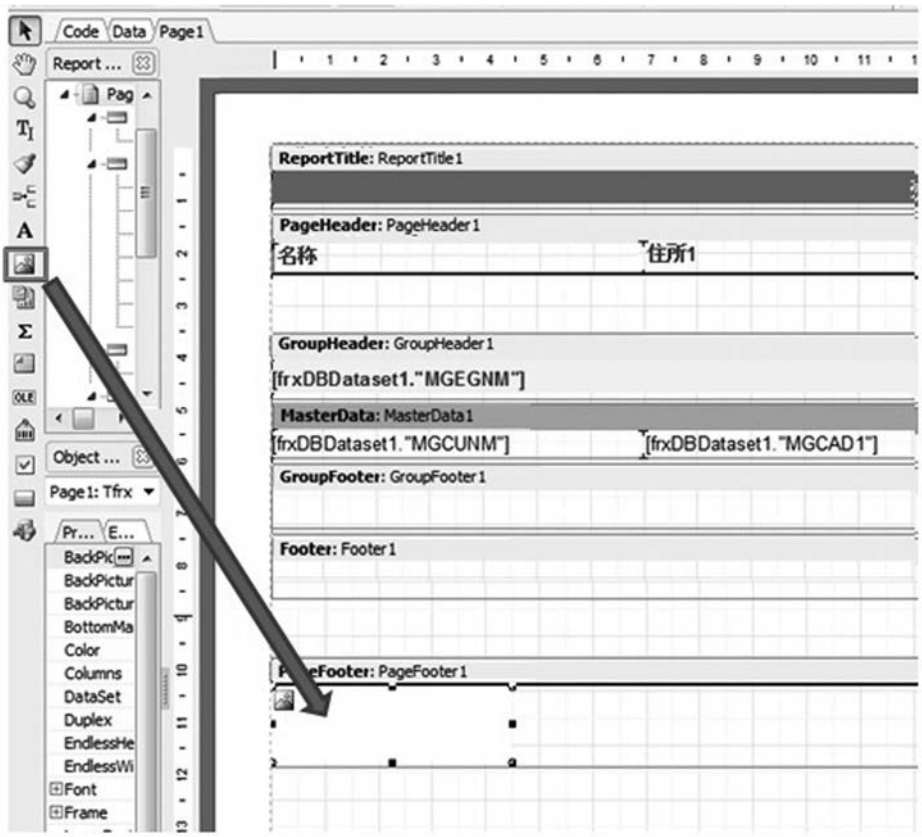


Chart Editor
Chart
Bar - Series1
Data Source
DataSet
frxDBDataset1
Band source
Fixed data
Values
Label
frxDBDataset1."MGCUNM"
Y
frxDBDataset1."MGUR14"
X (optional)
Color (optional)
Other options
Sort order
Descending
TopN values
6
TopN caption
その他
X Axis
Text
OK
Cancel
グラフで出力する項目を選択する。
グラフのオプションを設定する。

図21


株式会社四万十商店
高知県四万十市
1−1−5
XXX-XXXX-XXX
XXX-XXXX-XXX
10,000
小計:
12,910
合計:
73,110
2014/08/28 22:51:06
Page 1

図22


Code
Data
Page1
ReportTitle: ReportTitle1
PageHeader: PageHeader1
名称
住所1
GroupHeader: GroupHeader1
[frxDBDataset1."MGEGNM"]
MasterData: MasterData1
[frxDBDataset1."MGCUNM"]
[frxDBDataset1."MGCAD1"]
GroupFooter: GroupFooter1
Footer: Footer1
Object ...
Page1: Tfrx
Color
Columns
DataSet
Duplex
Font
Frame

ソース1

```
{*************************************************************************************
 目的：プレビューボタン押下時処理
 引数：
 戻値：
*************************************************************************************}
procedure TForm1.Button1Click(Sender: TObject);
var
  frxPic: TfrxPictureView;
begin

  // 画像の読み込み
  frxPic := frxReport1.FindObject('PICTURE1') as TfrxPictureView;
  frxPic.Picture.LoadFromFile('C:¥Test¥migaro_logo.jpg');

  // プレビュー処理
  frxReport1.ShowReport;
end;
```

保存したパスを指定する。

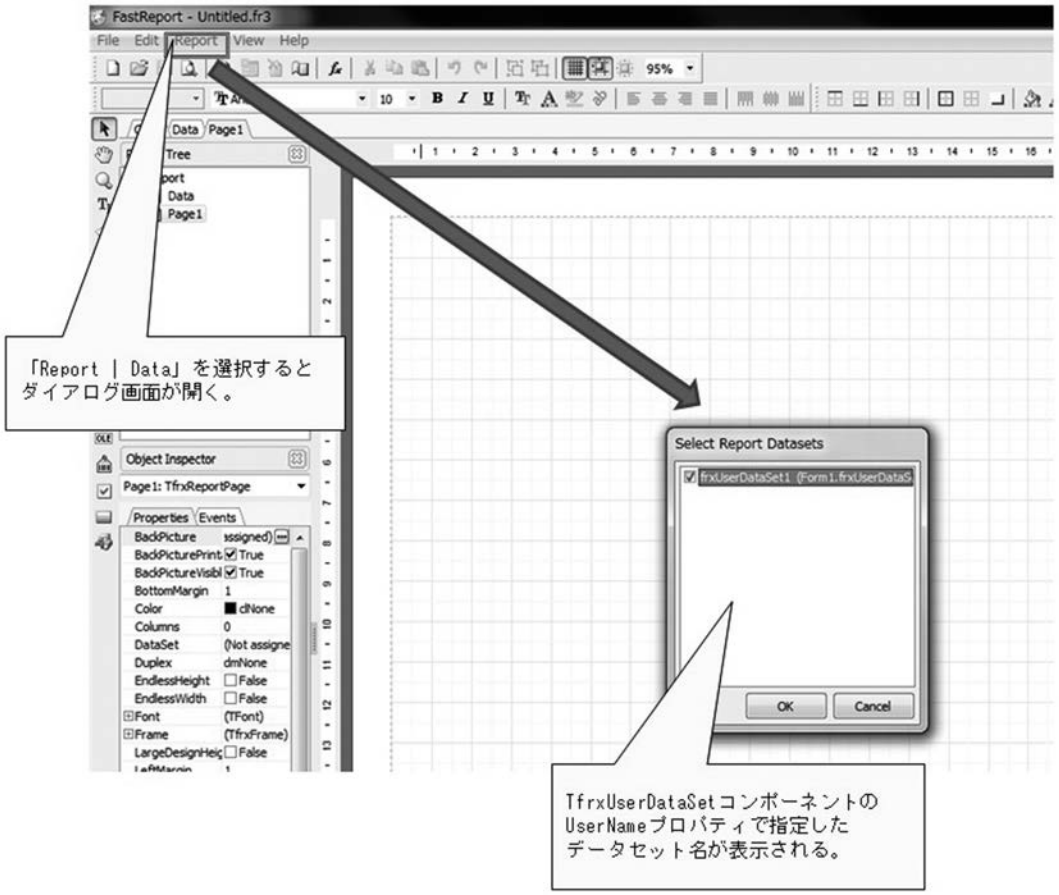
図23

営業所別売上一覧表
(過去5ヶ年)

営業所名	2010年	2011年	2012年	2013年	2014年
東京営業所	1,000	1,000	2,000	3,000	4,000
大阪営業所	2,000	2,000	4,000	6,000	8,000
名古屋営業所	3,000	3,000	6,000	9,000	12,000
青森営業所	4,000	4,000	8,000	12,000	16,000
広島営業所	5,000	5,000	10,000	15,000	20,000
高知営業所	6,000	6,000	12,000	18,000	24,000
仙台営業所	7,000	7,000	14,000	21,000	28,000
博多営業所	8,000	8,000	16,000	24,000	32,000
宮崎営業所	9,000	9,000	18,000	27,000	36,000
鹿児島営業所	10,000	10,000	20,000	30,000	40,000

図24

図25



ソース2

```
{*******************************************************************************
 目的：プレビューボタン押下時処理
 引数：
 戻値：
*******************************************************************************}
procedure TForm1.Button4Click(Sender: TObject);
begin

  // 配列データ取得処理
  GetURIRecord;

  // プレビュー処理
  frxReport1.ShowReport;

end;

{*******************************************************************************
 目的：データセットに配列データをセット
 引数：
 戻値：
*******************************************************************************}
procedure TForm1.frxUserDataSet1GetValue(const VarName: string;
  var Value: Variant);
begin
  if VarName = 'EGNAME' then
    Value := ArrayURIRec[frxUserDataSet1.RecNo].EGNAME   // 営業所名
  else
  if VarName = 'URKN10' then
    Value := ArrayURIRec[frxUserDataSet1.RecNo].URKN10   // 2010年売上金額
  else
  if VarName = 'URKN11' then
    Value := ArrayURIRec[frxUserDataSet1.RecNo].URKN11   // 2011年売上金額
  else
  if VarName = 'URKN12' then
    Value := ArrayURIRec[frxUserDataSet1.RecNo].URKN12   // 2012年売上金額
  else
  if VarName = 'URKN13' then
    Value := ArrayURIRec[frxUserDataSet1.RecNo].URKN13   // 2013年売上金額
  else
  if VarName = 'URKN14' then
    Value := ArrayURIRec[frxUserDataSet1.RecNo].URKN14;  // 2014年売上金額
end;

{*******************************************************************************
 目的：帳票出力終了チェック処理
 引数：
 戻値：
*******************************************************************************}
procedure TForm1.frxUserDataSet1CheckEOF(Sender: TObject; var Eof: Boolean);
begin
  // 配列数がデータセットのレコードNo.を超えた場合、出力処理終了
  Eof := frxUserDataSet1.RecNo > High(ArrayURIRec);
end;
```

