MIGARO. TECHNICAL REPORT

ミガロ.テクニカルレポート

No.13
2020年秋

株式会社ミガロ.

CONTENTS 目次

ごあいさつ

いつもミガロ.製品をご愛用いただき誠にありがとうございます。

「ミガロ.テクニカルレポート」は、Delphi/400、Valence、SP4i などのミガロ.製品を使った開発に関する技術情報をお届けする論文集で、このたび第 13 号を発刊する運びとなりました。

さて、2020 年の年頭から発生した新型コロナウイルスは、本号を作成している 2020 年秋の時点でも企業のビジネスや働き方に大きな影響を与え続けています。情報システム部門におかれては、急遽テレワーク環境の構築に取り組まれるなどご多忙を極めたお客様も多いと伺っております。このような状況のもと、今回のテクニカルレポートでは、お客様にご負担をおかけすることのないよう論文募集（ミガロ.テクニカルアワード）を取りやめ、ミガロ.SE 論文だけを掲載することにいたしました。ご理解とご了承を賜りますよう、お願い申し上げます。

変化の激しい近年のビジネス環境は、新型コロナウイルスの影響でさらに劇的な変化を迫られています。コロナ禍で既存のビジネスが厳しくなった代わりに、新規ビジネスに活路を見出された企業も多いのではないでしょうか。弊社では、ビジネスの急激な変化に対応するためには企業活動を支える IBM i 基幹システムも迅速に変化することが重要と考えており、超高速開発ツール Valence をはじめ、Delphi/400、SP4i などの開発ツールがそのお役に立てることに何よりも誇りと喜びを感じています。

今回の SE 論文は Delphi/400、SP4i、Valence のすべての製品をテーマとして含んでいます。Delphi/400 については、お客様からの関心が高いモバイル開発を 2 つの論文で取り上げました（Windows タブレットと iOS モバイル）。また、Delphi/400 の新機能を紹介する論文は開発効率アップのヒントとしてご活用ください。SP4i、Valence 論文は最新の製品情報を題材としました。SP4i は統合開発環境 Cobos と組み合わせることにより、すべての工程がオールインワンで開発可能となりました。Valence は新バージョンでセキュリティ向上、Web 版 5250 エミュレータとの融合、新規ウィジェット追加などの機能追加を行いました。これらについて今回の論文で詳しくご紹介しています。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。

弊社ではお客様向けのシステム構築やテクニカルサポートへのお問い合わせを通じて、多くの技術的な気づきやヒントをいただいております。今回テクニカルレポート第 13 号を発刊できたのは日頃のお客様からのご指導、ご鞭撻のたまものであり、この場をお借りしてあらためて厚く御礼申し上げます。

2020 年秋

株式会社ミガロ.
代表取締役社長
上甲 將隆

Migaro.
Technical Report
2020

SE 論文／ミガロ. テクニカルレポート

都地 奈津美

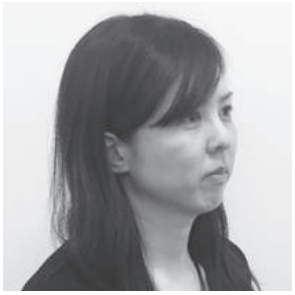
Delphi/400

株式会社ミガロ.

システム事業部 システム1課

[Delphi/400]

Windowsタブレット向けVCLアプリケーション作成テクニック



略歴
1989年8月19日生まれ
2012年3月 関西学院大学 理工学部卒業
2012年4月 株式会社ミガロ. 入社
2012年4月 システム事業部配属

現在の仕事内容
主にDelphi/400を使用したシステム受託開発とシステム保守を担当している。開発スキルの向上を目指し、日々精進している。

1. はじめに

近年、軽量で持ち運びやすいことやペーパーレス化の観点から、PCの代わりにタブレット端末を使用して業務を行うことが増えている。タブレット端末を導入する際、PCと同様のソフトウェアを利用できることや、PCと同じ操作感で使えることから、Windows OSを搭載したタブレットを選定する場合も少なくないだろう。

Windowsタブレット向けアプリケーションの場合、VCLフレームワークを使用することで、これまでのPC向けの開発と同様の方法でアプリケーションを開発できる。

しかしタブレットはPCに比べ、画面のサイズが小さいことや、マウス操作ではなくタッチ操作が主体となることなどから、PCとは違ったUI（ユーザーインターフェースの略で、ユーザーがPCとやり取りする際の入力や表示方法などの仕組みのこと）や、UX（ユーザーエクスペリエンスの略で、ユーザーがサービスを利用する際に体験したことや感じたこと）を考慮する必要がある。

そこで本稿では、Windowsタブレット向けVCLアプリケーションの作成テクニックとして、2.で日付と時間の入力方法について紹介する。次に3.では、複数端末でアプリケーションを使用する場合に、1つの画面設計から各端末サイズを基準に、項目の位置・サイズ自動調整する方法について紹介する。

2. UI／UXデザインを考慮した入力機能

2-1. アプリケーションの画面設計の違い

ここでは、タブレット向けアプリケーションを作成する際の入力方法について紹介する。

まず本題に入る前に、PC向けアプリケーションとタブレット向けアプリケーションの画面設計の違いについて説明する。

PCではマウス操作が主体となるのに対し、タブレットではタッチペンや指でのタッチ操作が主体となる。タッチ操作の場合、マウス操作とは異なり、狙ったポイントとはずれた位置をタッチしてしまうことがある。

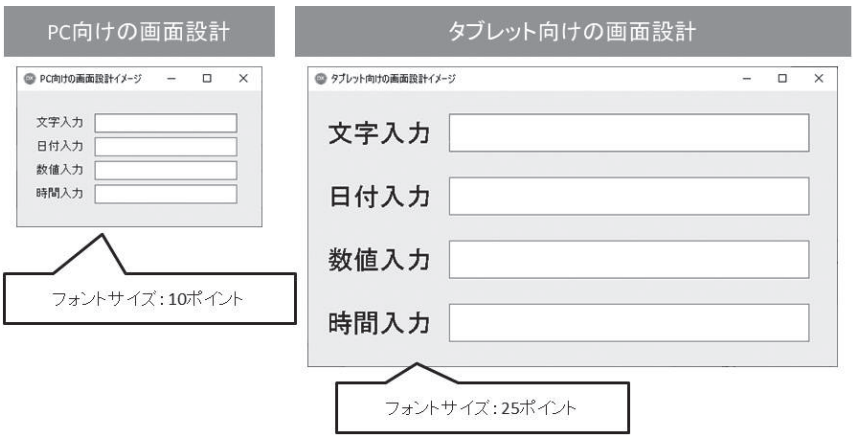
そのため入力項目やボタンのサイズ、フォントサイズは、PC向けアプリケーションに比べて大きく設計する必要がある。さらにタッチミスを考慮し、項目間の余白を広めにとることも必要である。【図1】

またタブレットは片手で操作する可能性が高いことから、キーボードでの入力をできるだけ少なくし、コンボボックスやチェックボックスなど直感的に入力できるよう工夫する必要がある。

今回は項目の入力方法について注目し、その中でも日付や時間を入力する方法について説明していく。日付や時間をソフトウェアキーボードで入力するには、ソフトウェアキーボードを起動したり、手入力で1文字ずつ入力する必要があり、入力に時間がかかる。

そこでカレンダーや時計を使用して、直感的かつ簡単に日付や時間を入力する

図1　PCとタブレットの画面設計の違い



＜タブレットアプリ設計時のポイント＞
①項目自体のサイズやフォントサイズを大きめに設計する
②項目間の余白を広めにとる

図2　日付のカレンダー入力

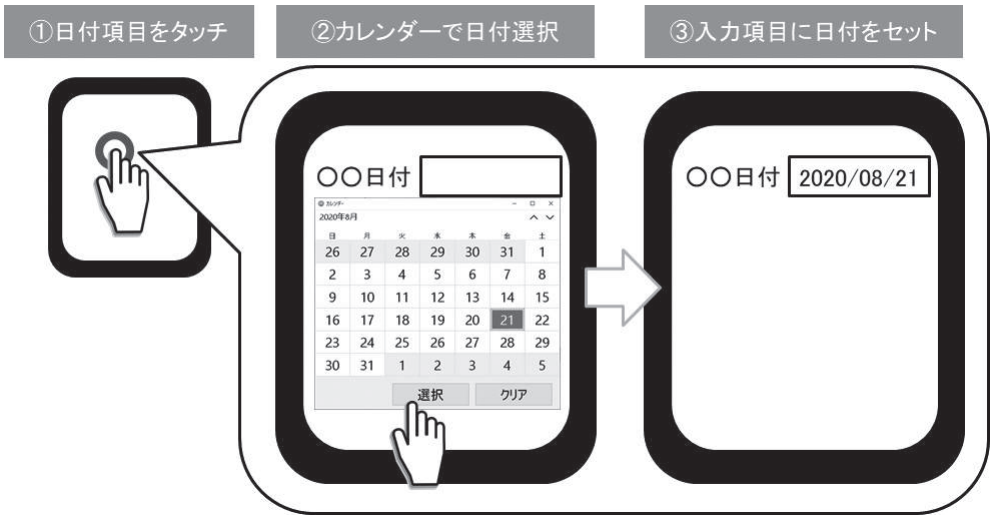
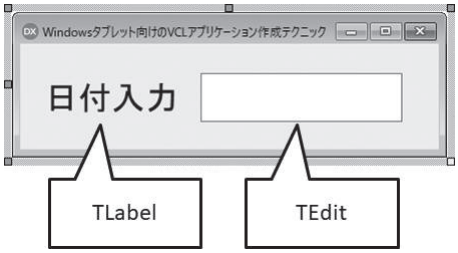


図3　コンポーネントの配置



方法について紹介する。カスタムソフトウェアキーボードを使用した文字入力や、テンキーを使用した数値入力については、2015 年発行のミガロ .テクニカルレポート 2015 にある『Windows タブレット用にカスタムソフトウェアキーボードを実装』に紹介されているので、こちらを参考にしていただきたい。

なお本章で作成しているプログラムは、Delphi/400 10.2Tokyo を使用している。

2-2. 日付のカレンダー入力

日付入力が必要な項目に対し、カレンダーを使用した入力方法の実装手順について説明する【図 2】。まずは、メインの入力画面について実装していく。

(1) コンポーネントの配置

日付入力を行うための TEdit、並びに TLabel を画面に配置する。【図 3】

(2) プロパティの設定

日付入力用の TEdit は、キーボードでの入力を禁止し、カレンダーでの入力のみを許可するため、ReadOnly プロパティに True を設定する。

次にカレンダー画面について、メイン画面とは別に新規フォームを作成し実装していく。

(1) コンポーネントの配置

カレンダー表示用の TCalendarView、日付選択用、選択値クリア用の TButton を配置する。【図 4】

(2) 処理の実装

①カレンダー画面の onShow イベントの実装

メイン画面の日付入力 Edit 用のプロパティを宣言し【ソース 1】、フォームの onCreate イベントを【ソース 2】のように実装する。メイン画面より日付入力 Edit をプロパティとして受け取り、カレンダー画面の onShow イベントで日付入力 Edit の値をカレンダーにセット、日付入力 Edit がブランクの場合はシステム日付をセットする。

これによりシステム日付、もしくは日付入力 Edit の値がカレンダーにセットされた状態でカレンダー画面が表示される。

②カレンダー画面の選択ボタン onClick イベントの実装

選択ボタンをクリック時に、カレンダーで選択されている日付をプロパティの日付入力 Edit にセットする。選択日付をセット後、カレンダー画面を終了する。【ソース 3】

③カレンダー画面のカレンダー onDblClick イベントの実装

カレンダーをダブルクリック時に、選択ボタンクリック時の処理を呼び出す。【ソース 4】

④カレンダー画面のクリアボタン onClick イベントの実装

クリアボタンをクリック時に、メイン画面の日付入力 Edit の日付をクリアするため、プロパティの日付入力 Edit にブランクをセットする。【ソース 5】

⑤メイン画面の日付 Edit の onEnter イベント

メイン画面の日付入力 Edit にフォーカスがセットされたタイミングで、カレンダー画面を表示するため、onEnter イベントにてカレンダー画面の生成・表示処理を行う。その際、プロパティの受け渡しと、カレンダー画面の表示位置を調整する。【ソース 6】

以上で、カレンダーを使用した日付入力の実装は完了である。

2-3. 時間の時計入力

時間入力が必要な項目に対し、時計を使用した入力方法の実装手順について説明する【図 5】。まずは、メインの入力画面について実装していく。

(1) コンポーネントの配置

2-2 にて作成したサンプルプログラムに対し、時間入力を行うための TEdit、並びに TLabel を画面に配置する。【図 6】

(2) プロパティの設定

日付入力用の TEdit と同様、時間入力用の TEdit の ReadOnly プロパティに True を設定する。

次に時計画面について、メイン画面とは別に新規フォームを作成し実装していく。

(1) コンポーネントの配置

時計表示用の TListBox（時間用、分用の 2 種類）、時間選択用、選択値クリア用の TButton を配置する【図 7】。時計表示用のコンポーネントとして、ドロップダウンで時間を選択できる TTimePicker でも代用が可能である。

今回、TListBox を使用する理由としては、TTimePicker では 0:00 ～ 23:59 のみが入力可能であるのに対し、TListBox は選択内容をソース内で設定するため、作業時間の入力など 24:00 以降の入力も可能となることが挙げられる。

(2) 処理の実装

①時計画面の onShow イベントの実装

メイン画面の時間入力 Edit 用のプロパティを宣言し【ソース 7】、フォームの onCreate イベントを【ソース 8】のように実装する。メイン画面より時間入力 Edit をプロパティとして受け取り、時計画面の onShow イベントで時間入力 Edit の値を時計にセット、時間入力 Edit がブランクの場合はシステム時間をセットする。

これにより、システム時間もしくは時間入力 Edit の値が時計にセットされた状態で時計画面が表示される。

②時計画面の選択ボタン onClick イベントの実装

選択ボタンをクリック時に、時計で選択されている時間をプロパティの時間入力 Edit にセットする。選択時間をセット後、時計画面を終了する。【ソース 9】

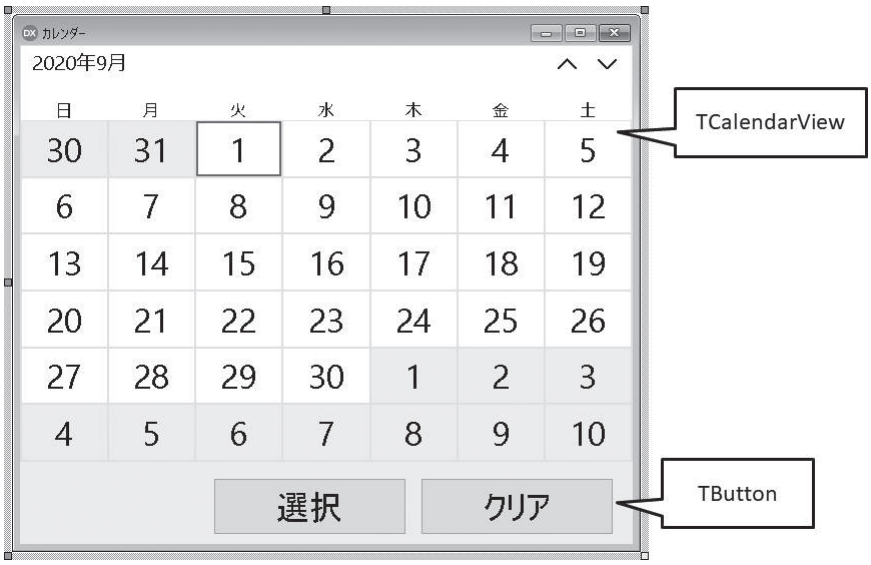
③時計画面のクリアボタン onClick イベントの実装

クリアボタンをクリック時に、メイン画面の時間入力 Edit の時間をクリアするため、プロパティの時間入力 Edit にブランクをセットする。【ソース 10】

④メイン画面の時間 Edit の onEnter イベント

メイン画面の時間入力 Edit にフォーカスがセットされたタイミングで、時計画面を表示するために、onEnter イベントにて時計画面の生成･表示処理を行う。

図4　コンポーネントの配置



ソース1

【カレンダー画面】グローバル変数

```
private
  { Private 宣言 }
  FDateEdit: TEdit;  // 日付入力Edit

public
  { Public 宣言 }
  property DateEdit: TEdit read FDateEdit write FDateEdit;  // 日付入力Edit
end;
```