帳票出力メイン処理

OnGetValueイベントの処理

If文で項目名を判断し、
Valueに値をセットする。

frxUserDataSet1.RecNoで
データセットの行番号を表
している。

OnCheckEOFイベントの処理

図26

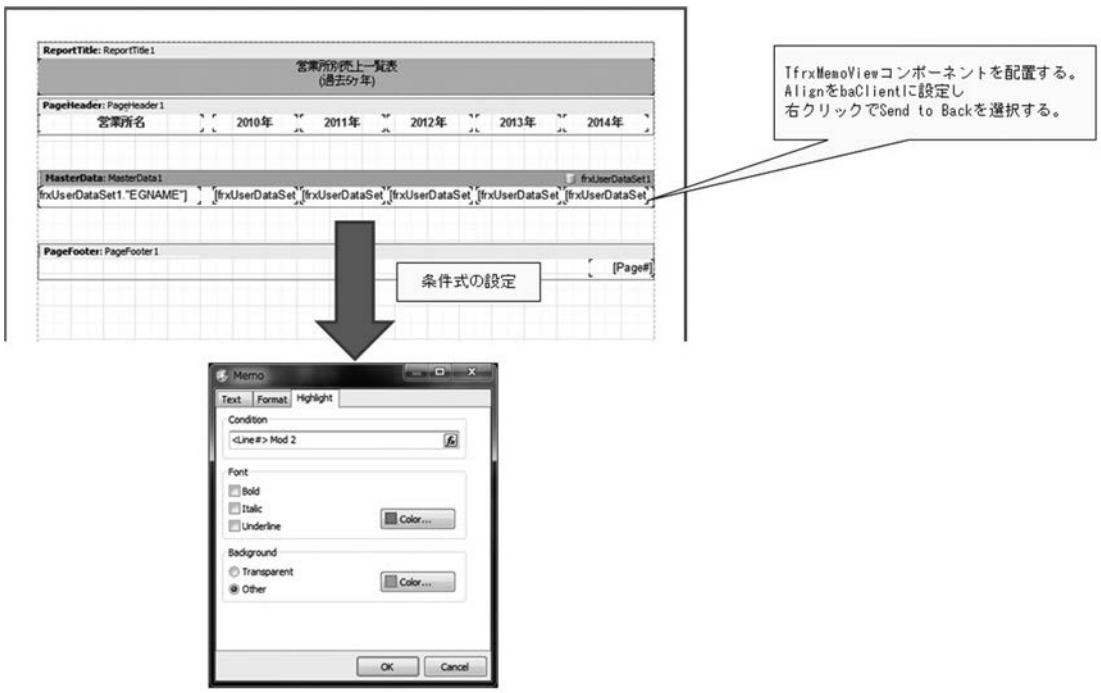


図27

営業所別売上一覧表
(過去5ヶ年)

営業所名	2010年	2011年	2012年	2013年	2014年
東京営業所	1,000	1,000	2,000	3,000	4,000
大阪営業所	2,000	2,000	4,000	6,000	8,000
名古屋営業所	3,000	3,000	6,000	9,000	12,000
青森営業所	4,000	4,000	8,000	12,000	16,000
広島営業所	5,000	5,000	10,000	15,000	20,000
高知営業所	6,000	6,000	12,000	18,000	24,000
仙台営業所	7,000	7,000	14,000	21,000	28,000
博多営業所	8,000	8,000	16,000	24,000	32,000
宮崎営業所	9,000	9,000	18,000	27,000	36,000
鹿児島営業所	10,000	10,000	20,000	30,000	40,000

ソース3

```
{*************************************************************************
 目的：プレビューボタン押下時処理
 引数：
 戻値：
*************************************************************************}
procedure TForm1.Button4Click(Sender: TObject);
begin

  // 配列データ取得処理
  GetURIRecord;

  // レイアウトの読み込み処理
  frxReport1.LoadFromFile('C:¥Report¥SampleReport.fr3');

  // プレビュー処理
  frxReport1.ShowReport;

end;
```

帳票出力メイン処理に
レイアウトの読み込み処理
を追加する。

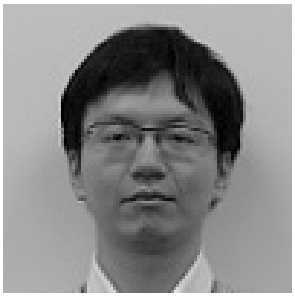
保存したパスを指定する。

佐田 雄一

株式会社ミガロ.

システム事業部

[Delphi/400] 大量データ処理テクニック
ーFTPを利用したデータ転送



略歴
1985 年 12 月 6 日生
2009 年 甲南大学 経営学部卒
2009 年 4 月 株式会社ミガロ. 入社
2009 年 4 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日々、精進している。

1.はじめに

クライアント PC から IBM i にデータ更新を行うプログラムにおいて、例えばバッチ系処理であれば何万件、もしくは何十万件という大量データを一括で更新する場合がある。こうした大量データ処理では、処理量に比例して時間を要するため、パフォーマンスが課題になることが多い。本稿では、Delphi/400 機能の 1 つである「Indy」を使用して、このような大量データ処理を行う際のレスポンス改善手法を紹介する。

2.Indyを利用したFTP転送

「Indy」とは、Delphi/400 で使用できるオープンソースのネットワーク関連コンポーネント群のことであり、Delphi/400 に標準で付属している。本稿ではこの Indy が持つ FTP 転送機能を用いて、クライアント PC から IBM i に、大量データの更新を行う手法を紹介する。

2-1. FTP転送の流れ

FTP（File Transfer Protocol）とは、サーバー間、またはサーバーとクライアント PC の間でファイルの送受信を行う際に利用される手段の 1 つである。IBM i とクライアント PC 間の FTP 転送は、主に SAVEFILE の送受信や、IBM i をファイルサーバーの代替として使用する場合に用いられる。

クライアント PC から IBM i へファイルを送信する場合、送信先には IBM i 内のライブラリを指定することが多いが、今回の処理では IFS（Integrated File System、統合ファイルシステム）領域を指定する。【図 1】

Delphi/400 と IFS の間でファイルの送受信を行う方法はほかにも存在するが、FTP 転送の技術は、一度習得すれば IBM i 以外のサーバー通信でも汎用的に活用することができる。

2-2. FTP転送を行うメリット

本稿のテクニックにおいて FTP 転送を利用する最大のメリットは、通信回数の削減によるレスポンスの向上である。

通常、アプリケーションのデータベース処理は 1 レコードごとに更新を行うため、処理レコード数が多い場合、同じ回数の通信も発生してしまう。しかし、クライアント PC から IFS 上に CSV 形式のファイルで更新内容を転送することができれば、IBM i 上では RPG で一括処理することができる。その際に便利なのが、IBM i の「CPYFRMIMPF」コマンドである。このコマンドを実行すると、その CSV ファイルの内容をデータベースファイルに直接転送することができる。

このコマンドと FTP でのファイル転送を組み合わせることで、従来であればレコード単位で発生していたクライアント PC と IBM i の間の通信が一度で済む。通信回数が削減された結果、処理時間を大幅に短縮することができるのであ

図1　IFSとクライアント間の通信イメージ

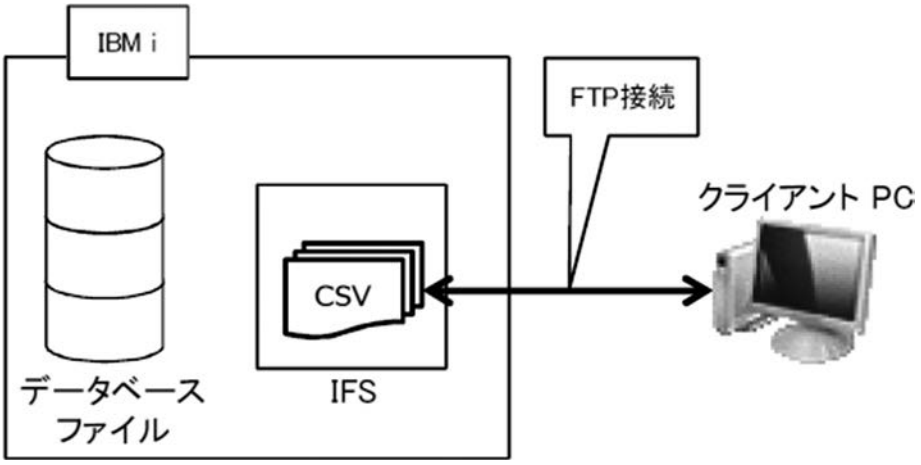


図2　10万件のレコードを同条件で更新した場合のレスポンス比較

更新方法	通信回数	処理時間
① 1レコード毎に都度RPGを呼び出して更新	10万回	約2～4分
② CSVをワークファイルに転送し、RPGで一括更新	2回(※)	約9～14秒

※　CSVファイルのFTP送信、更新RPGのCALLの計2回。

図3　FTP接続許可の確認

る。

【図2】は、IBM i上のあるデータベースファイル内のレコード10万件に対して、それぞれの手法で同じ更新を実行した場合の、通信回数と処理時間の例である。

この結果を比較すると、通信回数、処理時間ともにその差は大きく、②の手法が効果的であることがわかる。次章では、ここまでに紹介したテクニックをプログラムに実装する方法をサンプルとして紹介する。

3.クライアントPCとIBM i間のファイル、レコード転送

本章では、サンプルプログラムの作成を通じて、クライアントPCからIBM iへのFTP転送、およびCPYFRMIMPFコマンドを使用する方法を紹介する。なお、CPYFRMIMPFコマンドは、V4R3以降のIBM iで使用可能となっている。

3-1. FTP接続許可の確認

IBM iに対してファイルのFTP転送を行うための前提条件として、まずIBM iでFTP転送が許可されている必要がある。

確認方法を説明する。5250画面で「NETSTAT *CNN」コマンドを実行すると、IPV4接続状況の一覧が表示される。【図3】

この一覧の中で、「ftp-con」または21番のローカルポートが「接続待機」になっていれば、FTP接続が許可されている。存在しない場合は、5250画面側で「STRTCPSVR SERVER（*FTP)」コマンドを実行すると、開始することができる。

また、IBM i側で接続待機になっていてもFTP接続ができない場合は、ファイアウォールの設定でIBM iとのFTP接続が許可されているか確認していただきたい。

3-2. CSVファイルの書式

次に、IBM iに転送するCSVファイルの書式について説明する。【図4】はCSVファイルに格納したデータの例である。

CSVファイルの項目の並び順は、レコード転送先となるデータベースファイルのフィールド順と揃える必要がある。また、文字フィールドの値がブランクにならないように注意していただきたい。ブランクが指定されているとCPYFRMIMPFコマンド実行時にNULLとして認識されるため、意図的にNULLを渡す場合を除き、ブランクを転送するフィールドには1文字以上の半角スペースを指定する。

それ以外の制約についてはCPYFRMIMPFコマンドのパラメータに依存するため、あとはフィールドの過不足と桁あふれに気を付ければよい。CPYFRMIMPFコマンドの詳細については後述する。

3-3. サンプルプログラムの作成

上記の準備事項が確認できたら、CSVからワークファイルへの基本的なレコード転送と、更新RPGの呼び出しを行うプログラムを実際に作成していく。なお、本稿で作成しているサンプルプログラムは、Delphi/400 Version XE3を使用している。

本項でのサンプルプログラムは、次の手順で作成する。

(1)コンポーネントの配置
(2)ファイル指定［…］ボタンクリック時処理の作成
(3)送信ボタンクリック時処理の作成
(4)コマンド実行処理の作成

なお、事前準備として、IFS領域の「/QIBM/UserData」フォルダ内に「TR07/CSV」フォルダを作成しておく。本稿のサンプルプログラムでは、このフォルダに対してFTP転送を行う。また、CSVファイル名、更新先のライブラリ名およびワークファイル名は、環境に合わせて適宜読み替えていただきたい。本稿のサンプルプログラムにおけるデータの流れを【図5】に示す。

(1)コンポーネントの配置

このサンプルプログラムでは、Indyの機能の1つである「TIdFTP」コンポーネントを用いて、IBM iとのFTP転送を行う。

最初に、ネイティブ接続を行うためのTAS400、接続先情報と送信ファイルパスを入力または指定するためのTEdit、FTP転送を行うためのTIdFTP、送信ファイル指定のためのTOpenDialog、ならびにTLabelやTButtonをそれぞれ画面に配置する。【図6】

(2)ファイル指定［…］ボタンクリック時処理の作成

TOpenDialogコンポーネントを使用し、アップロードするファイルを選択するダイアログを表示させる。【ソース1】

(3)送信ボタンクリック時処理の作成

配置した送信ボタンのOnClick処理を作成したら、TIdFTPコンポーネントを使った接続・転送のソース記述を行う。【ソース2】

以下に、TIdFTPコンポーネントが持つプロパティや、今回の処理で行っているメソッドについて解説する。

(3)-①接続設定と接続処理

Host、Username、Passwordの各プロパティに、FTP転送を行うための値を設定する。Hostには接続先IBM iのサーバー名（IPアドレス)、UsernameとPasswordには接続に使用するユーザー名とパスワードを指定する。接続設定が完了したら、Connectメソッドで接続を行う。接続後は、try～finallyで処理を囲み、処理終了後はDisConnectメソッドで接続を終了する。

(3)-② TransferTypeプロパティ

ftASCII、ftBinaryの2種類が存在し、ファイルの送受信をテキスト形式、バイナリ形式のどちらで行うかを事前に指定することができる。しかし、ftASCIIを指定するとテキスト形式であっても送受信の結果に文字化けが発生する場合があるため、常にftBinaryに設定しておくことを推奨する。

なお、ftBinaryはIdFTPCommon.pasで定義されているので、uses節に「IdFTPCommon」を追加する。

(3)-③ ChangeDirメソッド

引数で指定した名前のディレクトリを、カレントディレクトリ（ユーザーが現時点で作業を行っているディレクトリ）に設定する。

《第1引数》カレントディレクトリに

図4　転送するCSVの項目設定例

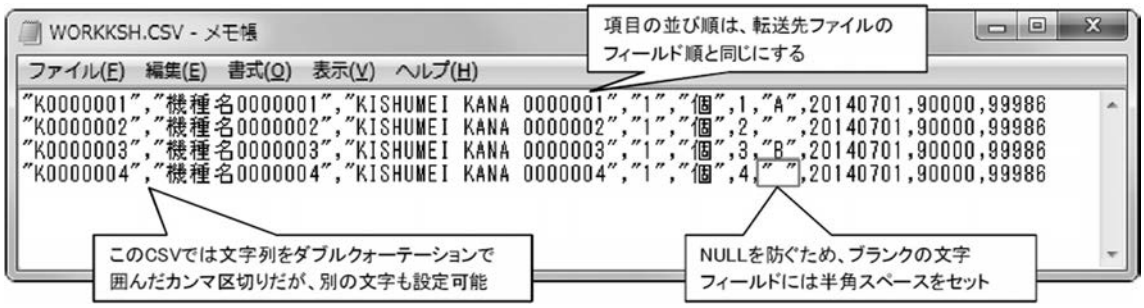


図5　サンプルプログラムのデータの流れ

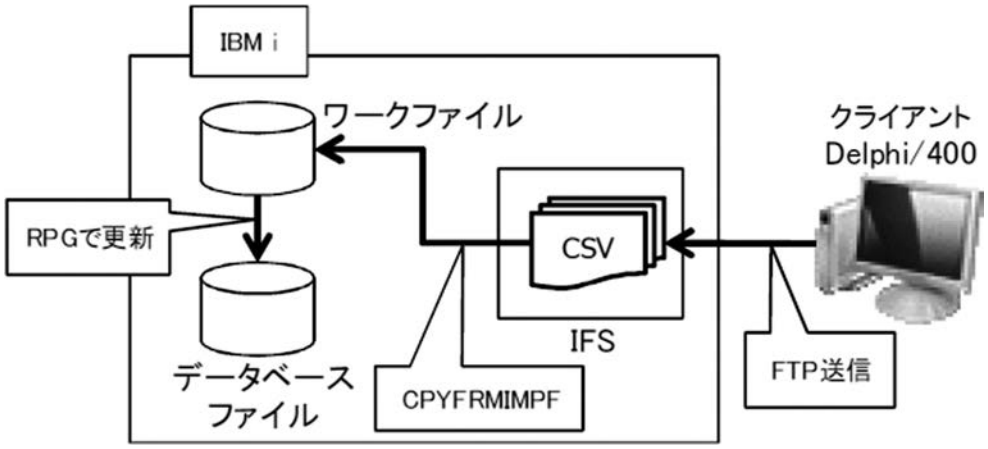


図6　サンプルプログラム画面とコンポーネント配置

設定するディレクトリパスを指定する。
※ ChangeDirUp メソッドを使うと、名前を指定せずに1つ上のディレクトリをカレントディレクトリに設定する。

(3)-④ Put メソッド

引数で指定した条件で、ファイルをFTP送信先にアップロードする。第2以降の引数は省略可能である。
《第1引数》アップロード元ファイルのフルパスを指定する。
《第2引数》アップロード先のファイル名を指定する。省略時は、第1引数と同じファイル名になる。
《第3引数》True を指定すると、上書きする。省略時は False。

(4) コマンド実行処理の作成

上記（3）の処理の最後に、IBM i 側で取り込んだ CSV のレコード転送コマンド、および更新 RPG を実行するコマンドを記述する。【ソース 3】ではコマンドとして発行しているが、TCall400 コンポーネントを利用して CL として呼び出して処理を行うこともできる。

ここで、使用する CPYFRMIMPF コマンドで使用する主なパラメータについて解説する。

(4)-① FROMSTMF（FROM ストリームファイル）

先述の TIdFTP コンポーネントで IFS 内に送信したファイル（今回は「WORKKSH.CSV」）をパス付きで指定する。

(4)-② TOFILE（TO データベースファイル）

CSV の内容を更新するデータベースファイルを指定する（今回は「TR07LIB/WORKKSH」）。

(4)-③ MBROPT（追記 / 上書きの指定）

「*REPLACE」指定時は、更新するデータベースファイルの全レコードを①のファイルの値に置き換える。「*ADD」指定時は、レコードを追記する。レコード追記の際はエラーを防ぐため、ユニークキーの重複がないか転送前に確認しておく。

(4)-④ RCDDLM（レコード区切り文字の指定）

「*CRLF」指定時は、改行とそれに続く行送りのコードを識別して、レコードの分割を行う。改行コード以外の文字も指定可能である。

(4)-⑤ STRDLM（ストリング区切り文字）※任意

文字列を囲む際の囲み文字の指定を行う。デフォルト値は「*DBLQUOTE」で、ダブルクォーテーションで囲まれた文字列を認識する。半角文字列指定時（例：「'#'」）はその文字で認識する。「*NONE」指定時は認識を行わない。

(4)-⑥ FLDDLM（フィールド区切り文字）※任意

一般的な CSV と同様に、デフォルト値はカンマ（','）となっている。タブ文字区切りにする場合は「*TAB」を、その他の文字で区切る場合はその文字を指定する。

これらのパラメータのうち①～④が必須指定、⑤と⑥が任意指定となっている。ここに記載した以外のパラメータについては、5250 上での実行において、ヘルプなどを確認していただきたい。以上で、サンプルプログラムは完成である。
実行すると、この FTP を用いた仕組みでの大量レコード処理のパフォーマンスを確認できる。