メイン画面の日付入力Edit用のプロパティ

ソース2

【カレンダー画面】onShowイベント

```
procedure TfrmCalendar.FormShow(Sender: TObject);
var
  sDate: String;  // システム日付
begin
  // システム日付の取得
  sDate := FormatDateTime('YYYY/MM/DD', Now);

  // 遷移元画面の日付＝ブランクの場合、カレンダーにシステム日付をセット
  if (FDateEdit.Text = '') then
  begin
    CalendarView1.Date := StrToDate(sDate);
  end
  // 遷移元画面の日付≠ブランクの場合、カレンダーに遷移元画面の日付をセット
  else
  begin
    CalendarView1.Date := StrToDate(FDateEdit.Text);
  end;
end;
```

メイン画面の日付入力Editの日付 or
システム日付をカレンダーにセット

ソース3

【カレンダー画面】選択ボタンonClickイベント

```
procedure TfrmCalendar.Button1Click(Sender: TObject);
begin
  // 遷移元画面の日付に選択値をセット
  FDateEdit.Text := FormatDateTime('YYYY/MM/DD', CalendarView1.Date);

  // 画面を終了する
  Self.Close;
end;
```

その際、プロパティの受け渡しと、時計画面の表示位置を調整する。【ソース 11】

以上で、時計を使用した時間入力の実装は完了である。

2-4. 動作比較

日付、時間の入力について、本章のサンプルプログラムで入力した場合と、ソフトウェアキーボードより入力した場合を比較して確認してみる。

ソフトウェアキーボードから入力した場合、日付または時間を 1 文字ずつ入力、もしくはクリアする必要がある。また、日付や時間でない値を誤入力する可能性もある。【図 8 ～ 9】

それに対しサンプルプログラムのカレンダー、時計より入力する場合、入力値を選択するため誤入力もなく、簡単に入力・クリアしていると確認できる。【図 10 ～ 12】

またカレンダー、時計での入力のほうが、ソフトウェアキーボードでの入力よりも入力完了までのステップを少なく済ませられる。

3. 画面サイズに合わせた項目の自動調整

3-1. 複数端末でのアプリケーション使用時の課題

Windows タブレット向け VCL アプリケーションを作成する場合、Windows OS であれば特定機種のタブレット端末だけではなく、複数機種のタブレット端末や PC でもアプリケーションを使用できる。

ただし複数機種で 1 つのアプリケーションを使用する場合、画面サイズの違いから項目の位置やサイズが課題となるだろう。それぞれの画面サイズにそれほど差がない場合は問題ないが、そうでない場合は使用する端末に合わせてそれぞれの画面を設計する必要がある。また使用する機種が増えた場合、その画面に合わせて新たに画面を設計せねばならない。【図 13】

そこで 1 つのアプリケーションを複数機種で使用する場合、それぞれの機種の画面サイズに合わせて項目の位置やサイズ等を自動調整することで、使用する機種ごとに画面を設計する必要がなくなる。

3-2. 項目の位置・サイズの自動調整

項目の位置・サイズを自動調整するプログラムを作成していく。今回のサンプルプログラムでは、メインで使用する端末を画面解像度 1800 × 1200 のタブレット端末で縦向き使用とし、タブレット端末以外に画面解像度 2560 × 1440 の PC を使用すると想定する。

(1) 画面サイズの設定

画面設計については、メインで使用するタブレット端末の画面を基準に設計していく。そのため、タブレット端末の画面解像度に合わせて画面サイズを Height プロパティ = 1800、Width プロパティ = 1200 とする。

フォームの BorderStyle プロパティが bsSizeable（デフォルト値）の場合、画面サイズが一定サイズ以上大きく、もしくは小さくできないため、bsSizeable 以外に設定する必要がある。今回は bsSingle に設定する。【図 14】

今回はメイン端末をタブレット端末と想定しているため、タブレット端末を基準として画面サイズを設定したが、PC がメイン端末の場合、PC を基準に画面サイズを設定する。

開発環境により、1800 × 1200 では画面サイズが大きく画面設計しにくい場合、縦横比が画面解像度と同様（今回の場合は 3 : 2）となるように調整する。

後続の処理にて画面サイズを調整するため、画面解像度と同サイズでなくても縦横比が崩れてさえいなければサイズ調整しても問題ない。今回のサンプルプログラムでは、900 × 600 にサイズ調整している。

(2) コンポーネントの配置

今回のサンプルプログラムでは、商品検索の画面を想定してコンポーネントを配置する。画面上部に TPanel、TPanel 上に検索条件用の TComboBox、TEdit、検索ボタン用の TButton 並びに TLabel、画面下部に検索結果表示用の TStringGrid を配置する。【図 15】

各コンポーネントは、ParentFont プロパティを False に設定しておく。ParentFont プロパティが True の場合、フォームのフォントサイズを自動調整する際に、各コンポーネントのサイズがフォントサイズに合わせて自動で変更されてしまう。

今回はメイン端末をタブレット端末と想定しているため、2-1 に記載した項目のサイズ、フォントサイズを大きく設計し、項目間の余白を広めにとっている。

(3) 処理の実装

①画面の onCreate イベントの実装

画面起動時に、画面項目のサイズ・位置を調整する【ソース 12】。サイズ・位置調整については、private 宣言部に function GetScale（AForm: TForm）: Currency、procedure MagnificationFrm（AForm: TForm; AScale: Currency）を追加し、追加した独自の関数・手続き内でサイズ・位置を調整する。処理内容の詳細は、以下の②～④に記載する。

②サイズ調整に必要な倍率の算出

まず端末の画面サイズと、アプリケーションの画面サイズをもとに、サイズ調整に使用する倍率を算出する。倍率については、端末の画面の高さを基準に算出し、算出した倍率で調整したアプリケーション画面の幅が端末の画面の幅を超える場合、再度端末の画面の幅を基準に倍率を算出する。【ソース 13】

③画面サイズに合わせた項目サイズ・位置の調整

次に、②で算出した倍率をもとに、画面サイズ、フォームのフォントサイズを調整する【ソース 14】。コンポーネントごとのサイズ、位置調整については、次の④に記載する。

④コンポーネントのサイズ・位置の調整

次にフォームの ComponentCount を取得し、フォーム上に配置されているすべてのコンポーネントに対してサイズ・位置を調整する。【ソース 15】

④-1. TEdit、TComboBox、TButton、TLabel の場合

②で算出した倍率をもとにフォントサイズ、項目のサイズ、項目の位置を調整する【ソース 16】。サンプルソースでは TEdit の場合のみを記載しているが、TComboBox、TButton、TLabel につ

ソース4

【カレンダー画面】カレンダーonDblClickイベント

```
procedure TfrmCalendar.CalendarView1DblClick(Sender: TObject);
begin
  // 選択ボタン押下時処理
  Button1Click(Button1);
end;
```

ソース5

【カレンダー画面】クリアボタンonClickイベント

```
procedure TfrmCalendar.Button2Click(Sender: TObject);
begin
  // 遷移元画面の日付をクリア
  FDateEdit.Text := '';

  // 画面を終了する
  Self.Close;
end;
```

ソース6

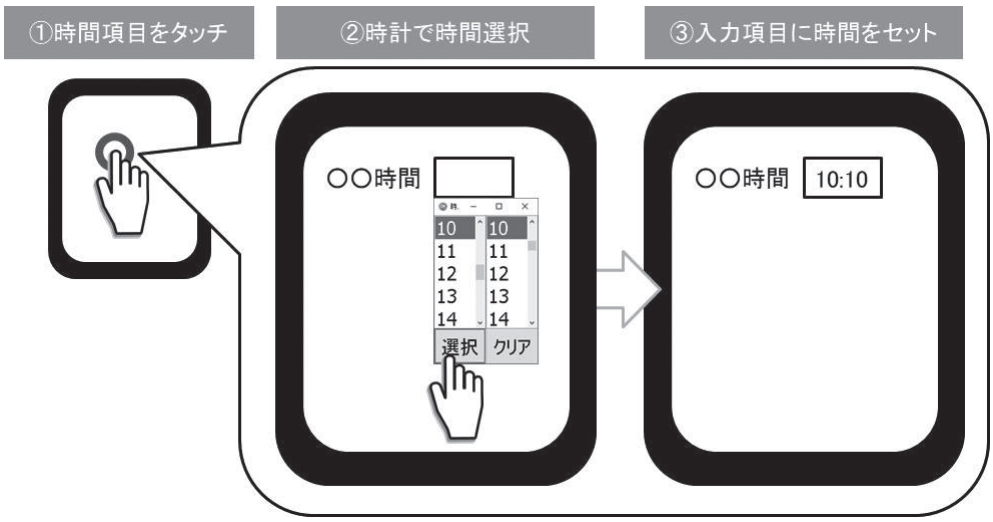
【メイン画面】日付EditのonEnterイベント

```
procedure TForm4.edtCalendarEnter(Sender: TObject);
begin
  // カレンダーフォームの起動
  frmCalendar := TfrmCalendar.Create(Self);

  // 日付入力Editのプロパティ
  frmCalendar.DateEdit := edtCalendar;

  // Left位置調整
  frmCalendar.Left := Self.Left                          // 画面のLeft位置
                    + edtCalendar.Left;                  // EditのLeft位置

  // Top位置調整
  frmCalendar.Top := GetSystemMetrics(SM_CYCAPTION)      // タイトルの高さ
                   + Self.Top                            // 画面のTop位置
                   + edtCalendar.Top                     // EditのTop位置
                   + edtCalendar.Height;                 // Editの高さ

  // カレンダーフォームの表示
  frmCalendar.Show;
end;
```

カレンダー画面のプロパティに日付入力Editをセットする

カレンダー画面の表示位置の調整

図5　時間の時計入力



いても同様に設定する。

④-2. TPanel の場合

④-1 と同様、フォントサイズ、項目のサイズ、位置を調整する。TPanel の場合、Align プロパティ = alTop、alBottom、alClient の際、項目幅は画面幅と同値になるため、Width プロパティの設定は不要である。

同様に、Align プロパティ = alRight、alLeft、alClient の場合、項目の高さは画面の高さと同値になるため、Height プロパティの設定は不要である。

さらに、Align プロパティ = alNone、alCustom 以外の場合、項目の位置は画面の空き領域の左端・上端になるため、Left プロパティ、Top プロパティの設定は不要である。【ソース 17】

④-3. TStringGrid の場合

④-1 と同様、フォントサイズ、項目のサイズ、位置を調整する。TStringGrid の場合、項目のサイズ調整のみでなく、明細の行列情報のサイズ調整も必要である。

設定が必要なプロパティは DefaultColWidth プロパティ、DefaultRowHeight プロパティ、ColWidths プロパティ、RowHeights プロパティであり、基本的にはフォントサイズ等と同様に調整する。

ただし、ここで 1 つ注意点がある。ColWidths プロパティと RowHeights プロパティは、DefaultColWidth プロパティと DefaultRowHeight プロパティの変更値に合わせて自動設定される。そのため、いったん値を内部保持したあと、再セットし直す必要がある。【ソース 18】

以上で、項目の位置・サイズの自動調整の実装は完了である。

上記プログラムを実行した場合と、自動調整を行わずにそのままの状態で表示した場合とを比較して確認してみよう。

自動調整を行わないアプリケーションを PC で実行時、表示スケール（Windows 10 のディスプレイの設定における拡大縮小の設定）100％の場合、アプリケーションが PC 画面内に収まって表示される。

しかし今回の PC の解像度では、表示スケール 200％がデフォルト設定（推奨）されており、その場合アプリケーションの一部が PC 画面内に表示されない。【図 16】

またタブレット端末で実行時、画面解像度の 1/2 のサイズで設計したため、タブレット画面に対して小さく表示される。【図 17】

それに対してサンプルプログラムの項目の場合、端末の画面サイズに対してアプリケーションのサイズ、各項目のサイズ、位置、フォントサイズを自動調整できていることが確認できる。

また PC の場合、自動調整前は表示スケールごとにアプリケーションのサイズが異なっていたが、自動調整により表示スケールに依存せず自動調整できている。【図 18 ～ 19】

4. さいごに

本稿では、Windows タブレット向け VCL アプリケーションを作成する際の作成テクニックについて紹介してきた。

PC の代わりにタブレット端末をビジネスで活用する場面が増えたことで、Windows タブレット向けアプリケーションを開発する局面も増えているだろう。VCL フレームワークを選択することで、これまでと同様の手法での開発が可能となる反面、本稿で取り上げたような PC とは操作方法が異なる点や、PC とタブレット端末双方の使用を考慮することなど、UI ／ UX を考慮して、いかにユーザーが見やすく、使いやすいアプリケーションを作成するかが重要になる。

Windows タブレット向け VCL アプリケーションを作成時に、本稿で紹介した開発テクニックを役立てていただければ幸いである。

M

図6　コンポーネントの配置

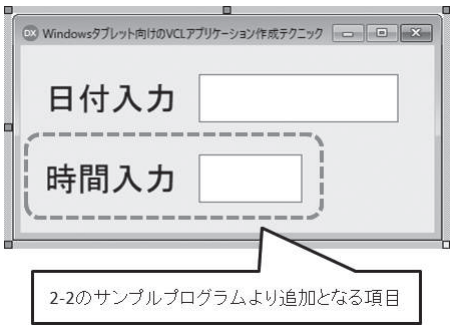
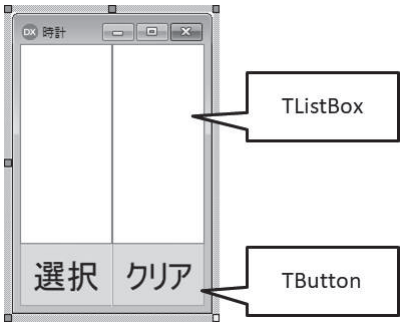


図7　コンポーネントの配置



ソース7

【時間画面】グローバル変数

```
private
  { Private 宣言 }
  FTimeEdit: TEdit;   // 時間入力Edit

public
  { Public 宣言 }
  property TimeEdit: TEdit read FTimeEdit write FTimeEdit;   // 時間入力Edit
end;
```

メイン画面の時間入力Edit用のプロパティ

ソース8

【時間画面】onShowイベント

```
procedure TfrmTime.FormShow(Sender: TObject);
var
  i: Integer;
  sTime: String;  // システム時間
begin
  // 初期化
  ListBox1.Items.Clear;  // 時間
  ListBox2.Items.Clear;  // 分

  // 時間のセット
  for i := 0 to 23 do
  begin
    ListBox1.Items.Add(FormatFloat('00', i));
  end;

  // 分のセット
  for i := 0 to 59 do
  begin
    ListBox2.Items.Add(FormatFloat('00', i));
  end;

  // システム時間の取得
  sTime := FormatDateTime('HH:MM', Now);

  // 遷移元画面の時間＝ブランクの場合、時間にシステム時間をセット
  if (FTimeEdit.Text = '') then
  begin
    ListBox1.ItemIndex := ListBox1.Items.IndexOf(Copy(sTime, 1, 2));
    ListBox1.TopIndex  := ListBox1.ItemIndex;
    ListBox2.ItemIndex := ListBox2.Items.IndexOf(Copy(sTime, 4, 2));
    ListBox2.TopIndex  := ListBox2.ItemIndex;
  end
  // 遷移元画面の時間≠ブランクの場合、時間に遷移元画面の時間をセット
  else
  begin
    ListBox1.ItemIndex := ListBox1.Items.IndexOf(Copy(FTimeEdit.Text, 1, 2));
    ListBox1.TopIndex  := ListBox1.ItemIndex;
    ListBox2.ItemIndex := ListBox2.Items.IndexOf(Copy(FTimeEdit.Text, 4, 2));
    ListBox2.TopIndex  := ListBox2.ItemIndex;
  end;
end;
```

時間と分のTListBoxにそれぞれの選択値をセット

メイン画面の時間入力Editの時間
orシステム時間を時計にセット

ソース9

【時計画面】選択ボタンonClickイベント

```
procedure TfrmTime.Button1Click(Sender: TObject);
begin
  // 選択時間をセット
  FTimeEdit.Text := ListBox1.Items[ListBox1.ItemIndex]   // 時間
                  + ':'
                  + ListBox2.Items[ListBox2.ItemIndex];  // 分

  // 画面を終了する
  Self.Close;
end;
```

ソース10

【時計画面】クリアボタンonClickイベント

```
procedure TfrmTime.Button2Click(Sender: TObject);
begin
  // 遷移元画面の時間をクリア
  FTimeEdit.Text := '';

  // 画面を終了する
  Self.Close;
end;
```

ソース11

【メイン画面】時間EditのonEnterイベント

```
procedure TForm4.edtTimeEnter(Sender: TObject);
begin
  // 時間フォームの起動
  frmTime := TfrmTime.Create(Self);

  // 時間入力Editのプロパティ
  frmTime.TimeEdit := edtTime;

  // Left位置調整
  frmTime.Left := Self.Left                          // 画面のLeft位置
                + edtTime.Left;                      // EditのLeft位置

  // Top位置調整
  frmTime.Top := GetSystemMetrics(SM_CYCAPTION)      // タイトルの高さ
               + Self.Top                            // 画面のTop位置
               + edtTime.Top                         // EditのTop位置
               + edtTime.Height;                     // Editの高さ

  // 時間フォームの表示
  frmTime.Show;
end;
```

時計画面のプロパティに時間入力Editをセットする

時計画面の
表示位置の調整

図8　ソフトウェアキーボードを使用した日付入力

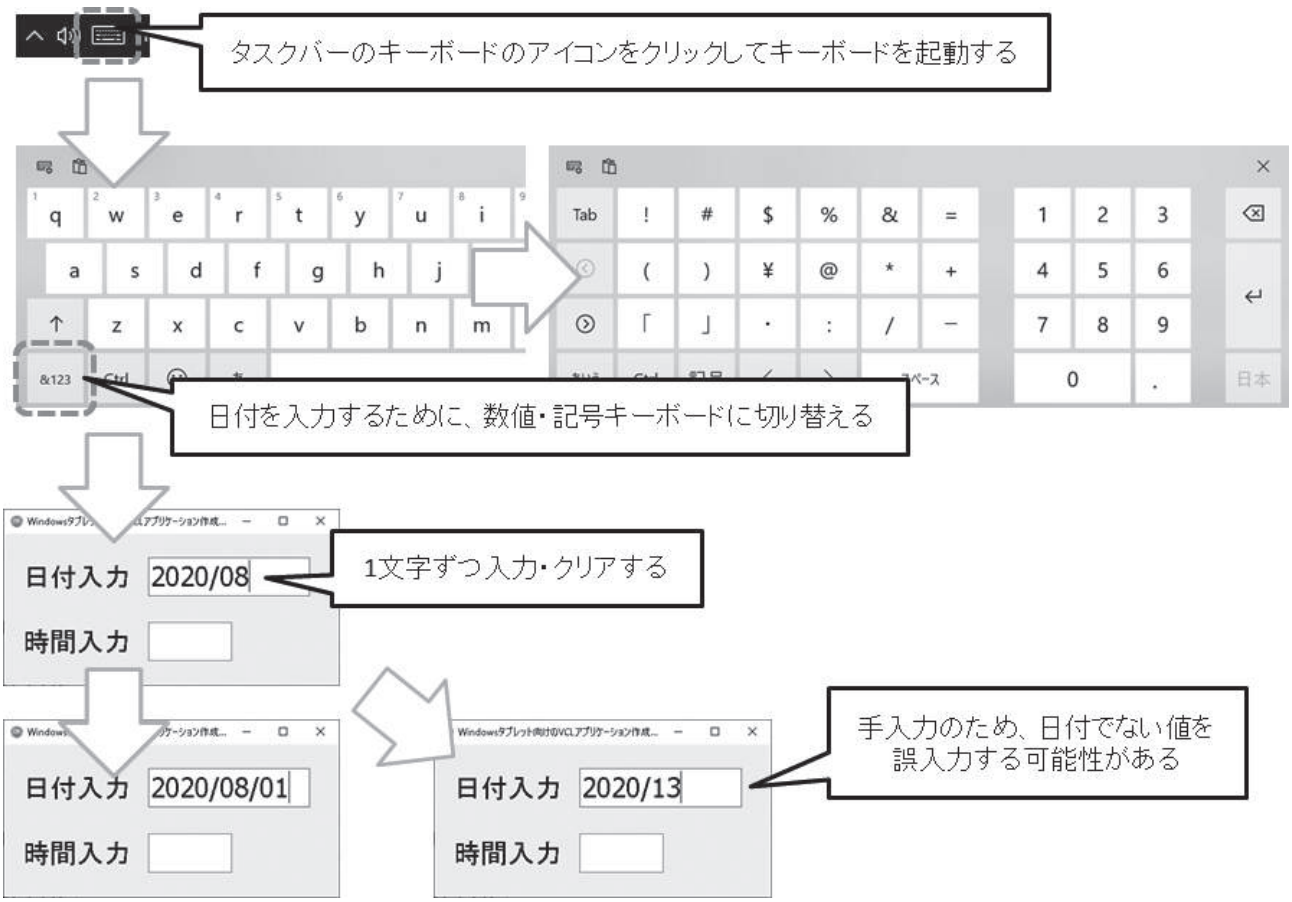


図9　ソフトウェアキーボードを使用した時間入力

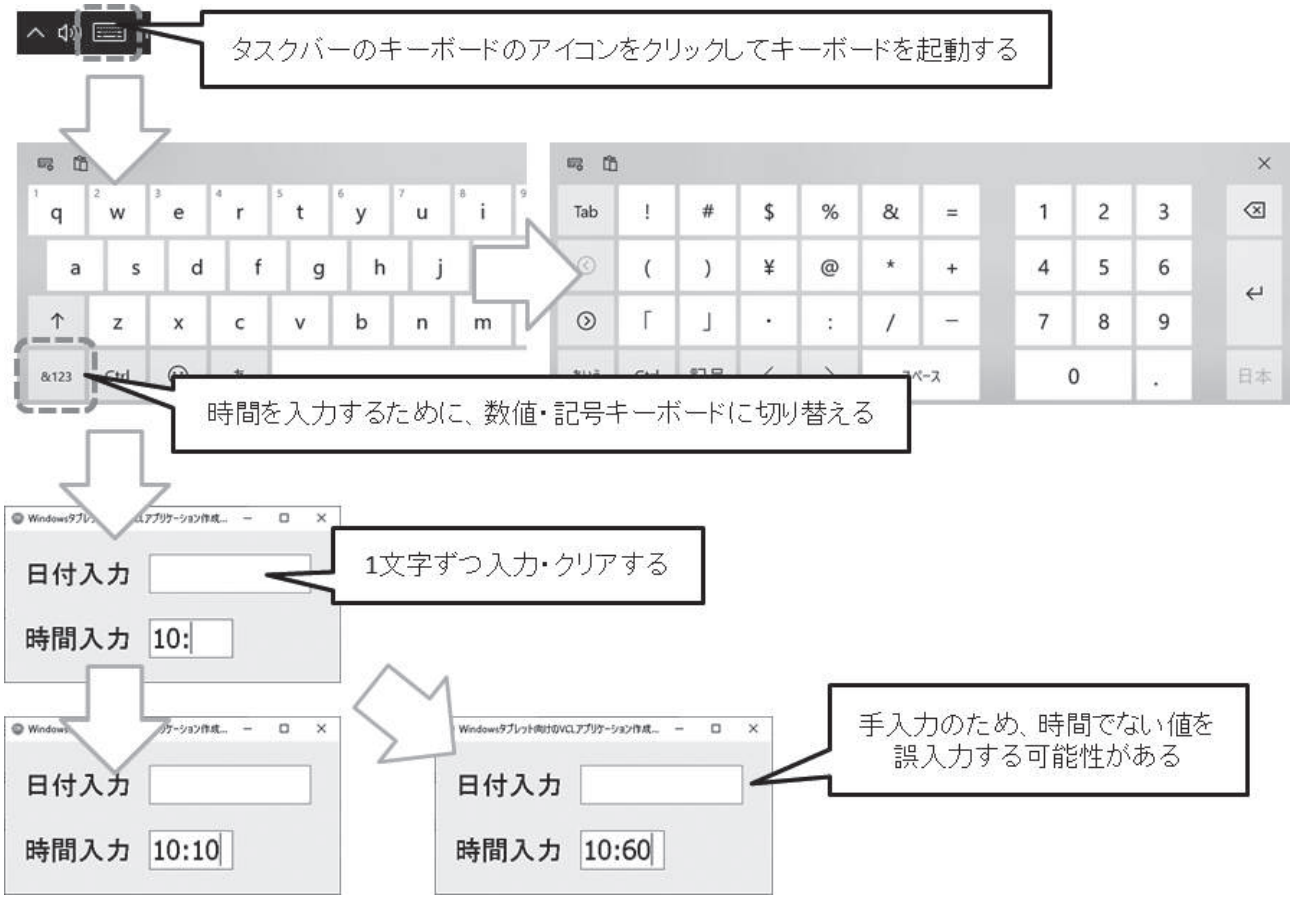


図10　カレンダー画面を使用した日付入力①

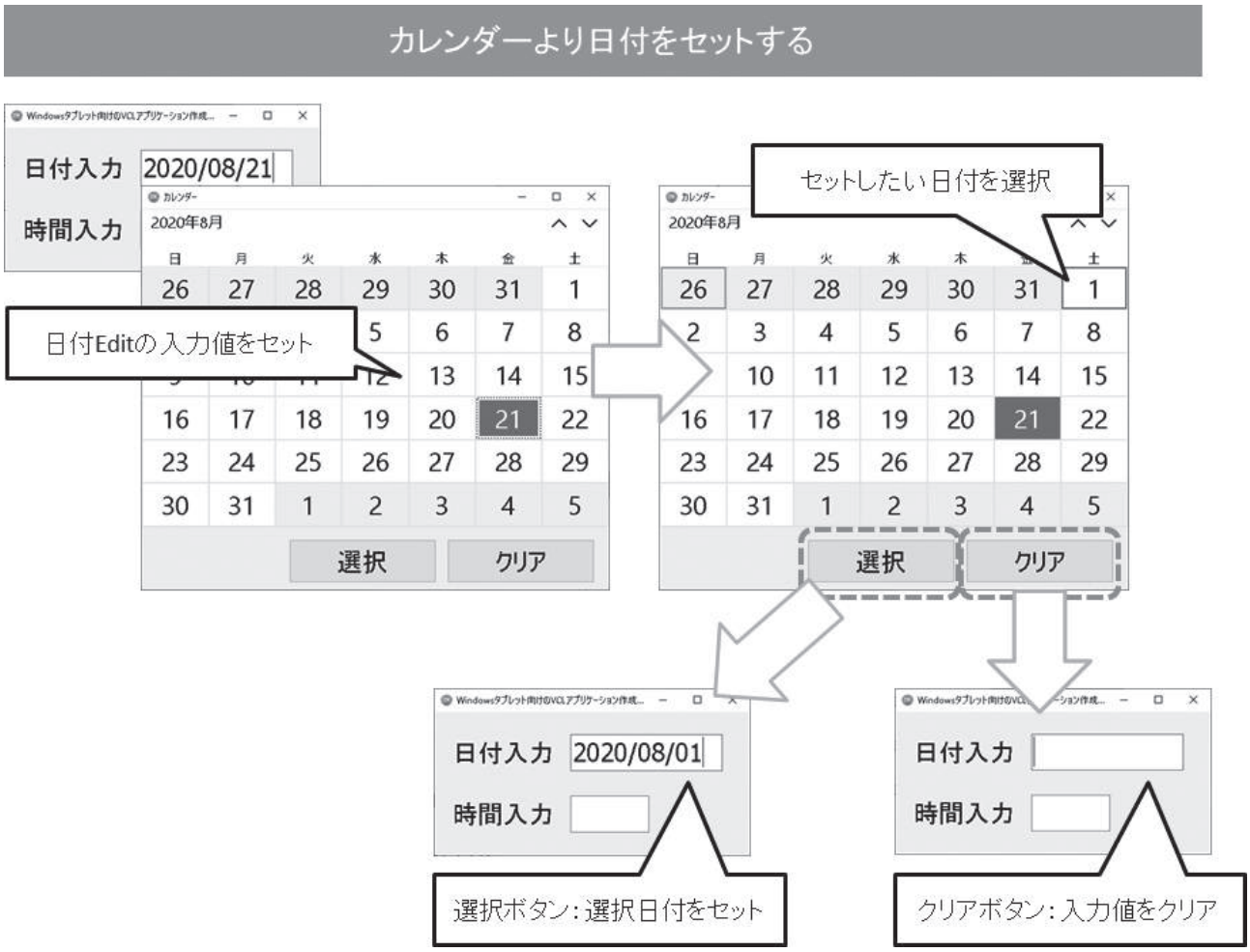


図11　カレンダー画面を使用した日付入力②

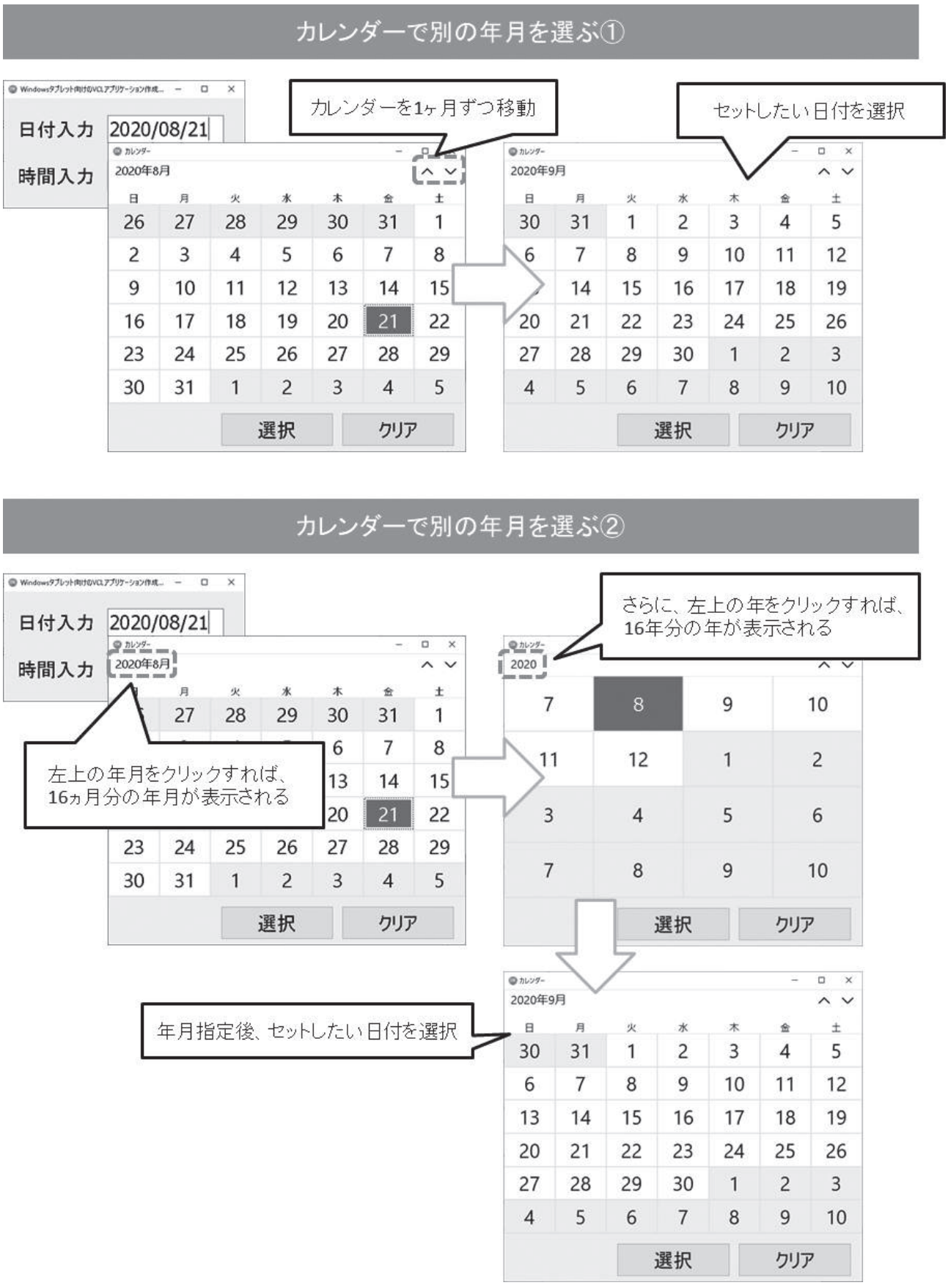


図12　時計画面を使用した時間入力

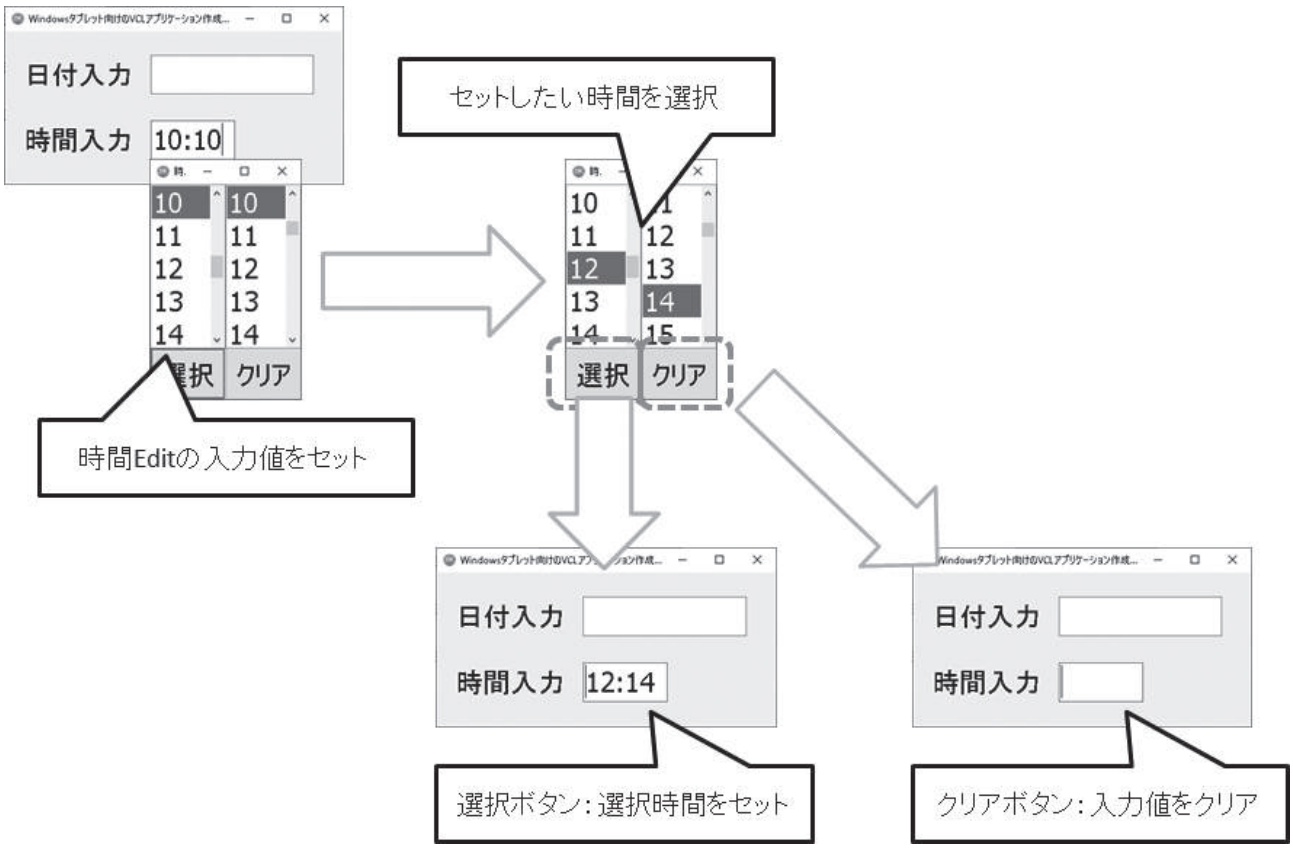


図13　アプリケーションを複数で使用する場合の画面設計

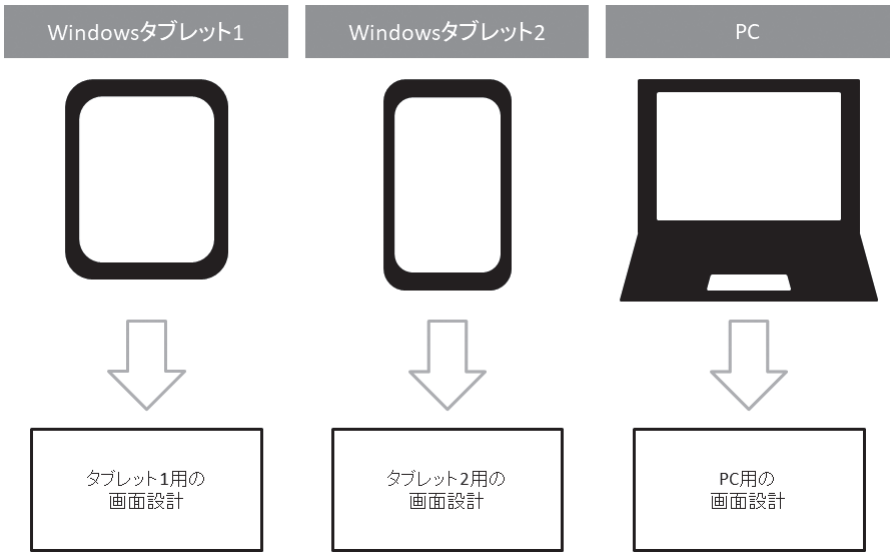


図14　画面サイズの設定

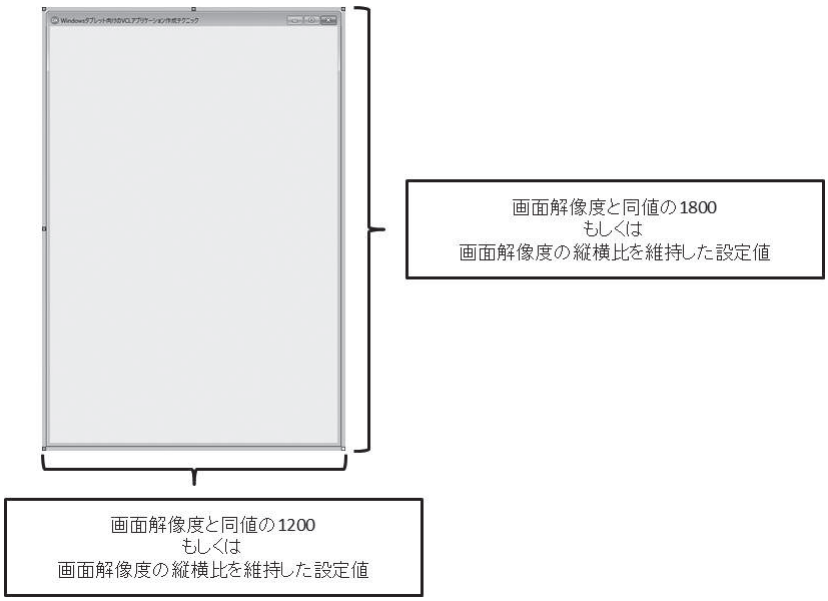
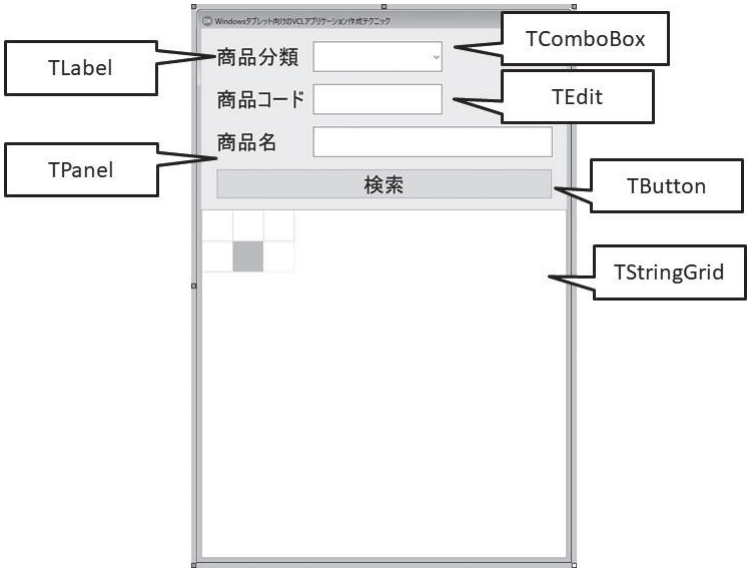


図15　コンポーネントの配置



ソース12

画面のonCreateイベント

```
procedure TForm1.FormCreate(Sender: TObject);
var
  cScale: Currency;  // 倍率
begin
  // 倍率を取得
  cScale := GetScale(Self);

  // 画面等倍処理
  MagnificationFrm(Self, cScale);
end;
```

ソース13

ソース14

ソース13

サイズ調整に必要な倍率の算出

```
function TForm1.GetScale(AForm: TForm): Currency;
var
  cScale: Currency;
begin
  // 画面の高さを基準に算出
  cScale := Screen.Height / 900;

  // 調整した幅が画面幅を超えている場合、画面の幅を基準に再度算出
  if (Trunc(AForm.Width * cScale) > Screen.Width) then
  begin
    cScale := Screen.Width / 600;
  end;

  // 算出結果を返却
  Result := cScale;
end;
```

ソース14

画面サイズに合わせた項目サイズ・位置の調整

```
procedure TForm1.MagnificationFrm(AForm: TForm; AScale: Currency);
var
  i: Integer;
begin
  // 倍率＝1の場合、処理終了
  if (AScale = 1) then
  begin
    Exit;
  end;

  // 画面大きさ制御
  if (Screen.Width <> AForm.Width) then
  begin
    // 大きさ変更
    for i := 0 to AForm.ComponentCount - 1 do
    begin
      ComponentScale(AForm, AForm.Components[i], AScale);
    end;

    // 画面サイズ
    AForm.Width  := Trunc(AForm.Width  * AScale);
    AForm.Height := Trunc(AForm.Height * AScale);

    // フォントサイズ
    AForm.Font.Size := Trunc(AForm.Font.Size * AScale);
  end;
end;
```

ソース15

ソース15

コンポーネント毎のサイズ・位置の調整①メイン処理

```
procedure TForm1.ComponentScale(AForm: TForm; AComponent: TComponent;
  AScale: Currency);
var
  i: Integer;
  slGridColWidth: TStringList;    // TStringGridの明細列幅内部保持用
  slGridRowHeight: TStringList;   // TStringGridの明細行の高さ内部保持用
begin
  // すでに変更されている場合、処理しない
  if (AComponent.Owner is TForm) and
     (Screen.Width = TForm(AComponent.Owner).Width) then