4.まとめ

大量データ処理のプログラムにおいて、どれだけパフォーマンスよく処理できるかは、1つの重要なポイントであり、課題になることも多い。もしこうした大量データ処理でパフォーマンスが課題となった場合には、本稿のレスポンス向上テクニックを一度お試しいただきたい。1レコードごとの処理で何時間もかかっているようであれば、この仕組みによって大幅に処理時間を短縮できる可能性がある。
本稿で紹介したテクニックが、業務システムをさらによくするための一助となれば幸いである。

M

ソース1　送信元ファイル指定処理

```
{*************************************************************************
 目的：送信元ファイル指定
 引数：
 戻値：
*************************************************************************}
procedure TForm1.Button2Click(Sender: TObject);
begin
  if (OpenDialog.Execute) then
  begin
    edtFromFile.Text := OpenDialog.FileName;
  end;
end;
```

ファイル選択ダイアログが開き、
OKが押されるとFileNameがセットされる。
キャンセルされた場合はif～の中を通らない。

ソース2　クライアントのファイルをIBM iに送信

```
// FTP接続
if IdFTP.Connected then
begin
  IdFTP.DisConnect;
end;
IdFTP.Host     := edtHOST.Text;
IdFTP.Username := edtUSER.Text;
IdFTP.Password := edtPASS.Text;
IdFTP.Connect;

if IdFTP.Connected then
begin
  try
    IdFTP.TransferType := ftBinary; // バイナリ形式

    IdFTP.ChangeDir('/QIBM/UserData/TR07/CSV');

    IdFTP.Put(edtFromFile.Text, '', False);
  finally
    IdFTP.DisConnect; // 最後に接続終了
  end;
end;
```

①接続設定と接続処理

②TransferTypeプロパティ
テキストでもバイナリ形式を指定

③ChangeDirメソッド
転送先ディレクトリを設定

④Putメソッド
ファイルをアップロード

ソース3　転送および更新のコマンド実行

```
// IFS→データベースファイル 転送コマンドの実行
AS400.RemoteCmd(' CPYFRMIMPF ' +
                ' FROMSTMF(''/QIBM/UserData/TR07/CSV/WORKKSH.CSV'') ' +
                ' TOFILE(TR07LIB/WORKKSH) ' +
                ' MBROPT(*REPLACE) ' +
                ' RCDDLM(*CRLF) ');

// 更新RPGの実行
AS400.RemoteCmd(' CALL PGM(TR07LIB/TR07RPG) ');
```

更新RPGの実行

IFSからデータベースファイルへのレコード転送処理
（引数について、下記1.～4.の4パラメータが必須）

1. FROMSTMF	IFS内の転送元CSVファイル
2. TOFILE	転送先データベースファイル
3. MBROPT	追記/上書きの指定（上書き）
4. RCDDLM	レコード区切り文字の指定（改行キー）

尾崎 浩司 / 國元 祐二

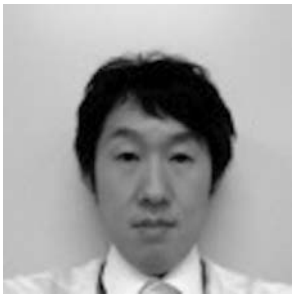
株式会社ミガロ.

RAD事業部 営業・営業推進課 / RAD事業部 技術支援課

[SmartPad4i] スマートデバイスWebアプリケーション入門
—HTMLを使ったユーザーインタフェースの工夫

略歴　尾崎 浩司
1973 年 8 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 4 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・運営などを主に担当しています。



略歴　國元 祐二
1979 年 3 月 27 日生
2002 年 3 月 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、Business4 Mobile の製品試験やサポート業務などを行っています。

1.はじめに

近年、スマートフォンやタブレットといったスマートデバイスを業務システムでも活用したいという要求が大きくなっている。

JC/400 は、IBM i を活用した PC 向け Web アプリケーション開発ツールだが、そのオプションである SmartPad4i を使用するとスマートデバイス向けの Web アプリケーションも作成できるようになる。

PC 向けの画面の場合、十分なサイズの画面や操作性の高いマウスやキーボードが使用できるが、スマートデバイスの場合、画面サイズに制約があり、入力操作などもソフトキーボードとなるため、表示や入力に制約が発生しやすい。

つまり、使いやすいスマートデバイス向けの Web アプリケーションを作成するには、ユーザーインターフェースの工夫が必要となるのである。

本稿では、SmartPad4i を使用した、スマートデバイス向けの Web アプリケーションを作成する際に役立つスキルを紹介したい。

2.JC/400、SmartPad4iにおける画面作成の基本

JC/400 や SmartPad4i でアプリケーションを作成する場合、従来の 5250 アプリケーションで作成していた DSPF（画面ファイル）の代わりに HTML を使用して画面を作成する。その際、入出力項目の要素にユニークな id 名を設定するのがルールである。DSPF におけるフィールド ID 同様、HTML の要素に id 名を設定することで、RPG あるいは COBOL のプログラムから入出力項目にアクセスできるようになる。

作成した HTML を Designer で読み込み、入出力フィールドの型や長さを指定後、IBM i に RPG あるいは COBOL を配布する。配布されたスケルトンの RPG/COBOL に業務ロジックを記述することで、簡単に Web アプリケーションが作成できるのが特徴だ。【図 1】

JC/400 や SmartPad4i では、型や文字長に応じたエラーチェックや入力制御が自動的に設定される。また、RPG/COBOL 側のサブルーチンを呼び出すだけで、HTML の属性を変更する機能も用意されているため、JavaScript などを使用しなくても、使いやすい Web アプリケーションが構築できるようになっている。

もちろん、標準機能で一通りの Web アプリケーションを構築できるのだが、さらに HTML や JavaScript などを活用すれば、よりさまざまな機能やインターフェースを作成できる。

それでは本稿のテーマである、スマートデバイス向けの Web アプリケーション作成に役立つスキルとして、まずは HTML5 を活用した SmartPad4i のインターフェースの拡張に焦点をあてて紹介していこう。

図1　JC/400, SmartPad4iWEBアプリケーション開発手法

図2　HTML5 PlaceHoder属性

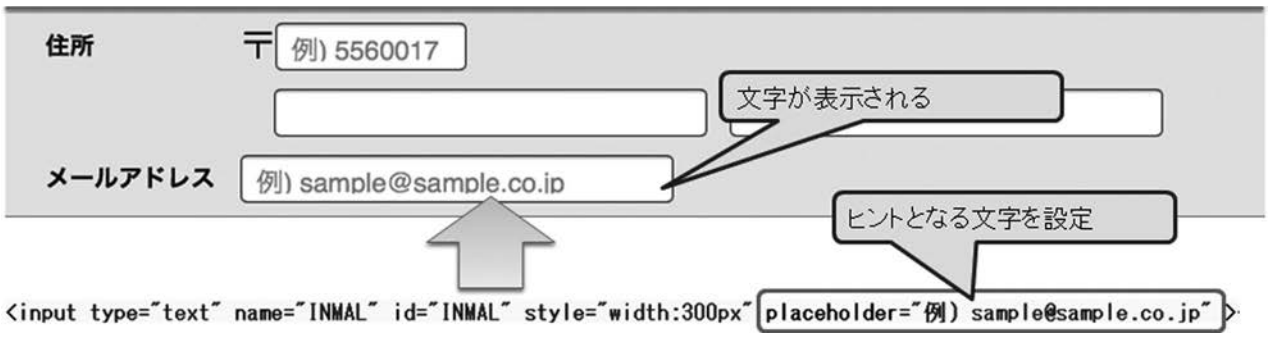
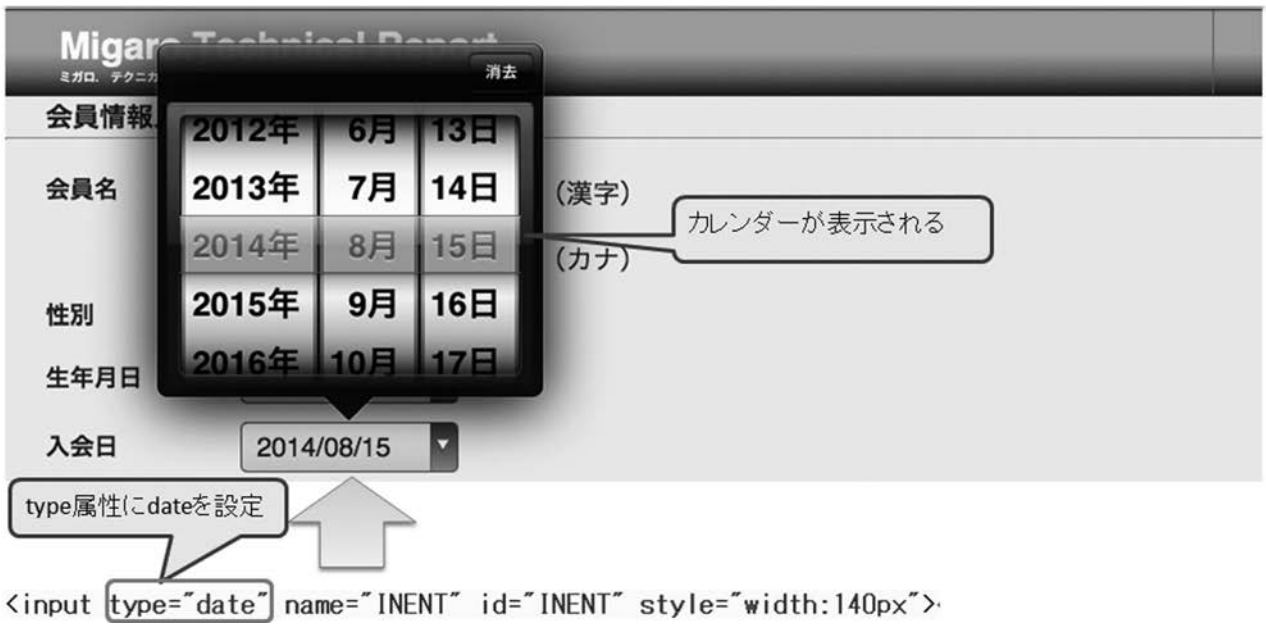


図3　HTML5 type属性date



3.HTML5を利用した画面項目の工夫

HTML5とは?