  begin
    Exit;
  end;
```

＜コンポーネント毎のサイズ・位置の調整＞
ソース16～19

```
end;
```

ソース16

コンポーネント毎のサイズ・位置の調整②TEdit、TComboBox、Tbutton、TLabel

```
// TEdit
if (AComponent is TEdit) then
begin
  with TEdit(AComponent) do
  begin
    // フォントサイズ
    Font.Size := Trunc(Font.Size * AScale);

    // サイズ
    Width  := Trunc(Width  * AScale);  // 幅
    Height := Trunc(Height * AScale);  // 高さ

    // 位置
    Left := Trunc(Left * AScale);  // Left位置
    Top  := Trunc(Top  * AScale);  // Top位置
  end;
end
else
```

コンポーネント毎のサイズ・位置の調整③TPanel

```
// TPanel
if (AComponent is TPanel) then
begin
  if (TPanel(AComponent).Align <> alClient) then
  begin
    with TPanel(AComponent) do
    begin
      // フォントサイズ
      Font.Size := Trunc(Font.Size * AScale);

      // 幅
      if not (Align in [alTop, alBottom, alClient]) then
      begin
        Width := Trunc(Width  * AScale);
      end;

      // 高さ
      if not (Align in [alRight, alLeft, alClient]) then
      begin
        Height := Trunc(Height * AScale);
      end;

      // 表示位置を指定していない場合
      if (Align in [alNone, alCustom]) then
      begin
        Left := Trunc(Left * AScale);  // Left位置
        Top  := Trunc(Top  * AScale);  // Top位置
      end;
    end;
  end;
end
else
```

ソース18

コンポーネント毎のサイズ・位置の調整⑤TStringGrid

```
// TStringGrid
if (AComponent is TStringGrid) then
begin
  with TStringGrid(AComponent) do
  begin
    // フォントサイズ
    Font.Size := Trunc(Font.Size * AScale);

    // 明細自体のサイズ
    if (not AutoSize) then
    begin
      Width  := Trunc(Width  * AScale);  // 幅
      Height := Trunc(Height * AScale);  // 高さ
    end;

    // 位置
    Left := Trunc(Left * AScale);  // Left位置
    Top  := Trunc(Top  * AScale);  // Top位置

    // サイズ変更前の状態を内部保持用
    slGridColWidth  := TStringList.Create;  // 幅
    slGridRowHeight := TStringList.Create;  // 高さ

    try
      // 明細のデフォルトサイズを変更すると全て同一値に変更されるため
      // 変更前の幅と高さを内部保持する
      // 明細の列幅
      for i := 0 to ColCount - 1 do
      begin
        slGridColWidth.Add(IntToStr(ColWidths[i]));
      end;

      // 明細の行の高さ
      for i := 0 to RowCount - 1 do
      begin
        slGridRowHeight.Add(IntToStr(RowHeights[i]));
      end;

      // 明細のデフォルトサイズ
      DefaultColWidth  := Trunc(DefaultColWidth  * AScale);  // 幅
      DefaultRowHeight := Trunc(DefaultRowHeight * AScale);  // 高さ

      // 明細の列幅（内部保持値より変換）
      for i := 0 to slGridColWidth.Count - 1 do
      begin
        ColWidths[i] := Trunc(StrToInt(slGridColWidth[i]) * AScale);
      end;

      // 明細の行の高さ（内部保持値より変換）
      for i := 0 to slGridRowHeight.Count - 1 do
      begin
        RowHeights[i] := Trunc(StrToInt(slGridRowHeight[i]) * AScale);
      end;
    finally
      FreeAndNil(slGridColWidth);
      FreeAndNil(slGridRowHeight);
    end;
  end;
end;
```

ColWidthsプロパティ、RowHeightsプロパティの設定値を内部保持

DefaultColWidthプロパティ、DefaultRowHeightプロパティの調整

内部保持値を基にColWidthsプロパティ、RowHeightsプロパティを調整

図16　項目の自動調整を行わない場合（PC）

表示スケール100%

表示スケール200%

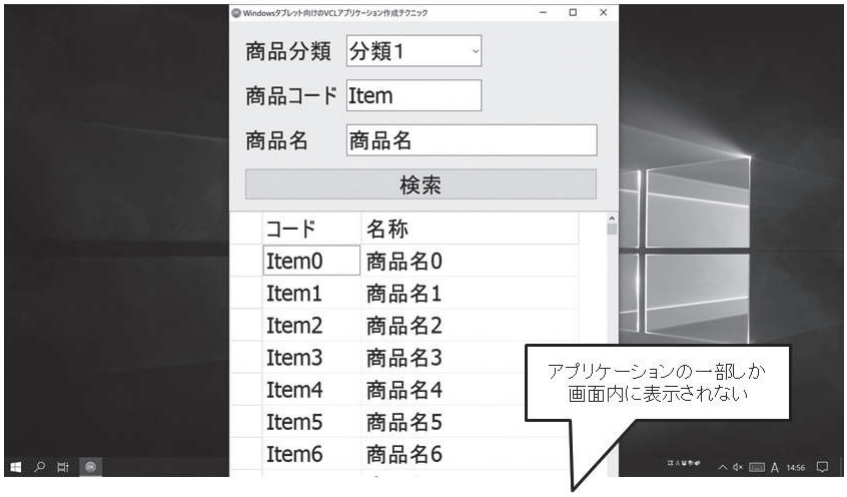


図17　項目の自動調整を行わない場合（タブレット）

図18　項目の自動調整を行った場合（PC）

表示スケール100%

表示スケール200%

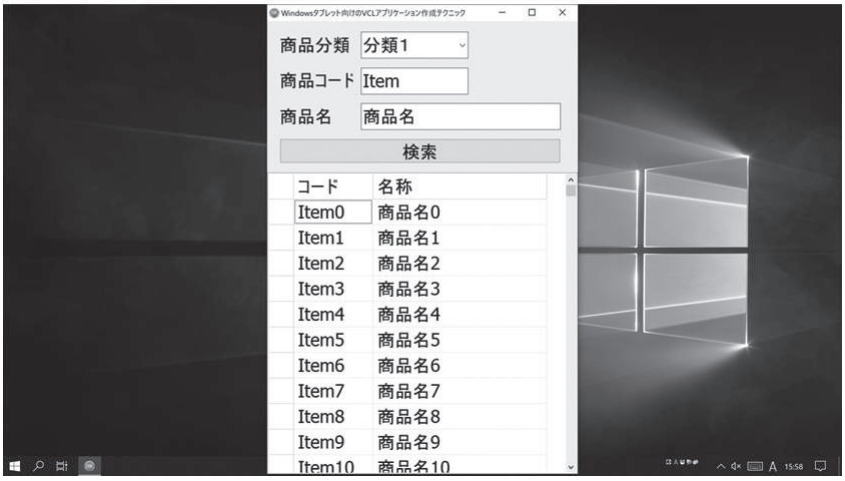
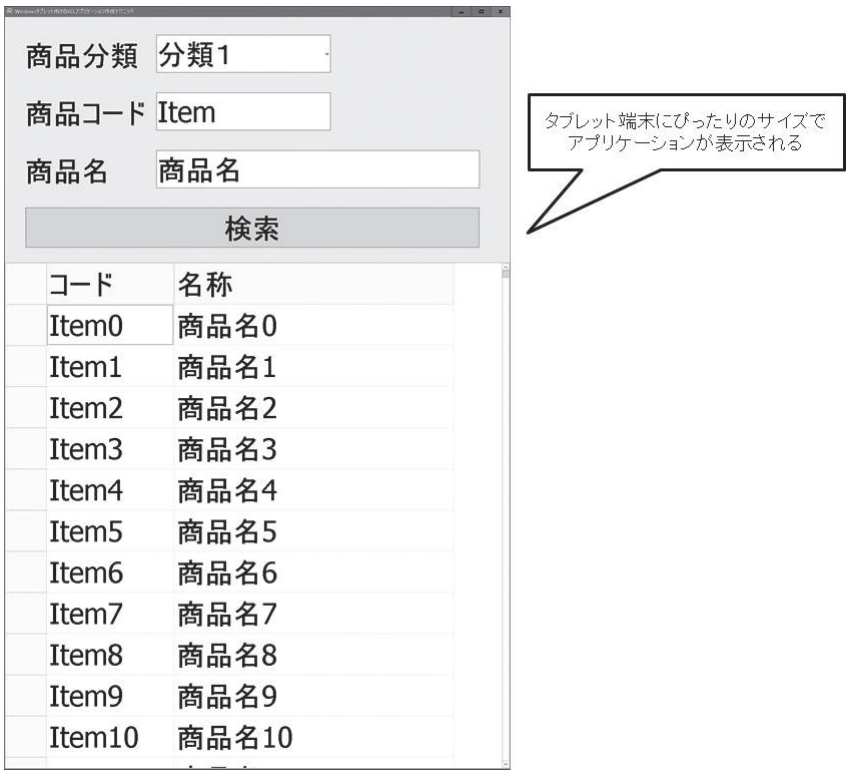


図19　項目の自動調整を行った場合（タブレット）



タブレット端末にぴったりのサイズで
アプリケーションが表示される

前坂 誠二

株式会社ミガロ.

システム事業部 システム2課

[Delphi/400] iOSモバイルアプリケーションによるファイル閲覧機能作成

略歴
1989 年 3 月 21 日生まれ
2011 年 3 月 関西大学 文学部卒業
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

近年、テレワークの導入や外出先で仕事をするといったように、場所にとらわれない働き方が増えている。これに伴って従来の PC 端末だけではなく、持ち運びに便利なタブレット端末を業務で使用する機会も増えている。

今はタブレット端末でも、社内のネットワークに接続すればファイルサーバー上のファイルを参照可能である。しかしタブレット端末を業務で使用する際に、「もっと効率よくファイルサーバー上の PDF ファイルや Excel ファイルなどを確認したい」と感じたことはないであろうか。

そこで本稿では、Delphi/400 10.2Tokyo を用いて、タブレット端末によるファイル閲覧機能の作成方法を紹介する。

本稿の実装例では、ファイルサーバーに配置されているファイルがコード単位で管理されているケースを想定し、一覧照会画面から画面タップのみで、そのままファイル内容を閲覧できる機能を作成する。【図 1】

なお、本稿で使用する Delphi/400 のバージョンは 10.2Tokyo、端末は iPad（iOS13.4.1）である。

2. 照会機能の実装

今回、開発フレームワークとして FireMonkey を使用し、DataSnap による 3 層構造でアプリケーションを作成する。3 層構造アプリケーションについては、2012 年発行のミガロ . テクニカルレポートにある『DataSnap を使用した 3 層アプリケーション構築技法』、また iOS での基本的な開発手法については 2014 年発行の『iOS/Android ネイティブアプリケーション入門』で詳しく説明しているので、それぞれ参考にしていただきたい。

照会機能については、上記レポートを参考に作成可能なので、本稿では詳細な説明は割愛する。今回は【ソース 1】の DDS より作成された「商品マスタ（ファイル名：MSSYHF）」を使用する。

2-1. DataSnap サーバープログラム

今回、サーバープログラムはサービスアプリケーションとして新規に作成する。新規作成のウィザードに沿って作成すると、以下の 2 つのユニットをもつプロジェクトが生成される。

(1) ServerControllerUnit1.pas
(2) ServerMethodsUnit1.pas

サーバープログラムをサービスアプリケーションとして登録する場合は、Windows のタスクマネージャーによりサーバーの開始・停止をコントロールするので、ServerControllerUnit1 の Name および DisplayName をタスクマネージャー上で確認しやすい名前に変更することを推奨する。【図 2】

次に、ServerMethodsUnit1 には IBM i と接続するためのデータベース関連のコンポーネントを配置する。本稿では FireDAC 接続を使用して IBM i と接続するため、TFDPhysCO400DriverLink、TFDTable、TDataSetProvider

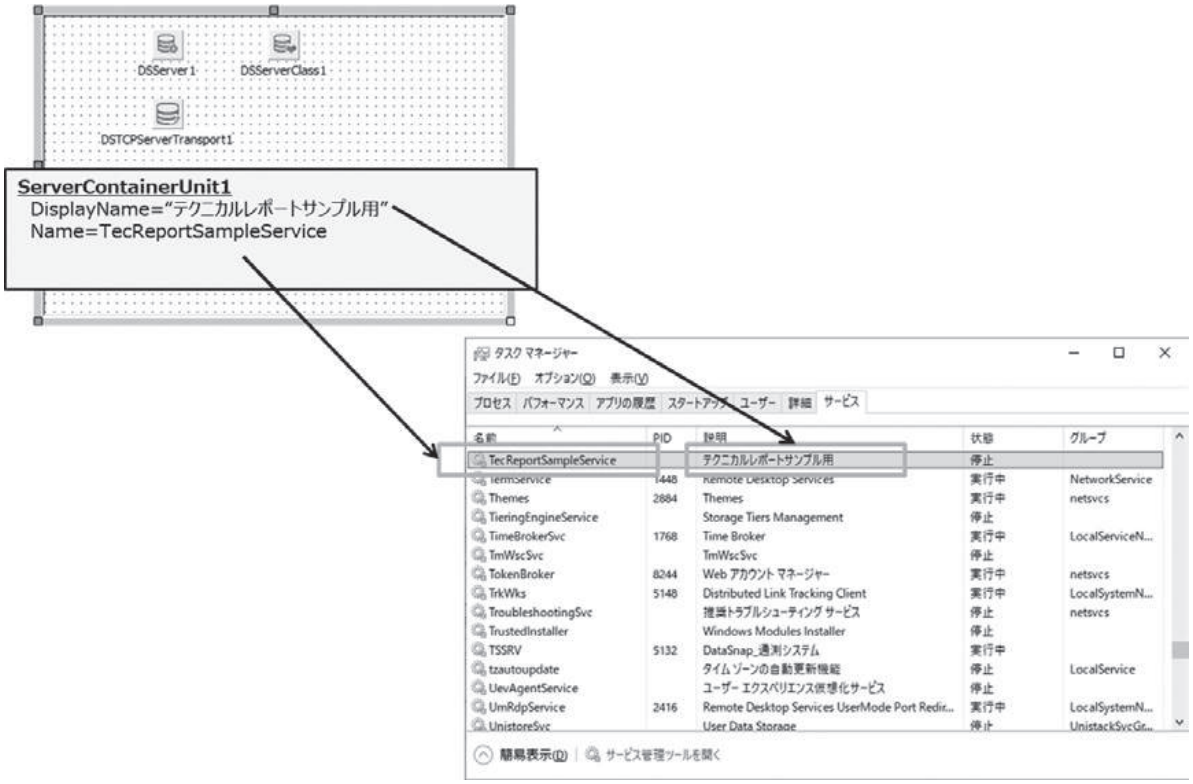
図1　本稿での作成画面

ソース1

DDS：商品マスタ

```
     ****************** データの始め **********************************
0001.00      A*************************************************************
0002.00      A*       FILE-ID    :    MSSYHF
0003.00      A*       FUNCTION   :    商品マスタ
0004.00      A*************************************************************
0005.00      A                                      UNIQUE
0006.00      A          R MSSYHR                    TEXT('商品マスタ')
0007.00      A            SYSHCD          5A        COLHDG('商品コード')
0008.00      A            SYSHNM         520        COLHDG('商品名')
0009.00      A          K SYSHCD
```

図2　サービスの名前指定



を貼り付け、プロパティを設定する。【図3】

一通りの設定が終わったら、サービスプログラムをコマンドプロンプトにてインストールし、タスクマネージャーからサービスの「開始」を実行する。【図4】

2-2. DataSnap クライアントプログラム

クライアントプログラムはiOS用のアプリケーションを作成するため、FireMonkeyのフレームワークを使って作成する。

[ファイル¦新規作成¦マルチデバイスアプリケーション]を選択し、新規作成を行うとForm1が生成される。画面デザインのスタイルはiOSに設定しておき、併せてプロジェクトマネージャのターゲットプラットフォームもiOSデバイスに設定しておく。【図5】

まずは、取得したマスタデータを表示するための画面レイアウト部分から作成する。TToolBar、TTabControlのコンポーネントを貼り付け、TTabControlは右クリックで「TTabItemの追加」を選択する。すると、TabItem1が追加されるので、追加したTabItem1上にTListViewを配置するようレイアウトする。【図6】

次に、先ほど作成したDataSnapサーバーアプリケーションとの接続処理部分を作成する。

TSQLConnection、TDSProviderConnection、TClientDataSetのコンポーネントを貼り付け、各プロパティの設定を行う。【図7】

あとは、画面表示時にconnServerのConnectedをTrueにしたあと、cdsSHDataClientをOpenし、lvSHDataにマスタの取得値を追加していけば、照会画面の完成である。【ソース2】【図8】

3. ファイル一覧機能の実装

ここでは、ファイルサーバーを参照してファイルの一覧を取得する。今回、ファイルサーバーには、「¥¥osknas02¥users¥osk前坂¥TecReport¥」に商品マスタの商品コード単位でフォルダが配置され、各フォルダ内にはその商品に関連するPDFファイルやExcelファイル等が配置されているものとする。【図9】

実装例では、前項で作成したプログラムを拡張してアプリケーションを作成するが、DataSnapサーバープログラムとDataSnapクライアントプログラムの両方を拡張する必要がある。そのため、まずは両方をひとまとめにしたプロジェクトグループを作成することを推奨する。【図10】

プロジェクトグループを作成しておくことで、「プロジェクトを閉じる→プロジェクトを開く」を繰り返し行う手間が省け、メンテナンス効率を向上させられる。

3-1. DataSnap サーバープログラム

まず、DataSnapサーバープログラムの拡張を行う。ServerMethodsUnit1にTClientDataSet、TDataSetProviderを貼り付け、プロパティを設定する。また配置したcdsFileListには、画面表示用項目としてファイル名（FLNAME）、ファイルサイズ（FLSIZE）を追加し、内部保持用項目としてファイルパス（FLPATH）、フォルダ区分（FLDRKB）の合計4つの項目を追加する。【図11】

なお項目の追加は、フィールドエディタより右クリックで「フィールドの新規追加」にて追加する。

次にソースを修正する。ServerMethodsUnit1のpublic宣言部にprocedure GetFileList（ADIR: String）を追加し、ロジックを記述する。GetFileListの手続きでは、引数でフォルダパスを受け取り、受け取ったパスでフォルダ内を参照する。

フォルダ参照時、ファイル名、ファイルサイズ、ファイルのフルパスをcdsFileListにAppendする。今回はロジックで指定した拡張子のみを一覧表示の対象とし、Word（.docx、.doc）、Excel（.xlsx、.xls）、PDF（.pdf）、画像（.jpg、.jpeg、.png）のファイルを拡張子に指定する。

なお、参照対象がファイルでなくフォルダであった場合は、FLDRKBに1をセットする【ソース3】。ロジックの修正が完了すれば、コンパイルを実施し、再度サーバープログラムを配置し直す必要がある。サーバープログラムのサービス実行中に再配置しようとするとエラーになるので、再配置の際は一度タスクマネージャーよりサービスを停止する必要がある。【図12】

また、今回のようにファイルサーバー接続時にサーバーへのアクセス権限が必要であれば、サーバー起動の際に併せて権限を付与する必要がある。

これにはまず、タスクマネージャーのサービスタブ下部にある「サービス管理ツールを開く」を選択する。サービス管理対象の一覧が表示されるので、自身が作成したサービスを選択し、右クリックからプロパティを選択する【図13】。すると、プロパティのダイアログ画面が表示されるので、ログオンタブを指定する。

設定値を「ローカルシステムアカウント」から「アカウント」に変更し、ファイルサーバーにアクセス可能なアカウント情報を設定してOKを押下すれば、権限の付与は完了である【図14】。あとは再度右クリックから「開始」を選択すると、サービスが開始される。

3-2. DataSnap クライアントプログラム

次に、DataSnapクライアントプログラムを拡張する。画面レイアウトの修正でまず行うのは、以下の2点である。

1点目はDataSnapサーバープログラムで取得したファイル一覧を表示するためのTListViewの追加、2点目はファイル一覧から照会画面に戻るためのTButtonの追加である。

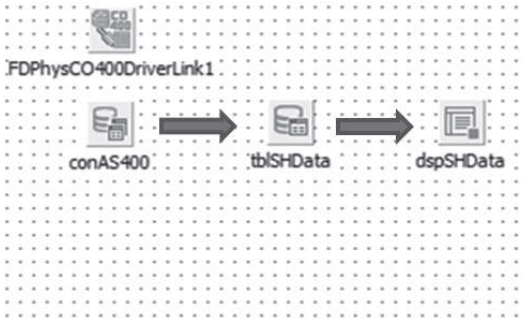
1点目のTListViewの追加は、前述したようにtbcMainより「TTabItemの追加」を行ったあと、TListViewを貼り付ける。2点目のボタン追加は、tbHeader上にTButtonを貼り付けるのみである。

画面レイアウトの修正が完了すると、次はDataSnapサーバープログラムとの連携用のコンポーネントを貼り付ける。TClientDataSet、TSqlServerMethodを貼り付け、プロパティを設定する。【図15】

ここでのコンポーネントの貼り付けは以上であり、次はソースを修正する。今回は商品一覧照会画面でレコードをタップした際、紐づく商品コードのファイル一覧を表示する。ソースの修正では、以下の2点の処理を追加する。

1点目は、商品のリストを選択した際のイベントを作成し、ファイル一覧データを取得して画面に表示する処理を追加する（追加処理1）。2点目は、前画面に戻るボタンを押下した際の処理を追加す

図3　コンポーネント設定値



FDPhysCO400DriverLink1 : TFDPhysCO400DriverLink1
設定値変更なし

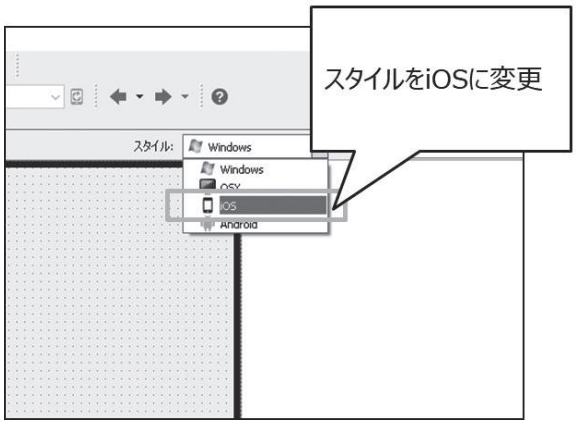
conAS400 : TFDConnection
DriverName=CO400
LoginPrompt=False
Params=(IBMi上のデータライブラリに接続する設定)