HTML5 は、World Wide Web Consortium（W3C）で進められているHTML の 5 度目のメジャーバージョンアップで、2008 年頃に草案が発表され、現在正式勧告に向けて策定中の最新仕様である。ブラウザの種類ごとに対応状況は異なるが、さまざまな新しい機能が各ブラウザに実装されてきている。

従来の HTML では、flash や Java などのプラグインを利用しないと動画や音声、グラフィックなどを制御することは難しかったが、HTML5 では、新たに追加された video タグや、audio タグ、canvas タグなどを使用することで、プラグインなしでこれらを取り扱えるようになった。それ以外にも、クライアント上に大量データを保持する機能や、現在位置情報の取得、フォーム上の入力書式設定や妥当性チェックなどさまざまな機能が追加されている。

スマートデバイスに対応した SmartPad4i では、この HTML5 を利用することが可能である。特にスマートデバイスで利用される Safari や Chrome といったブラウザは HTML5 への対応が進んでいるため、HTML5 を活用することで、今までにないインターフェースや表現を容易に実現できるようになった。

まずは、SmartPad4i で比較的簡単に利用でき、効果的な HTML5 の機能について紹介したい。

placeholder

Web アプリケーションで、薄いグレーの文字があらかじめ入っている入力欄を見たことがないだろうか。HTML5 で追加された placeholder 機能を利用すると、フォームの入力欄にあらかじめ入力のヒントとなる文字列を設定しておくことができるのである。例えば、メールアドレスの入力欄にあらかじめ、“例）sample@sample.co.jp”のような入力例を初期表示しておくことができる。

以前の HTML では、同様の機能を JavaScript で実装する必要があったが、placeholder を利用すると、input タグや textarea タグに placeholder 属性を追加するだけで利用できるようになる。【図 2】

スマートフォンなどの限られた画面サイズでは、別途説明エリアを設けるのは難しいことが多いため、こういった工夫が効果的だろう。これは簡単に追加できるので、ぜひ入力項目に設定することをお勧めしたい。

フォーム機能の拡張 type属性

HTML5 では、フォーム機能が拡張されて、さまざまな type 属性が利用できるようになった。その中で今回は、type 属性の date について取り上げたい。

input タグの type 属性に date を設定すると、HTML5 に対応しているブラウザであれば、カレンダーなどの入力補助が表示されるようになる。例えば、iPhone や iPad で利用される Safari では、ドラム式のカレンダーが表示される。【図 3】

このように type 属性を利用するだけで、リッチなインターフェースを利用できるのだ。しかも date を指定したものは日付値の入力ということが明確なため、日付値以外の値が入力できなくなるのも利点であろう。

画面自体はこれだけで利用できるようになるが、これを SmartPad4i の入出力項目として利用するには、ひと工夫必要だ。なぜならば、この date を指定した input 項目は、“2014-01-01”のように 10 桁の文字として値がセットされるからである。つまり date を利用する場合、SmartPad4i の Designer における System i Type（属性）は Atype に、System i length（桁数）は 10 に設定する必要がある。

IBM i 上のファイルでは、日付値を“20140101”のような数値 8 桁で保持することが多いだろう。そのため、例えば入力項目であれば、IBM i 側で日付文字列 10 桁を数値 8 桁に変換するようなロジックを作成すればよいのである。【ソース 1】は、RPG を使用した場合の記述例だ。【ソース 1】

逆に出力項目の場合には、8 桁の数値を“2014-01-01”のような書式に変換して出力すればよいこともご理解いただけるであろう。

入力妥当性チェックの追加

HTML5 では、入力欄の妥当性チェック機能として、未入力チェックの required 属性や、正規表現を使ったチェックが可能な pattern 属性が追加された。

SmartPad4i でも、これらの属性と CSS の invalid フィルターとを組み合わせて利用することで、入力誤り通知機能を追加することができる。

必須入力欄に文字が未入力である場合や、形式に合わない文字列が入力されているなど、入力に不備がある場合に、入力欄の背景色を変更するには、以下の設定を行う。

まず、【ソース 2】のように HTML の head タグ内に style タグで CSS の定義を追加する。これは、項目値が不正（invalid）となった場合の書式を設定しており、今回の場合は、background-color（背景色）の色コードを定義しているのである。【ソース 2】

あとは、HTML 内の各 input タグに必須項目であれば、required 属性を、書式をチェックしたい場合は、pattern 属性を追加するだけで実現できる。【図 4】

書式をチェックする pattern の場合、チェックしたい書式を正規表現で定義するのがポイントである。

正規表現とは、文字列の集合を 1 つの文字列で表現する方法で、例えば、郵便番号やメールアドレスなど特定の文字列パターンで表せるものをチェックするために利用することが多い。

例えば、郵便番号であれば、一般に 7 桁の数字で表現できる。数字 7 桁を定義する正規表現は、“^¥d {7}”のようになる。

これは、先頭文字（^）から数字（¥d）が 7 桁（{7}）であることを表現しているのである。また、メールアドレスの場合は、例えば、“^[a-zA-Z0-9. !#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:¥.[a-zA-Z0-9-]+)*$”のように表現するとよいだろう。

簡単な実装で、リアルタイムに入力の誤りを表現することができるので、ぜひ活用していただきたい。

ソース1　type属性date RPG側で変換

```
0432.00      *----------------------------------------------------------------
0433.00      *  文字型を日付に変換する
0434.00      *----------------------------------------------------------------
0435.00     C           SBCRDY    BEGSR
0436.00     C*
0437.00     C                     MOVEL*BLANK    WKYYMM
0438.00     C                     MOVEL*BLANK    WKYYMD
0439.00     C*
0440.00     C           4         SUBSTWKGDAY:1  TYEAR   4        年 YYYY
0441.00     C           2         SUBSTWKGDAY:6  TMON    2        月 MM
0442.00     C           2         SUBSTWKGDAY:9  TDAY    2        日 DD
0443.00     C**
0444.00     C           TYEAR     CAT  TMON:0    WKYYMM
0445.00     C           WKYYMM    CAT  TDAY:0    WKYYMD
0446.00     C                     MOVE WKYYMD    WKIDAY           日付に変換
0447.00     C                     ENDSR
```

10桁の文字を8桁の数値にする

ソース2　CSS3 invalidフィルターの定義

HTMLファイル

```
<!DOCTYPE html>↵
<html>↵
<head>↵
<meta charset="Shift_JIS" />↵
<meta name="viewport" content="width=device-height" />↵
<meta name="format-detection" content="telephone=no" />↵
<link rel="stylesheet" href="DEMOCSS.css" type="text/css">↵
<title>MIGARO. Customer System</title>↵
<style>↵
input:invalid {↵
 background-color: #ffcccc;↵
}↵
</style>↵
</head>↵
<body>↵
```

入力に誤りがある場合、背景色変更する定義

図4　HTML5 require、pattern属性による入力確認

require属性

未入力の場合は背景色が変わる

会員名　例) 山田　太郎　(漢字)

会員名　ミガロ　(漢字)

requiredを設定

```
<input type="text" name="INNM1" id="INNM1" required placeholder="例) 山田□太郎">
```

pattern属性

数値7桁でない場合は背景色が変わる

住所　〒 55600

住所　〒 5560017

patternを設定

```
<input type="text" name="INAD1" id="INAD1" maxlength="7" pattern="^¥d{7}$">
```

4.HTML5を利用した画面表示の設定

前節では、入出力項目となるinputタグで活用可能なHTML5を紹介したが、本節では画面全体の制御に関連する項目を紹介したい。

viewport

SmartPad4iはスマートデバイスに対応したWebアプリケーションが作成できるツールである。スマートフォンや、タブレットはデバイスごとに画面サイズがそれぞれ異なるため、画面サイズに適切なインターフェースの作成が必要となる。

スマートデバイスに対応していない、Webサイトをスマートフォンで見たことがあるだろうか。

スマートデバイスで、PC向けのサイトをそのまま表示すると、縮小してサイト全体を表示する。その結果、文字が小さくなってしまい、拡大しないときちんと文字が読めない状態になるだろう。【図5左】

しかし、最近のWebサイトでは、スマートフォン用画面が用意されており、スマートフォンでアクセスした際に利用しやすい表示になることが多い。【図5右】

この時、スマートデバイス向けにHTMLを作成していることをブラウザに通知する方法が、viewportである。

スマートフォンやタブレット向けにインターフェースをデザインする場合には、headタグ内にmeta要素としてviewportを指定すると、画面がデバイスのサイズに合わせて適切なサイズで表示してくれるのである。

例えば、【ソース3】のようにviewportを定義した場合、デバイスの横幅に合わせて描画をする指定となる。【ソース3】

また、viewportには、画面の横/縦幅に合わせて描画する機能だけでなく、初期表示時の拡大率や、拡大縮小の操作の制御の可否なども設定することができる。【ソース4】

例えば、初期表示される拡大率はinitial-scaleで設定可能である。100%サイズの場合を1.0とし、0～10の範囲で数値が大きくなるに従って、初期表示される拡大率が大きくなるのである。同様にminimum-scaleは最小の拡大率、maximum-scaleは最大の拡大率で、initial-scaleと同様に0～10の範囲を指定可能だ。user-scalableはユーザーのピンチ操作（縮小や拡大）の可否を設定できる。

このようにviewportの設定を適切に指定することで、ユーザーが利用しやすい画面を作成することができる。スマートフォンやタブレット向きのアプリケーション画面を作成する場合には、viewportの設定をお勧めしたい。

レスポンシブルデザイン

viewportの項でも触れたが、PCとスマートフォンとでは、画面サイズが異なるため、PCブラウザ向け画面と、スマートフォン向け画面の2つを用意したい場合があるだろう。その場合、通常2種類のHTMLを作成する必要があるが、レスポンシブルデザインというCSSの機能を利用したWebデザインの手法により、1つのHTMLをPCブラウザとスマートフォンの両方に対応させることが可能になる。【図6】

このレスポンシブルデザインは、SmartPad4iのアプリケーションでも有効で、1つのHTMLでPCブラウザとスマートフォンの両方に対応した画面を表示するアプリケーションを作成することができるのだ。【図7】

このレスポンシブルデザインで使用するCSS（カスケーディング・スタイル・シート）とは、Webページのスタイルを指定するための言語で、文書定義であるHTMLと組み合わせて利用するものだ。ここではCSSの基本については触れないが、インターネットなどで検索すると説明があるので、ご存じない方は一度確認しておいてほしい。

レスポンシブルデザインは、HTML5で追加されたCSSのメディアクエリーという機能を利用することで作成できる。この機能を用いると、画面サイズごとにデザイン定義であるCSSを切り替えて画面を表示できるのだ。

メディアクエリーの記述方法は簡単である。

例えば、@media内にmax-width：640pxと指定すると、表示横幅640px以下の場合のみ適用するCSSができるのである。【ソース5】

作成したCSSに名前を付けて保存しておき（ここでは“DEMOCSS_mobile.css”として保存）、文書定義であるHTMLと同一階層へ配置の上、HTMLの外部リンクで、通常サイズ用の外部スタイルシートを読み込んだあとに、640px以下用の外部スタイルシートの読み込みを追加すればよい。【ソース6】

このような定義を行うと、PCやタブレットを想定した640pxより大きいサイズをもつデバイスの場合は、標準のCSS（この例では、“DEMOCSS.css”）のみが適用されるが、スマートフォンを想定した640px以下のデバイスの場合、標準CSSに付加して、“DEMOCSS_mobile.css”が適用されるのである。

SmartPad4iの場合には、1つのHTMLが1つのRPGプログラムと関連付けられるため、レスポンシブルデザインを利用することで、異なる画面サイズ用に類似のプログラムを複数作成しなくてもよいというメリットも出てくるであろう。そのため、PC用とスマートフォン用の2つのデザインを用意したい場合には、レスポンシブルデザイン対応のHTMLの作成をお勧めしたい。

5.HTML5を活用したグラフの実現

canvasについて

データを分析する際、最も有効なのはグラフの利用である。直感的に状況がわかるグラフをIBM i上に登録されたデータから表示できると便利であろう。そこで今回は、HTML5のcanvasを使ったグラフ表示方法を紹介したい。

canvasは、HTML5で追加された要素で、ブラウザ上に図形を描画するために利用するものである。

従来のHTMLでは、FlashやJavaを別途利用しなければブラウザ上に図形を描画するのは困難であったが、HTML5で追加されたcanvasにより、JavaScriptベースで簡単に図形描画できるようになった。

もちろん、canvasを利用すれば、自由に図形描画ができるため、独自の実装でグラフを作成することも可能だが、それでは手間がかかるだろう。そこで今回は、canvasにグラフを描画する方法として、オープンソースで提供されている

図5　yahoo! トップページ

ソース3　Viewportの記述1

```
<!DOCTYPE html>↵
<html>↵
<head>↵
<meta charset="Shift_JIS" />↵
<meta name="viewport" content="width=device-width" />↵
<meta name="format-detection" content="telephone=no" />↵
<link rel="stylesheet" href="DEMOCSS.css" type="text/css">↵
<title>MIGARO.Technical Report</title>↵
```

headタグ内にviewportを定義

ソース4　Viewportの記述2

```
<!DOCTYPE html>↵
<html>↵
<head>↵
<meta charset="Shift_JIS" />↵
<meta name="viewport" content="width=device-width,initial-scale=1.0,↵
minimum-scale=1.0,maximum-scale=1.0,user-scalable=no">↵
<meta name="format-detection" content="telephone=no" />↵
<link rel="stylesheet" href="DEMOCSS.css" type="text/css">↵
<title>MIGARO.Technical Report</title>↵
```

headタグ内にviewportを定義
拡大率固定、ユーザの拡大縮小操作をさせない

【Flotr2】ライブラリを利用したい。（公式サイト：http：//humblesoftware.com/flotr2/）

Flotr2を利用すると、canvas上に、棒グラフや円グラフ、レーダーチャートグラフのような、さまざまなグラフを簡単に描画できる。

なお、グラフの描画には、外部Webサービスである Google Chart APIなどを組み合わせた方法もあるが、今回の方法はインターネットに接続しなくても実行できることがメリットである。また、外部Webサービスの場合、サービス提供者がAPIの仕様を変更する可能性もあるため、その際にはプログラムの変更対応が必要となるが、今回の方法では、ライブラリのスクリプトファイルをバージョンアップしない限り、同じプログラムを使い続けられることもメリットだ。

ここでは例として、会員の年代別割合を表示する円グラフの実現方法を説明する。データベースにある会員情報には、年齢が含まれるため、年齢から年代を取得して年代あたりの会員数を集計して、それをグラフで出力するのだ。【図8】

HTMLの作成

HTMLを利用したユーザーインターフェース部分を作成していこう。ここではグラフを埋め込むのに必要な箇所について説明したい。

(A) Flotr2 の設定

まず、HTMLにFlotr2ライブラリを読み込む必要がある。https：//github.com/HumbleSoftware/Flotr2からDownloadZIPボタンをクリックしてファイルをダウンロードする。【図9】

次に、ダウンロードしたファイルを展開したデータ内にあるflotr2.min.jsを取得して、HTMLと同階層にjsフォルダを作成後、flotr2.min.jsを配置する。

配置後、HTMLのheadタグ内にFlotr2の外部JavaScriptファイルの参照を追加すればよい。【ソース7】

(B) 表示領域の設定

次に、HTMLにグラフを表示する領域を定義する。今回、グラフ本体を描画するdivタグには、id名にgraphViewと指定している。【ソース8】

グラフ出力の実装例は後述するが、このgraphViewに対してFlotr2のグラフ出力処理を実行するのである。また、id名がGDATAのtable要素には、会員の年代と会員数を出力する。つまり、SmartPad4iのサブファイルを利用して、RPG側から年代と会員数を出力するのだ。

RPGの作成

(A) 配列定義の設定

次にIBM i側の処理例を紹介しよう。今回はRPGでの実装例を紹介したい。RPG側の定義では、年代別のラベルと年代別の会員数の合計を保持する配列を定義する。【ソース9】

今回の例では、コンパイル時配列を利用し、【ソース10】のように10代から100代までのラベル名称を定義した。【ソース10】

(B) サブファイル出力の設定

【ソース11】および【ソース12】が会員ファイルを読み込んで、会員一覧を画面出力する際に、年代別の会員数をカウントして、その結果を年代リスト用のサブファイルに出力する例である。【ソース11、ソース12】

年代別のラベルと会員数を出力可能な仕組みができれば、RPG側の作業は完了だ。

グラフの作成

(A) initpage 関数の設定

次は、グラフの表示元データとなる、年代別のサブファイルをJavaScriptから読み込み、Flotr2のグラフ出力機能を呼び出す処理を記述する。

ここで、initpage関数とcansubmit関数について説明しておこう。JC/400およびSmartPad4iでは、HTML内にinitpage関数を定義しておくと、RPG/COBOLプログラムが実行されて画面がブラウザに表示される時に、initpage関数に記述したJavaScriptが実行される。

また、cansubmit関数を定義しておくと、IBM iに値を送信（submit）する直前のタイミングにcansubmit関数が実行される。つまりページ表示時に、グラフを表示するには、initpage関数内でグラフ描画処理を記述することが必要だ。実装例が、【ソース13】である。

【ソース13】

(B) 要素の取得とグラフ描画の設定

関数の先頭部分でSP4i.getElementById メソッドを利用しているのがわかると思うが、このSP4i.getElementByIdメソッドは、引数に指定したid名を元に実行時に要素を取得する方法である。この方法は、SmartPad4iにおいてJavaScript側で処理を実行する際に多用するので、ぜひ覚えておこう。

では、ソース13の内容を確認していこう。まず初めに、①のように、SP4i.getElementById メソッドで年代別サブファイルのtable要素を取得する。

次に、②のように、サブファイルの情報をrowsプロパティとcellsプロパティを利用して配列に格納する。この際1行目は、列タイトルを表示しており、1行目を読み飛ばすためにforループの開始値は1と設定している点にご注意いただきたい。

グラフに必要なデータを配列に格納したら、あとはFlotr2を利用してグラフを出力すればよいのである。

出力には、Flotrオブジェクトのdrawメソッドを実行する。このdrawメソッドには、次の3つの引数がある。

1つ目には、図形を描画するdiv領域を指定する。今回はソース8で定義したgraphViewのid名をもつ要素を使用する。このgraphViewはSmartPad4iのDesignerで定義した項目ではないため、③のようにdocument.getElementByIdで要素を取得する必要がある。