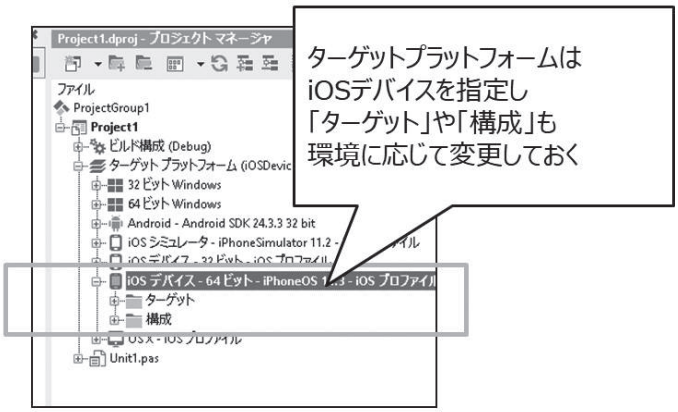
tblSHData : TFDTable
Connection=conAS400
TableName=MSSYHF

dspSHData : TDataSetProvider
DataSet=tblSHData

図4　サービスの開始

図5　iOSデバイスの指定





る（追加処理 2）。

追加処理 1 は、ファイル一覧データの取得と表示部分をまとめて、手続き GetFileListData にて記述する。

まずは Form1 の private 宣言部に、変数 FCurrDir と procedure GetFileListData を追加する。【ソース 4】

変数 FCurrDir では参照対象のフォルダパスを保持し、GetFileListData の手続きで svmGetFileList のパラメータとして渡す。その後、svmGetFileList の ExecuteMethod を呼び出すと、DataSnap サーバープログラムの GetFileList（【ソース 3】）が実行される。あとは cdsFileListClient を Open し、lvFileList に取得したファイル名（FLNAME）、ファイルサイズ（FLSIZE）を追加すると、GetFileListData の処理は完成である。

なお lvFileList に値をセットする際、取得内容がフォルダ（FLDRKB が 1）であった場合はファイルサイズが 0 となるため、フォルダ名のみを表示させている。【ソース 5】

次に、lvSHData の OnItemClickEx イベントにて処理を記述する。OnItemClickEx イベントでは、FCurrDir に参照フォルダパスの値をセットし、先ほど記述した GetFileListData の処理を呼び出す。

本サンプルでの参照フォルダは、「¥¥osknas02¥users¥osk 前坂 ¥TecReport¥」に商品コードを付与したパスであるため、まずは現在の選択行の商品コードを取得する処理を記述する。

商品コードは cdsSHData の SYSHCD で保持しており、RecNo のレコード位置を画面の選択行と同期をとることで、選択行の商品コードが取得可能である。OnItemClickEx イベントの引数である ItemIndex には選択行の番号を保持しているが、0 始まりであるため＋ 1 の値を RecNo にセットする。

RecNo の変更後、商品コード（SYSHCD）を付与した参照フォルダパスを FCurrDir にセットし、GetFileListData の処理を呼び出す。

あとは、ファイル一覧を表示するために ActiveTab を tbiFileList に変更する処理を追加すれば、追加処理 1 は完成である。【ソース 6】

追加処理 2 は、btnBack の OnClick イベントにて ActiveTab を tbiSHData に変更し、btnBack の Visible を False にする処理を記述する。また、画面起動時もボタンの表示は不要なので、Visible を False にしておく【ソース 7】。以上で、ファイル一覧機能の実装は完了である。【図 16】

4. ファイル閲覧機能の実装

ここでは、前の項で作成したファイル一覧より、選択ファイルの内容を表示する機能を実装する。ファイルの内容を表示する仕組みとしては、サーバープログラムにて取得したファイルデータをアプリ内の DataContainer の tmp フォルダに保存し、保存した内容を TWebBrowser コンポーネントで表示する仕組みである。【図 17】

実装例では、前の項で作成したプログラムを拡張してアプリケーションを作成する。

4-1. DataSnap サーバープログラム

まずは、DataSnap サーバープログラムを拡張する。

ファイル一覧機能作成時と同様に、ServerMethodsUnit1 に TClientDataSet と TDataSetProvider を貼り付け、プロパティを設定する。配置した cdsFileData には、内部保持用項目として、ファイルデータ（FLDATA）を追加する。FLDATA にはバイナリーデータとしてファイル内容を保持するため、TBlobField として項目を作成しておく。【図 18】

次にソースを修正する。ServerMethodsUnit1 の public 宣言部に procedure GetFileData（AFilePath: String）を追加し、ロジックを記述する。GetFileData の手続きでは、引数で表示対象ファイルのフルパスを受け取り、配置した cdsFileData の FLDATA にファイルデータを保持する処理を記述する。

ファイルデータを保持と聞くと、一見複雑な処理記述をイメージするかもしれないが、Delphi/400 では、作成した TBlobField に対して LoadFromFile（ファイルパス）を呼び出すだけで簡単にファイルデータを保持できる【ソース 8】。ロジックの修正が完了すれば、あとは前の項で実施したのと同様に、プログラムを再配置する。

4-2. DataSnap クライアントプログラム

次に、DataSnap クライアントプログラムを拡張する。画面レイアウトの修正でまず行うのは、以下の 2 点である。

1 点目は DataSnap サーバープログラムで取得したファイル内容を表示するための TWebBrowser の追加、2 点目はファイル一覧からフォルダを選択した際の、1 つ前のフォルダに戻るための TButton の追加である。

1 点目の TWebBrowser の追加には、まず前述したのと同様に TTabItem を追加し、TWebBrowser を貼り付ける。また 2 点目のボタン追加についても、前述したのと同様に tbHeader 上に TButton を貼り付ける。

画面レイアウトの修正が完了すると、次は DataSnap サーバープログラムで作成したファイルデータの取得処理を実行するためのコンポーネントを貼り付ける。こちらも前述したのと同様に、TClientDataSet と TSqlServerMethod を貼り付け、プロパティを設定する。【図 19】

コンポーネントの貼り付けは以上であり、次はソースを修正する。ここでは、ファイル一覧画面でレコードをタップした際に紐づくファイルの内容を表示する。ソースの修正では次の 3 点の処理を追加する。

1 点目は、ファイル一覧を選択した際のイベントを作成し、ファイルデータを wbFileData に表示するソースを記述する（追加処理 1）。2 点目は、1 つ上のフォルダに戻るボタンを押下した際の処理を追加する（追加処理 2）。そして 3 点目は、画面破棄時に tmp フォルダ以下に作成したファイルを削除する処理を追加する（追加処理 3）。

追加処理 1 はファイルデータの取得と表示部分をまとめて、手続き GetFileDataDisp にて記述する。

まずは、uses 節に System.IOUtils を追加し、private 宣言部にて変数 FTmpDir、FRootDir、FDirPathCnt、FBefoDir と procedure GetFileDataDisp を追加する。【ソース 9】

次に画面生成時に、FTmpDir の値セット処理、FBefoDir 変数の Create

図6　TTabItemの追加

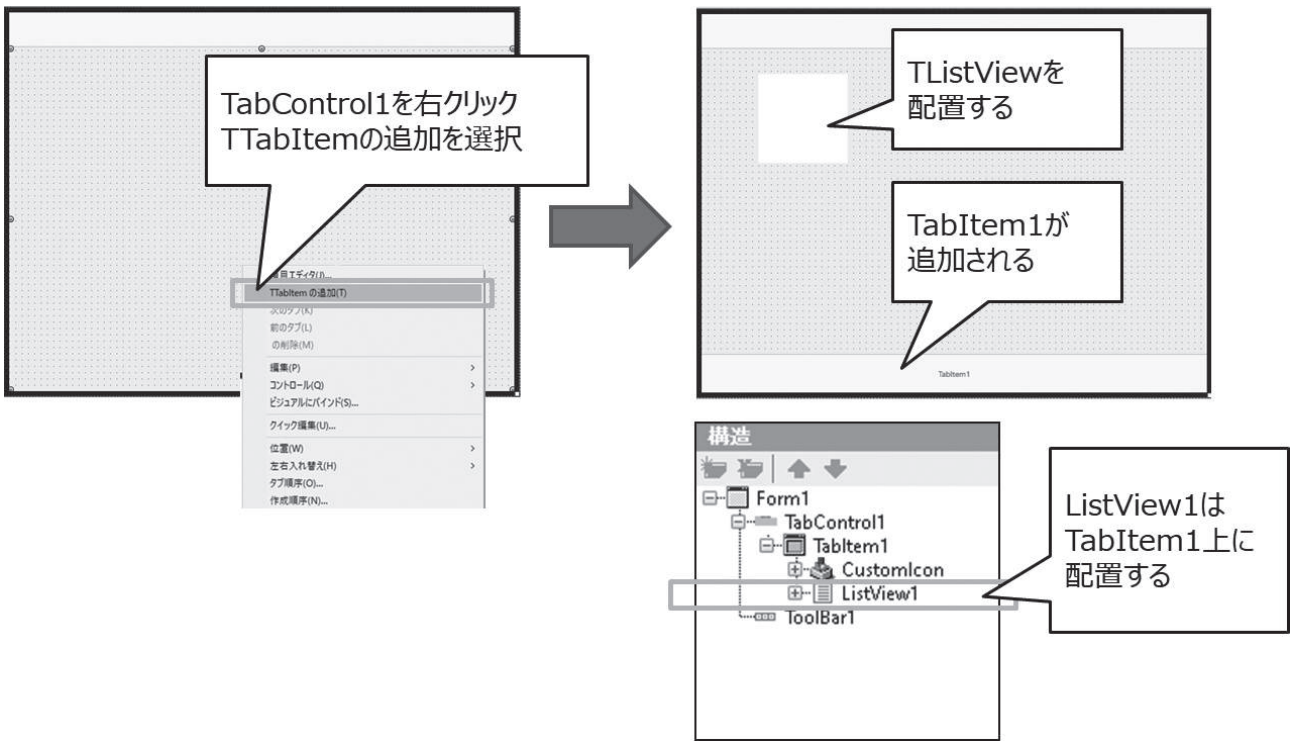
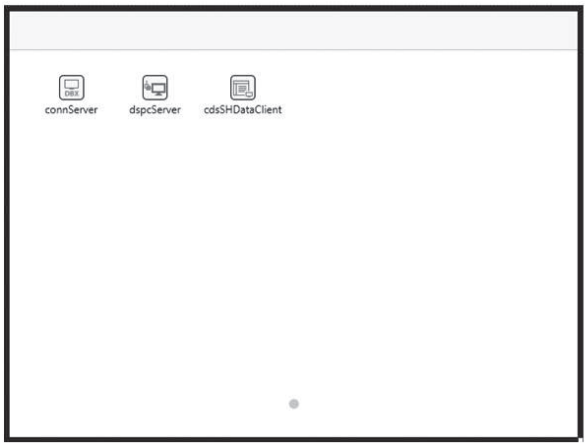


図7　コンポーネント設定値

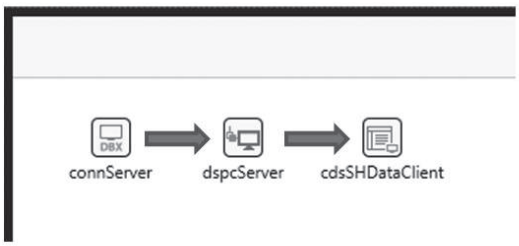


tbHeader : TToolBar
設定値変更なし

tbcMain : TTabControl
Align=Client
TabPosition=None

tbiSHData : TTabItem
設定値変更なし

lvSHData : TListView
Align=Client
ItemAppearance＞ItemHeight=80



connServer : TSQLConnection
Driver=DataSnap
ConnectionName=DataSnapCONNECTION
LoginPrompt=False
Params=(サーバのPort番号、HostNameを指定)

dspcServer : TDSProviderConnection
SQLConnection=connServer
ServerClassName=TServerMethods1

cdsSHDataClient : TClientDataSet
RemoteServer=dspcServer
ProviderName=dspSHData

処理を記述し、商品一覧選択時に、FRootDirの値セット処理を追加する【ソース10】。FTmpDirでは、tmpフォルダのパスを内部保持している。

それから、GetFileDataDispの処理を記述する。GetFileDataDispでは、パラメータに表示対象のファイルパスを渡し、svmGetFileDataのExecuteMethodを呼び出す。その後、cdsFileDataClientをOpenし、SaveToFileメソッドを呼び出すだけでtmpフォルダへの保存が可能である。

ただしiOS端末でファイル保存の際は、濁点の付いたファイルは保存できないため、いったん別名に変換して保存しなければならない。

なお本稿では、ファイル名を連番値に変換して保存を実施している。あとはwbFileDataのNavigateメソッドで、「'file://'＋保存ファイルパス」を指定すれば、GetFileDataDisp処理の完成である。【ソース11】

次に、lvFileListのOnItemClickExイベントにて処理を記述する。OnItemClickExイベントではcdsFileListClientのレコード位置の同期後、選択行がファイルデータの場合に、先ほど記述したGetFileListDataの処理を呼び出し、フォルダの場合は前項で記述したファイル一覧のセット処理を呼び出す。【ソース12】

追加処理2は、btnUPDirのOnClickイベントにてFCurrDirの値を変更し、画面内容を再セットする処理を記述する。また画面起動時にボタンの表示は不要であるため、VisibleをFalseにしておく。【ソース13】

追加処理3は、OnDestroyイベントにて削除処理を記述する。【ソース14】

以上で、ファイル閲覧機能の実装は完了である。【図20】

なお、ExcelやWordは端末にインストールされていなくても閲覧可能であるが、Windowsにて作成されたファイルの場合は、罫線やオートシェイプなど一部表示されないものもあるので注意が必要である。

5. 他アプリへのファイル保存機能の実装

ここでは、前項で閲覧可能となったファイルに対し、端末にインストールされているiBookなど他アプリへの保存機能を追加する。

まずは、画面の上部にファイル保存のTButtonを追加する。ここでのソース修正については、DataSnapクライアントプログラムのみ行う。ファイル保存ではIOSapiのUIDocumentInteractionControllerの機能を利用し、ダイアログを表示して行うが、その際にファイルのフルパスとダイアログの表示位置を指定する必要がある。

そのため、まずはprivate宣言部にFFullFileNameとFPointの変数を追加する。FFullFileNameは、ファイルデータ表示処理のタイミングで値を保持するよう修正する。【ソース15】

次に、ファイル保存用に配置したボタンのOnTapイベントでタップした位置をFPointに内部保持し、OnClickイベントでファイル保存ダイアログの表示処理を記述していく。【ソース16】

以上で、他アプリへのファイル保存機能の実装は完了である。【図21】

6. さいごに

本稿では、iOSアプリでのファイル閲覧機能の作成について紹介してきた。FireMonkeyフレームワークの使用や3層構造といったテクニックも絡めながら実装例を紹介したため、難易度が高く感じられたかもしれない。

しかし実際には、サーバープログラムとクライアントプログラムを合わせても500ステップ前後の記述で実装できているのである。

今後、変化していく生活環境の中、プログラム開発者もより高度な要望や技術が求められていくだろう。Delphi/400では、このように少量のロジック記述で実現できることが多くあるので、高度な内容でもぜひ開発にチャレンジしていただきたい。その際、本稿で紹介した内容が少しでも参考になれば幸いである。

M

ソース2

OnShowイベント

```
{*******************************************************************************
 目的：画面表示時処理
 引数：
 戻値：
*******************************************************************************}
procedure TForm1.FormShow(Sender: TObject);
begin
  // サーバと接続
  connServer.Connected := True;

  // 商品マスタのデータ取得
  cdsSHDataClient.Open;

  // 画面にセット
  cdsSHDataClient.First;
  while not cdsSHDataClient.Eof do
  begin
    // 商品コード + スペース + 商品名
    lvSHData.Items.Add.Text :=
        cdsSHDataClient.FieldByName('SYSHCD').AsString + '  '
      + cdsSHDataClient.FieldByName('SYSHNM').AsString;

    cdsSHDataClient.Next;
  end;

  // 1行目を選択行に
  lvSHData.ItemIndex := 0;
end;
```

図8　照会画面

図9　参照フォルダ内容

図10　プロジェクトグループの作成

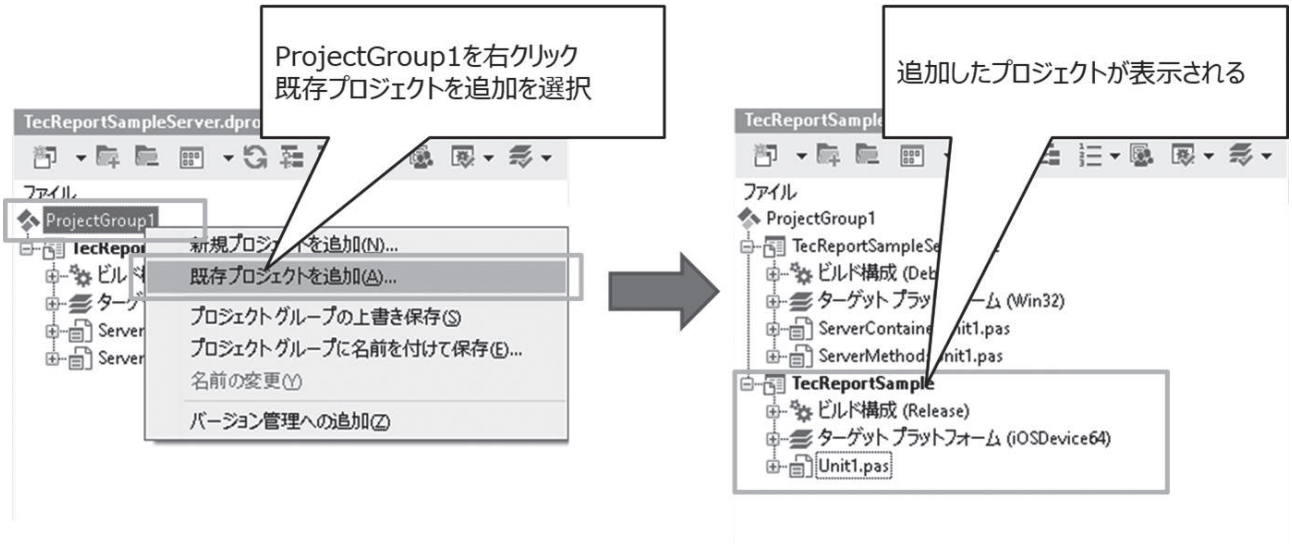
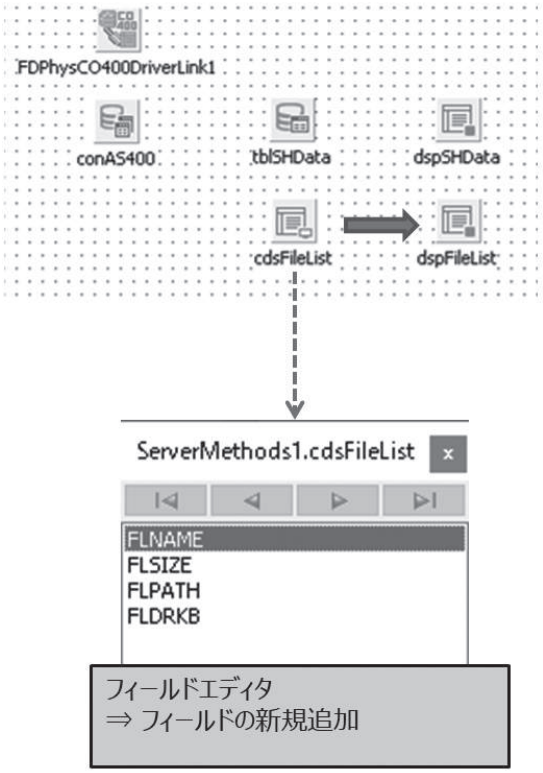


図11　コンポーネント設定値



cdsFileList : TClientDataSet
設定値変更なし

dspFileList : TDataSetProvider
DataSet=cdsFileList

FLNAME : TStringField
Size=500　※ 配置ファイル名の長さに合わせて指定

FLSIZE : TIntegerField
設定値変更なし

FLPATH : TStringField
Size=1000　※ 配置ファイルパスの長さに合わせて指定

FLDRKB : TIntegerField
設定値変更なし

GetFileList(指定フォルダのファイル一覧を取得)

```
{*****************************************************************************
  目的: ファイル一覧取得処理
  引数: ADIR - フォルダ名
  戻値:
*****************************************************************************}
procedure TServerMethods1.GetFileList(ADIR: String);
var
  SearchRec: TSearchRec;  // ファイル情報
  slExt: TStringList;     // 取得対象の拡張子保持
begin
  slExt := TStringList.Create;
  try
    // 指定した拡張子のファイルを表示する
    slExt.CommaText := '.DOCX,.DOC,.XLSX,.XLS,.PDF,.JPG,.JPEG,.PNG';

    // データセット準備
    cdsFileList.Close;
    cdsFileList.CreateDataSet;
    cdsFileList.EmptyDataSet;

    // フォルダが存在しない場合、処理中断
    if not System.SysUtils.DirectoryExists(ADIR) then
    begin
      Exit;
    end;

    // 指定ディレクトリのすべての種類のファイルを列挙
    if (System.SysUtils.FindFirst(ADIR + '*.*', faAnyFile, SearchRec) = 0) then
    begin
      try
        while (System.SysUtils.FindNext(SearchRec) = 0) do
        begin
          // カレントディレクトリ・親ディレクトリ・隠しファイルは対象外
          if (SearchRec.Name <> '.') and
             (SearchRec.Name <> '..') and
             (not Bool(SearchRec.Attr and FILE_ATTRIBUTE_HIDDEN)) then
          begin
            // 対象の拡張子のファイルかチェック
            if (slExt.Count <> 0) and
               (slExt.IndexOf(UpperCase(ExtractFileExt(SearchRec.Name))) < 0) and
               (not ((SearchRec.Attr and faDirectory) > 0)) then
            begin
              Continue;
            end;

            // ファイル名を明細に追加
            cdsFileList.Append;

            // ファイル名
            cdsFileList.FieldByName('FLNAME').AsString  := SearchRec.Name;

            // ファイルサイズ（キロバイト表示）
            cdsFileList.FieldByName('FLSIZE').AsInteger := SearchRec.Size div 1024;

            // ファイルパス
            cdsFileList.FieldByName('FLPATH').AsString  := ADIR + SearchRec.Name;
            if ((SearchRec.Attr and faDirectory) > 0) then
            begin
              // ディレクトリ
              cdsFileList.FieldByName('FLDRKB').AsInteger := 1;
            end
            else
            begin
              // ファイル
              cdsFileList.FieldByName('FLDRKB').AsInteger := 0;
            end;
            cdsFileList.Post;
          end;
        end;
      finally
        System.SysUtils.FindClose(SearchRec);
      end;
    end;

    cdsFileList.First;

  finally
    FreeAndNil(slExt);
  end;
end;
```

図12　サービス停止方法

図13　サービス管理ツールの起動

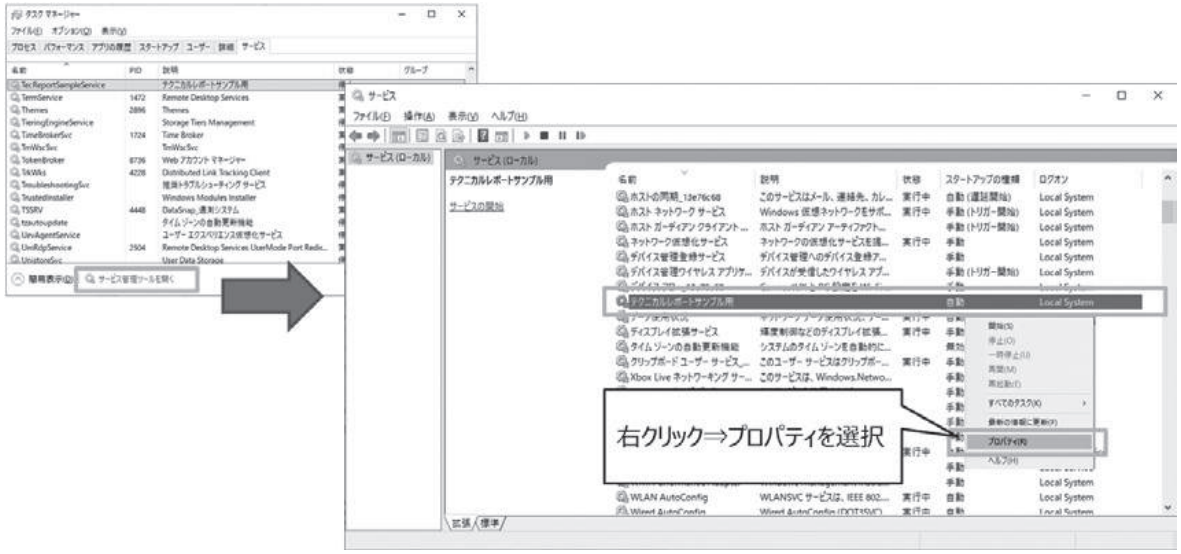


図14　ログオンアカウントの設定

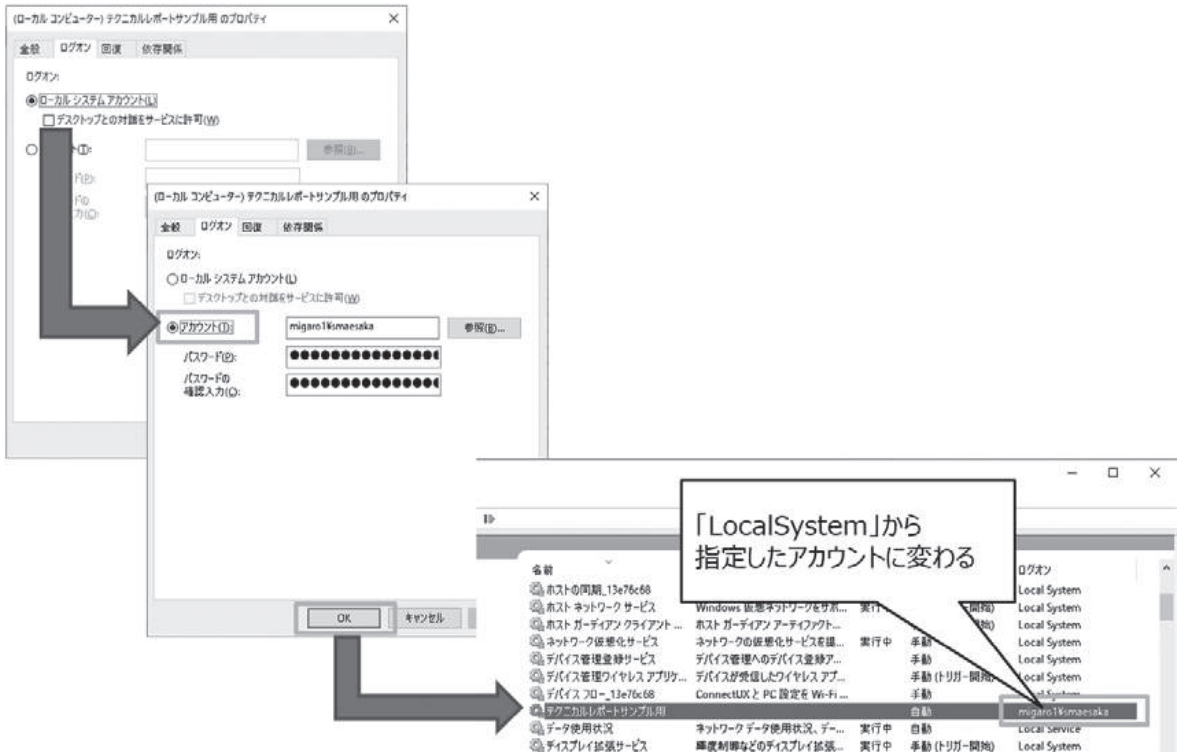


図15　コンポーネント設定値

tbiFileList : TTabItem
設定値変更なし

lvFileList : TListView
Align=Client
ItemAppearance> ItemHeight=80

btnBack : TButton
Align=MostLeft
Text="前画面へ戻る"

cdsFileListClient : TClientDataSet
RemoteServer=dspcServer
ProviderName=dspFileList

svmGetFileList : TSqlServerMethods
SQLConnection=connServer
ServerMethodName=TServerMethods1.GetFileList

ソース4

private宣言部に追加

```
  Datasnap.DSConnect, Data.SqlExpr;

type
  TForm1 = class(TForm)
    tbHeader: TToolBar;
    tbcMain: TTabControl;
    tbiSHData: TTabItem;
    tbiFileList: TTabItem;
    connServer: TSQLConnection;
    dspcServer: TDSProviderConnection;
    cdsSHDataClient: TClientDataSet;
    cdsFileListClient: TClientDataSet;
    cdsFileDataClient: TClientDataSet;
    lvSHData: TListView;
    lvFileList: TListView;
    btnBack: TButton;
  private
    { private 宣言 }
    FCurrDir: String;                    // フォルダパスを保持
    procedure GetFileListData;           // ファイル一覧取得処理
  public
    { public 宣言 }
  end;

var
  Form1: TForm1;

implementation

{$R *.fmx}

{ TForm1 }

procedure TForm1.GetFileListData;
begin

end;
```

追加

ソース5

GetFileListData（指定フォルダのファイル一覧を表示）

```
{*************************************************************************
 目的：ファイル一覧取得処理
 引数：
 戻値：
*************************************************************************}
procedure TForm1.GetFileListData;
begin
  // リストをクリア
  lvFileList.Items.Clear;
  cdsFileListClient.Close;

  // 参照対象のフォルダパスをパラメータに渡す
  svmGetFileList.ParamByName('ADIR').AsString := FCurrDir;

  // ファイル一覧を取得
  svmGetFileList.ExecuteMethod;
  cdsFileListClient.Open;

  // 画面にセット
  cdsFileListClient.First;
  while not cdsFileListClient.Eof do
  begin
    // フォルダの場合
    if cdsFileListClient.FieldByName('FLDRKB').AsInteger = 1 then
    begin
      // フォルダ名を表示
      lvFileList.Items.Add.Text :=
        cdsFileListClient.FieldByName('FLNAME').AsString;
    end
    // ファイルの場合
    else
    begin
      // ファイル名 + スペース + ファイルサイズ(KB)を表示
      lvFileList.Items.Add.Text :=
        cdsFileListClient.FieldByName('FLNAME').AsString + '    '
      + IntToStr(cdsFileListClient.FieldByName('FLSIZE').AsInteger) + 'KB';
    end;

    cdsFileListClient.Next;
  end;

end;
```

ソース6

OnItemClickEx（商品一覧の行選択時処理）

```
{*****************************************************************************
 目的：商品一覧選択時処理
 引数：
 戻値：
*****************************************************************************}
procedure TForm1.lvSHDataItemClickEx(const Sender: TObject; ItemIndex: Integer;
  const LocalClickPos: TPointF; const ItemObject: TListItemDrawable);
begin
  // 商品一覧のデータセットのレコード位置を同期
  cdsSHDataClient.RecNo := ItemIndex + 1;

  // 参照フォルダパスを保持
  // 固定参照先 + 商品コードのフォルダ
  FCurrDir :=
    '¥¥osknas02¥users¥osk前坂¥TecReport¥' +
    cdsSHDataClient.FieldByName('SYSHCD').AsString + '¥';

  // ファイル一覧を取得＆画面にセット
  GetFileListData;

  // ファイル一覧画面(タブ)を表示
  tbcMain.ActiveTab := tbiFileList;

  // 1行目を選択行に
  lvFileList.ItemIndex := 0;

  // 前画面に戻るボタンを表示
  btnBack.Visible := True;
end;
```

ソース7

OnClick（前画面へ戻るボタン処理）

```
{*****************************************************************************
 目的：前画面へ戻るボタン押下時処理
 引数：
 戻値：
*****************************************************************************}
procedure TForm1.btnBackClick(Sender: TObject);
begin
  if tbcMain.ActiveTab = tbiFileList then
  begin
    // 商品一覧を表示
    tbcMain.ActiveTab := tbiSHData;

    // 前画面へ戻るボタンを非表示
    btnBack.Visible := False;
  end;
end;
```

OnShow（画面表示時処理）

```
      + cdsSHDataClient.FieldByName('SYSHNM').AsString;

    cdsSHDataClient.Next;
  end;

  // 1行目を選択行に
  lvSHData.ItemIndex := 0;

  // 画面初期設定
  tbcMain.ActiveTab := tbiSHData;      // 商品一覧を表示
  btnBack.Visible   := False;          // 前画面へ戻るボタンを非表示
end;
```