2つ目には、グラフの元になる配列データをセットする。今回は、②の部分で取得した配列を④のようにdgAttArr変数として渡している。

3つ目は、描画するグラフ表示を設定するオプション設定である。⑤のように、横軸のラベルや、縦軸のラベル、グラフの形状などを設定すればよい。

今回のように円グラフを表示する場合、横軸ラベルや、縦軸ラベルは不要のため引数にfalseを設定している。

以上でプログラムは完了である。完成したプログラムを実行すると、集計された年代別の人数をもとに円グラフがきれいに表示されるのである。

図6　レスポンシブルデザイン

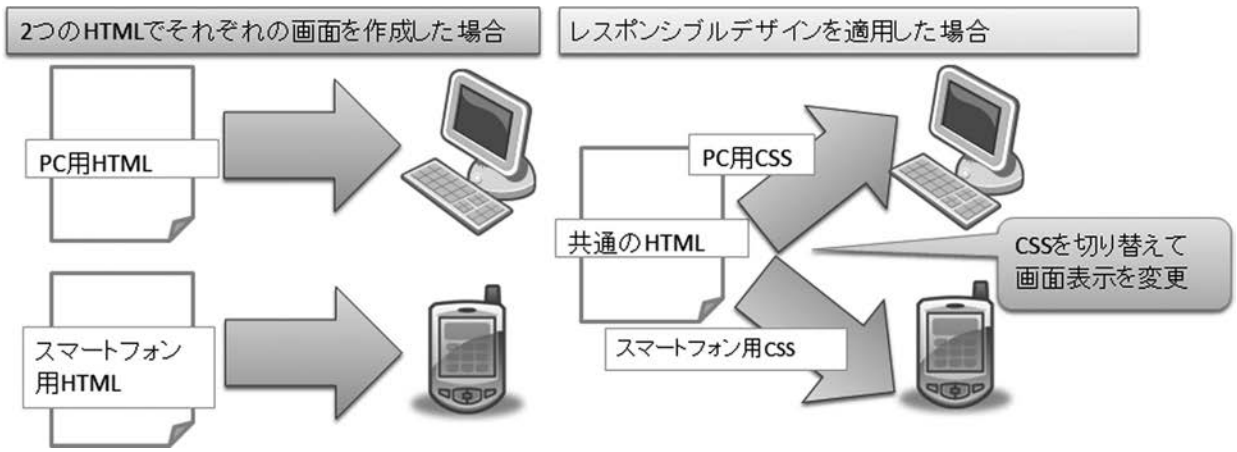


図7　レスポンシブルデザイン例

ソース5　レスポンシブルデザイン用CSS

```
@media
only screen and (max-width : 640px){
  /* デバイスの横幅が640以下の場合に適用するCSS を以下に記述する*/

  /* タイトルのフォントサイズ変更 */
  .Header h1{
    font-size:20px;
  }
  /* カナを表示しない */
  .name{
   display:none !important;
  }

  /* 区切り線を表示する */
  .mline{
     display:block;
  }
  /* 入力欄サイズを大きくする */
  .NNEN{
     width:360px !important;
  }
}
```

デバイスの横幅が640px以下の場合に適用するCSS

Flotr2について

なお、今回は円グラフの例を紹介したが、Flotr2では設定により、さまざまなグラフを出力することが可能である。さらに静的なグラフだけでなく、アニメーションを持つグラフも作成できるなど、非常に高機能なグラフ出力ライブラリである。いろいろと応用してみてほしい。Flotr2の公式サイトにはさまざまな例が記載されているので、ぜひ参考にしていただきたい。【図10】

6.最後に

本稿では、スマートデバイスに対応するWebインターフェースの工夫について紹介した。SmartPad4iではHTML5が利用できるため、リッチなインターフェースを簡単に作成できることをおわかりいただけたかと思う。

さらに、SmartPad4iは、Flotr2のような外部のライブラリと連携しやすい点も魅力の1つである、便利なライブラリが多数あるので、ぜひいろいろな連携に挑戦していただきたい。

M

ソース6　レスポンシブルデザイン用CSS　外部読み込み

```
<!-- saved from url=(0014)about:internet -->
<!DOCTYPE html>
<html>
<head>
<meta charset="Shift_JIS" />
<meta name="viewport" content="width=device-height" />
<meta name="format-detection" content="telephone=no" />
<link rel="stylesheet" href="DEMOCSS.css" type="text/css">
<link rel="stylesheet" href="DEMOCSS_mobile.css" type="text/css">
<title>MIGARO.Technical Report</title>
```

cssは最後に読み込まれるように追加

図8　グラフ完成画面

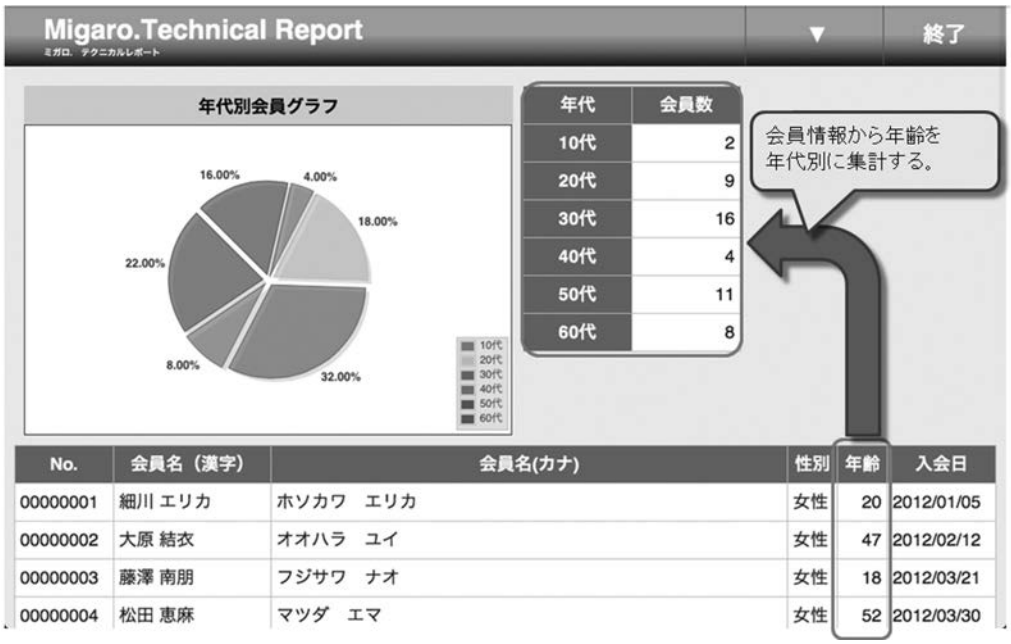
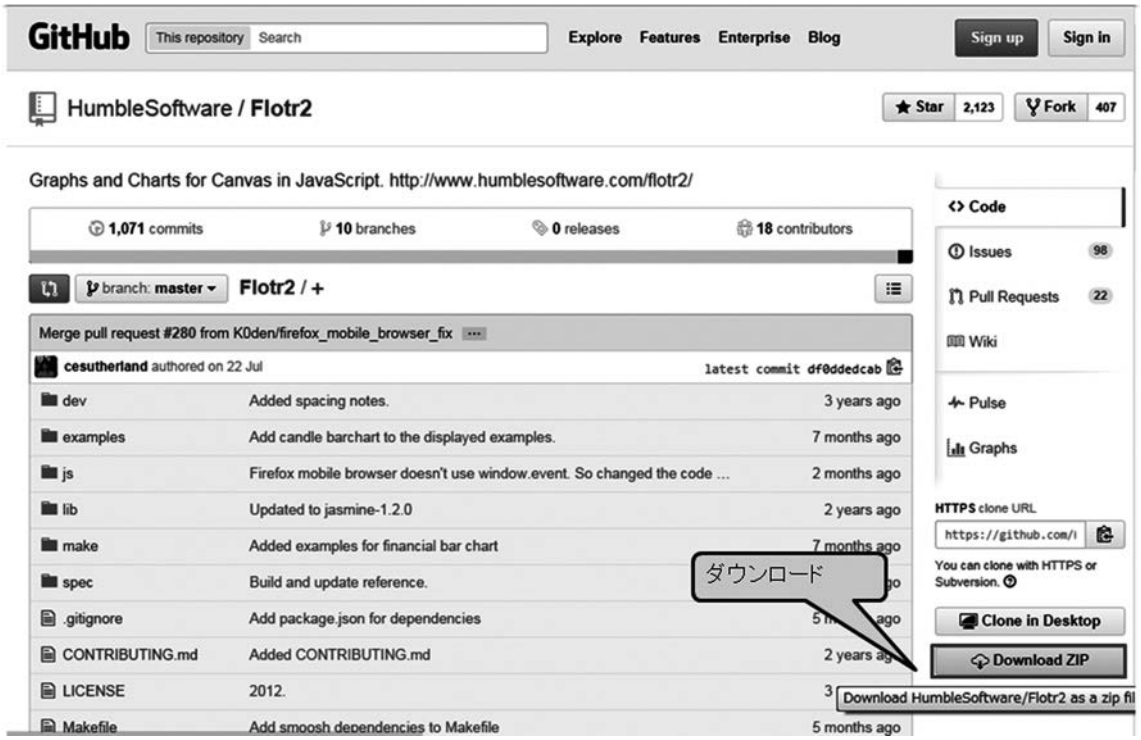


図9　flotr2 ダウンロード



ソース7　flotr2. min. jsの定義

```
<!-- saved from url=(0014)about:internet -->
<!DOCTYPE html>
<html>
<head>
<meta charset="Shift_JIS" />
<meta name="viewport" content="width=device-height" />
<meta name="format-detection" content="telephone=no" />
<link rel="stylesheet" href="DEMOCSS.css" type="text/css">
<title>MIGARO.Technical Report</title>
<script type="text/javascript" src="js/flotr2.min.js"></script>
```

外部JavaScriptファイルの参照追加

ソース8　グラフの表示域とグラフデータのサブファイルを定義

```
<!-- グラフの表示域 -->
<div id="gView" >
  <div id="graphTitle" class="graph-title"></div>
  <div id="graphView"></div>
</div>
<!-- グラフデータの表示域 -->
<div id="graphData">
  <table id="GDATA" class="PAGE" border="1px" cellspacing="0" cellpadding="5px">
    <thead>
      <th style="width:100px;">年代</th><th style="width:100px;">会員数</th>
    </thead>
    <tbody>
      <tr>
        <th id="GD1"></th><td id="GD2" style="text-align:right"></td>
      </tr>
    </tbody>
  </table>
</div>
<!-- 会員情報の表示域 -->
```

グラフを表示する領域

グラフデータを表示する領域

ソース9　RPG側　年代ラベルとデータの配列定義

```
0014.00   F010  * <YOURCODE>
0015.00   --->  * YOUR FILES
0016.00         F** <顧客ファイル>
0017.00         FCUSTFP  IF  E           K        DISK
0018.00   F010  * </YOURCODE>
0019.00         *
0020.00   E010  * <YOURCODE>
0021.00   --->  * YOUR ARRAYS
0022.00         E**<グラフラベル>
0023.00         E                    @GL1    1  10 30
0024.00         E**<グラフデータ>
0025.00         E                    @GD1    1  10  5
0026.00   E010  * </YOURCODE>
```

年代別のラベル

年代別の会員数を格納する配列

ソース10　RPG側　コンパイル時配列

```
0598.00    ** @GL1
0599.00    10代
0600.00    20代
0601.00    30代
0602.00    40代
0603.00    50代
0604.00    60代
0605.00    70代
0606.00    80代
0607.00    90代
0608.00    100代
0609.00    ** @GD1
0610.00    0
0611.00    0
0612.00    0
0613.00    0
0614.00    0
0615.00    0
0616.00    0
0617.00    0
0618.00    0
0619.00    0
```

ソース11　RPG側　ファイルの読み込み

```
0507.00      C**<会員ファイルの読み込み>
0508.00      C                   SETOF                     50
0509.00      C                   Z-ADD*ZERO     IND        40
0510.00      C                   CLEARSOOF02
0511.00      C          *LOVAL   SETLLCUSTFR
0512.00      C                   DO   *HIVAL
0513.00      C                   READ CUSTFR                81
0514.00      C          *IN81    IFEQ *ON
0515.00      C                   LEAVE
0516.00      C                   ENDIF
0517.00      C*
0518.00      C                   ADD  1         IND
0519.00      C          IND      OCUR SOOF03
0520.00      C                   EXSR RDFILE
0521.00      C                   ENDDO
0522.00      C                   Z-ADD1         JCL103  40
0523.00      C                   Z-ADDIND       JCL903  40
0524.00      C* <グラフ処理>
0525.00      C                   EXSR GRAPHD
```

ファイルの読み込み

データを出力するための
サブルーチン

グラフデータを出力するための
サブルーチン

ソース12　RPG側　会員データの表示

```
0550.00      *---------------------------------------------------------------
0551.00      * 顧客ファイル出力
0552.00      * 一覧に出力する項目を選択
0553.00      *---------------------------------------------------------------
0554.00     C           RDFILE    BEGSR
0555.00     C**<一覧出力>
0556.00     C                     MOVELCUSTNO    OLCNO             顧客番号
0557.00     C                     MOVELCUNAME    OLCNM1            名前
0558.00     C                     MOVELCUKANA    OLCNM2            カナ名
0559.00     C           CUSEX     IFEQ '0'
0560.00     C                     MOVEL'男性'    OLSEX             性別
0561.00     C                     ELSE
0562.00     C                     MOVEL'女性'    OLSEX
0563.00     C                     ENDIF
0564.00     C                     Z-ADDCUBIRT    OLBIRT            生年月日
0565.00     C                     Z-ADDCUENTD    OLENTD            入会日
0566.00     C**<一覧出力終了>
0567.00     C**<グラフ用データ取得>
0568.00     C           CUAGE     DIV  10       G       50        年代取得
0569.00     C           G         IFLE 10
0570.00     C           G         IFNE 0
0571.00     C                     MOVE @GD1,G    TGD1    50
0572.00     C                     ADD  1         TGD1              年代
0573.00     C                     MOVE TGD1      @GD1,G
0574.00     C                     ENDIF
0575.00     C                     ENDIF
0576.00     C*
0577.00     C**<グラフ用データ取得終了>
0578.00     C                     ENDSR
0579.00      *---------------------------------------------------------------
0580.00      * グラフデータ
0581.00      *
0582.00      *---------------------------------------------------------------
0583.00     C           GRAPHD    BEGSR
0584.00     C                     Z-ADD*ZERO     GDIND   40
0585.00     C                     Z-ADD*ZERO     I       40
0586.00     C           1         DO   10        I
0587.00     C           @GD1,I    IFNE '0'
0588.00     C                     ADD  1         GDIND
0589.00     C           GDIND     OCUR SOOFO2
0590.00     C                     MOVEL@GL1,I    OGD1      P       年代
0591.00     C                     MOVEL@GD1,I    OGD2      P       会員数
0592.00     C                     ENDIF
0593.00     C                     ENDDO
0594.00     C                     Z-ADD1         JCL102  40        FIRST LINE TO SEND
0595.00     C                     Z-ADDGDIND     JCL902  40        LAST LINE TO SEND
0596.00     C*
0597.00     C                     ENDSR
```

データを出力するための
サブルーチン

年代別の会員数を配列に格納

グラフデータを出力するための
サブルーチン

ソース13　initpage関数の定義

```
function initpage(){
  var gTable = SP4i.getElementById('GDATA'); //年代別のサブファイル取得
  document.getElementById("graphTitle").innerText= '年代別会員グラフ'; //タイトル
  var dgAttArr= new Array();
  for(var i=1; i<gTable.rows.length;i++){
    var slabel=gTable.rows[i].cells[0].innerText; //年代別ラベル
    var gdata=gTable.rows[i].cells[1].innerText;  //年代別会員数
    dgAttArr.push({data:[[0,parseInt(gdata)]], label:slabel }); //配列に追加
  }
  //グラフ描画
  Flotr.draw(document.getElementById("graphView"), dgAttArr
  , {
      HtmlText: false,
      grid: {
          verticalLines: false,      //canvas領域 縦グリッドの表示
          horizontalLines: false     //canvas領域 横グリッドの表示
      },
      xaxis: {
          showLabels: false          //横軸ラベル
      },
      yaxis: {
          showLabels: false          //縦軸ラベル
      },
      pie: {
          show: true                 //円グラフ
      },
      mouse: {
          track:false                //タッチイベントの取得
      },
      legend: {
          position: "se",            //凡例位置
          backgroundColor: "#D2E8FF" //背景色
      }
    });
}
```

①要素の取得

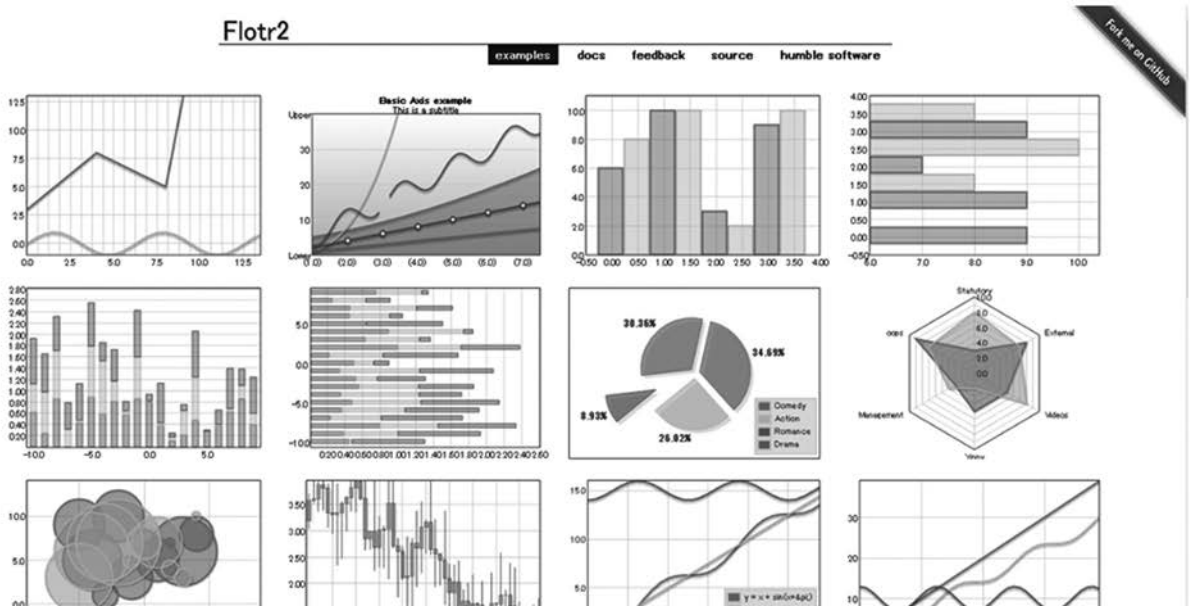
②年代別サブファイルの情報を配列へ格納

③id名graphViewのdiv要素を設定

④グラフ描画データ配列

⑤グラフ描画オプション

図10　flotr2 サンプルサイト



MIGARO.
TECHNICAL
REPORT

Migaro.Technical Report
No.7 2014 年秋
ミガロ.テクニカルレポート

2014 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