追加

図16　ファイル一覧画面

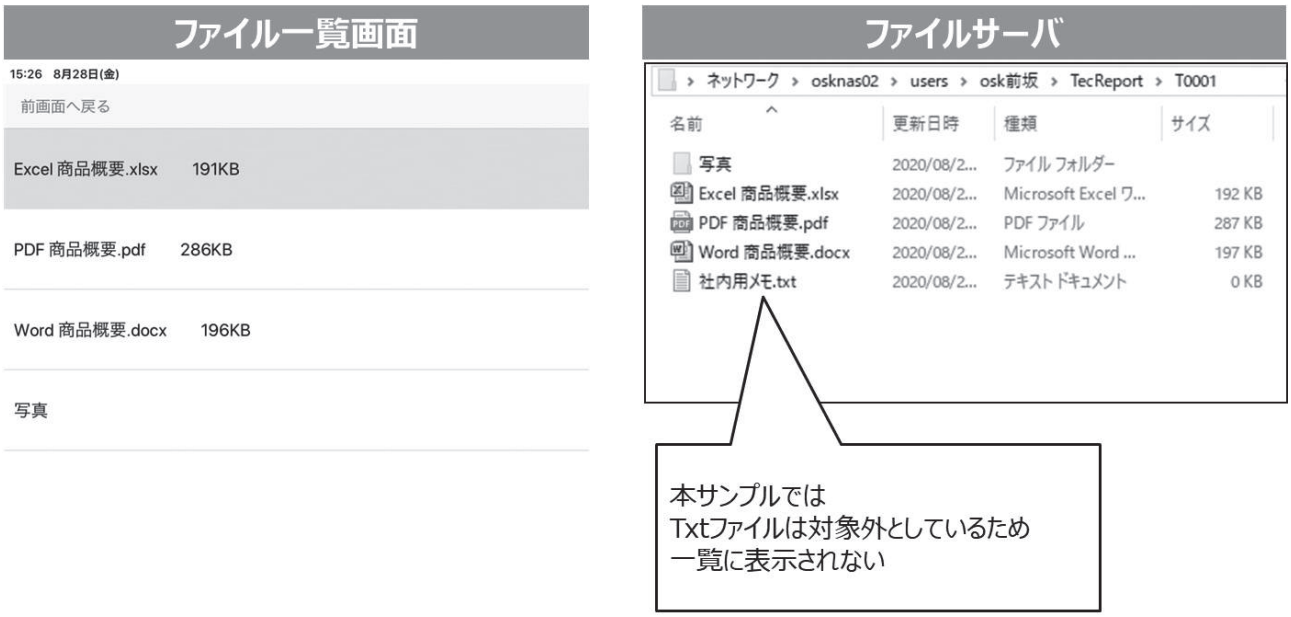


図17　ファイル内容表示の仕組み

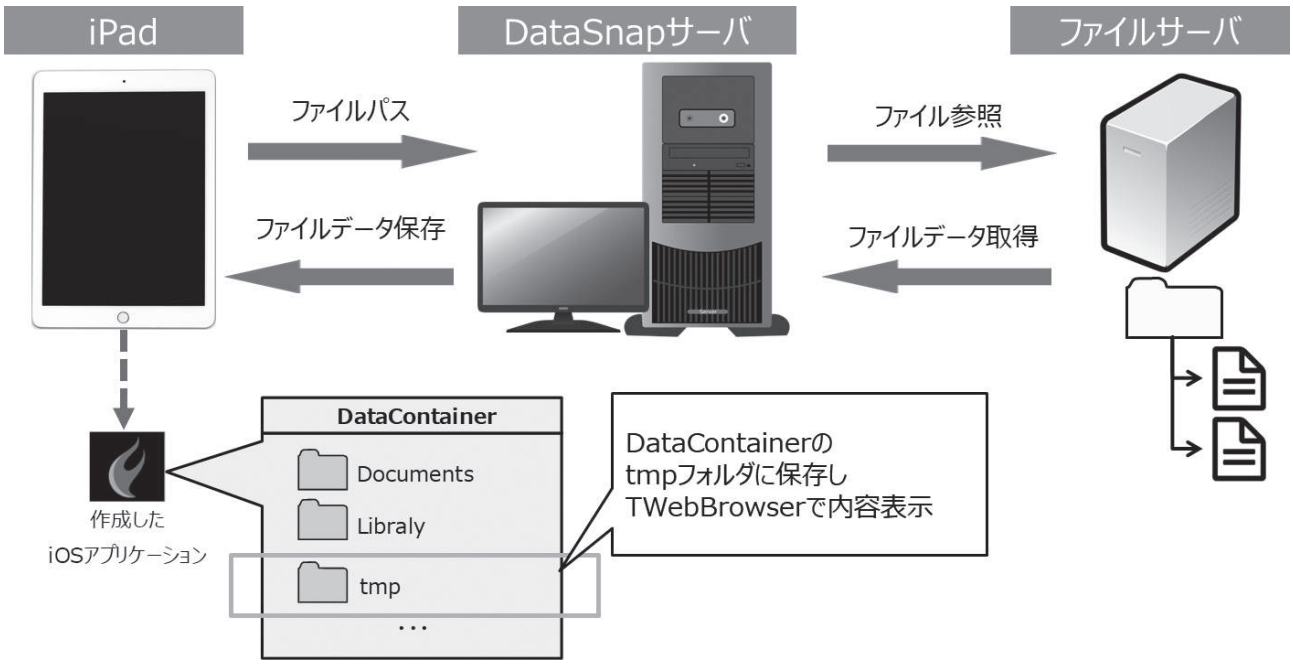
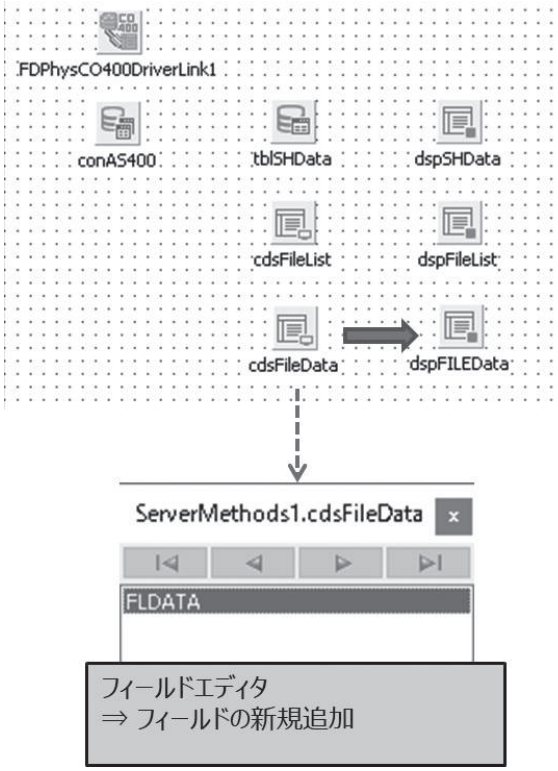


図18　コンポーネント設定値



cdsFileData : TClientDataSet
設定値変更なし

dspFileData : TDataSetProvider
DataSet=cdsFileData

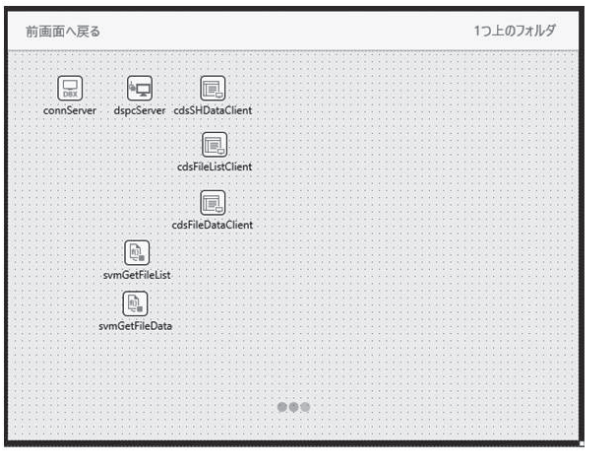
FLDATA : TBlobField
Size=1000000　※ 取得ファイルのファイルサイズに合わせて指定

ソース8

GetFileData（指定ファイルのデータを取得）

```
{*****************************************************************************
  目的：ファイルデータ取得処理
  引数：AFilePath - ファイルパス
  戻値：
*****************************************************************************}
procedure TServerMethods1.GetFileData(AFilePath: String);
begin

  // データセット準備
  cdsFileData.Close;
  cdsFileData.CreateDataSet;
  cdsFileData.EmptyDataSet;

  // ファイルが存在しない場合、処理中断
  if not System.SysUtils.FileExists(AFilePath) then
  begin
    Exit;
  end;

  // データセットにファイルを保持
  cdsFileData.Append;
  (cdsFileData.FieldByName('FLDATA') as TBlobField).LoadFromFile(AFilePath);
  cdsFileData.Post;

  // 先頭行
  cdsFileData.First;
end;
```

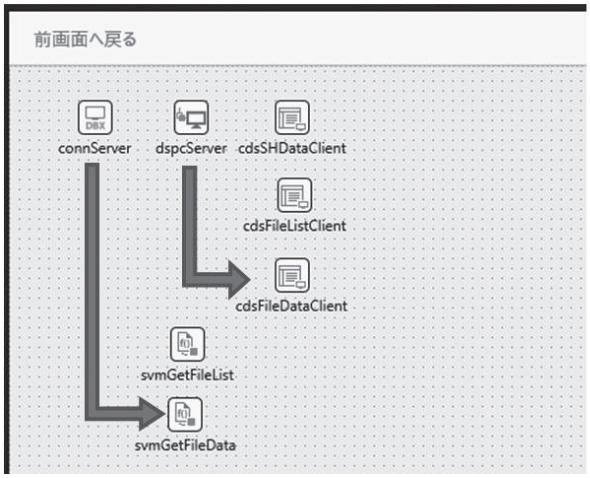
図19　コンポーネント設定値



tbiFileData : TTabItem
設定値変更なし

wbFileData : TWebBrowser
Align=Client

btnUpDir : TButton
Align=MostRight
Text="1つ上のフォルダ"



cdsFileDataClient : TClientDataSet
RemoteServer=dspcServer
ProviderName=dspFileData

svmGetFileData : TSqlServerMethods
SQLConnection=connServer
ServerMethodName=TServerMethods1.GetFileData

ソース9

uses節に追加

```
uses
  System.SysUtils, System.Types, System.UITypes, System.Classes, System.Variants,
  FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs, FMX.TabControl,
  FMX.Controls.Presentation, FMX.StdCtrls, Data.DBXDataSnap, Data.DBXCommon,
  IPPeerClient, FMX.ListView.Types, FMX.ListView.Appearances,
  FMX.ListView.Adapters.Base, FMX.ListView, Data.DB, Datasnap.DBClient,
  Datasnap.DSConnect, Data.SqlExpr, Data.FMTBcd, FMX.WebBrowser,
  System.IOUtils;

type
  TForm1 = class(TForm)
```

追加

private宣言部に追加

```
private
  { private 宣言 }
  FCurrDir: String;                 // フォルダパスを保持

  FTmpDir: String;                  // 端末内保存用フォルダパス
  FRootDir: String;                 // 初期表示フォルダパス
  FDirPathCnt: Integer;             // 保存ファイル数カウント
  FBefoDir: TStringList;            // 順に選択したフォルダパスを保持

  procedure GetFileListData;        // ファイル一覧取得処理

  procedure GetFileDataDisp;        // ファイルデータ表示処理
```

追加

追加

FormCreate(画面生成時処理)

```
{*************************************************************************
 目的：画面生成時処理
 引数：
 戻値：
*************************************************************************}
procedure TForm1.FormCreate(Sender: TObject);
begin
  // tmpフォルダのパス取得
  FTmpDir  := System.IOUtils.TPath.GetTempPath;

  // 内部保持用変数生成
  FBefoDir := TStringList.Create;
end;
```

OnItemClickEx(商品一覧選択時処理)

```
{*************************************************************************
 目的：商品一覧選択時処理
 引数：
 戻値：
*************************************************************************}
procedure TForm1.lvSHDataItemClickEx(const Sender: TObject; ItemIndex: Integer;
  const LocalClickPos: TPointF; const ItemObject: TListItemDrawable);
begin
  // 商品一覧のデータセットのレコード位置を同期
  cdsSHDataClient.RecNo := ItemIndex + 1;

  // 参照フォルダパスを保持
  // 固定参照先 + 商品コードのフォルダ
  FCurrDir :=
    '¥¥osknas02¥users¥osk前坂¥TecReport¥' +
    cdsSHDataClient.FieldByName('SYSHCD').AsStri + '¥';

  // 商品一覧選択時のパスを保持しておく
  FRootDir := FCurrDir;
```

追加

ソース11

GetFileDataDisp(ファイルデータ表示処理)

```
{*******************************************************************************
 目的: ファイルデータ表示処理
 引数:
 戻値:
*******************************************************************************}
procedure TForm1.GetFileDataDisp;
var
  sFullFileName: String;
  sSaveFileName: String;
begin
  cdsFileDataClient.Close;

  // 表示対象のファイルパスをパラメータに渡す
  svmGetFileData.ParamByName('AFILEPATH').AsString :=
    cdsFileListClient.FieldByName('FLPATH').AsString;

  // ファイルデータを取得
  svmGetFileData.ExecuteMethod;
  cdsFileDataClient.Open;

  // 端末内保存用にファイル名を変更（取得ファイル名⇒連番値）
  // ※ 濁点のファイル名が正しく処理されないため
  Inc(FDirPathCnt);
  sSaveFileName :=
    ChangeFileExt(cdsFileListClient.FieldByName('FLPATH').AsString, '');
  sSaveFileName :=
    StringReplace(
      cdsFileListClient.FieldByName('FLPATH').AsString,
      sSaveFileName,
      FormatFloat('00000', FDirPathCnt), [rfReplaceAll]);

  // デバイスに保存
  // フォルダ作成
  ForceDirectories(
    TPath.Combine(FTmpDir, FormatFloat('00000', FDirPathCnt) + '/'));

  // ファイル保存
  sFullFileName :=
    TPath.Combine(
      FTmpDir, FormatFloat('00000', FDirPathCnt) + '/' + sSaveFileName);
  (cdsFileDataClient.FieldByName('FLDATA')
    as TBlobField).SaveToFile(sFullFileName);

  // ファイル内容の表示
  wbFileData.Navigate('file://' + sFullFileName);
end;
```

ソース12

OnItemClickEx（ファイル一覧の行選択時処理）

```
{*****************************************************************************
 目的：ファイル一覧選択時処理
 引数：
 戻値：
*****************************************************************************}
procedure TForm1.lvFileListItemClickEx(const Sender: TObject; ItemIndex: Integer;
  const LocalClickPos: TPointF; const ItemObject: TListItemDrawable);
begin
  // ファイル一覧のデータセットのレコード位置を同期
  cdsFileListClient.RecNo := ItemIndex + 1;

  // 選択行がフォルダの場合
  if cdsFileListClient.FieldByName('FLDRKB').AsInteger = 1 then
  begin
    //  1つ上のフォルダを保持
    FBefoDir.Add(FCurrDir);

    // カレントフォルダをセット
    FCurrDir :=
      FCurrDir + cdsFileListClient.FieldByName('FLNAME').AsString + '¥';

    // ファイル一覧再表示
    GetFileListData;
  end
  else
  // 選択行がファイルの場合
  begin
    // ファイル内容表示処理
    GetFileDataDisp;

    tbcMain.ActiveTab := tbiFileData;
  end;

  // 先頭フォルダでない場合、1つ上のフォルダボタンを表示
  btnUPDir.Visible := (FCurrDir <> FRootDir);
end;
```

ソース13

OnClick（1つ上のフォルダボタン押下時処理）

```
{*****************************************************************************
 目的：1つ上のフォルダボタン押下時処理
 引数：
 戻値：
*****************************************************************************}
procedure TForm1.btnUPDirClick(Sender: TObject);
begin
  // カレントフォルダに1つ上のフォルダパスをセット
  FCurrDir := FBefoDir[FBefoDir.Count - 1];

  // カレントフォルダにセットしたフォルダパスを削除
  FBefoDir.Delete(FBefoDir.Count - 1);

  // データ再表示
  GetFileListData;
end;
```

OnShow（画面表示時処理）

```
  // 1行目を選択行に
  lvSHData.ItemIndex := 0;

  // 画面初期設定
  tbcMain.ActiveTab := tbiSHData;      // 商品一覧を表示
  btnBack.Visible   := False;          // 前画面へ戻るボタンを非表示
  btnUPDir.Visible  := False;          // 1つ上のフォルダボタンを非表示

end;
```

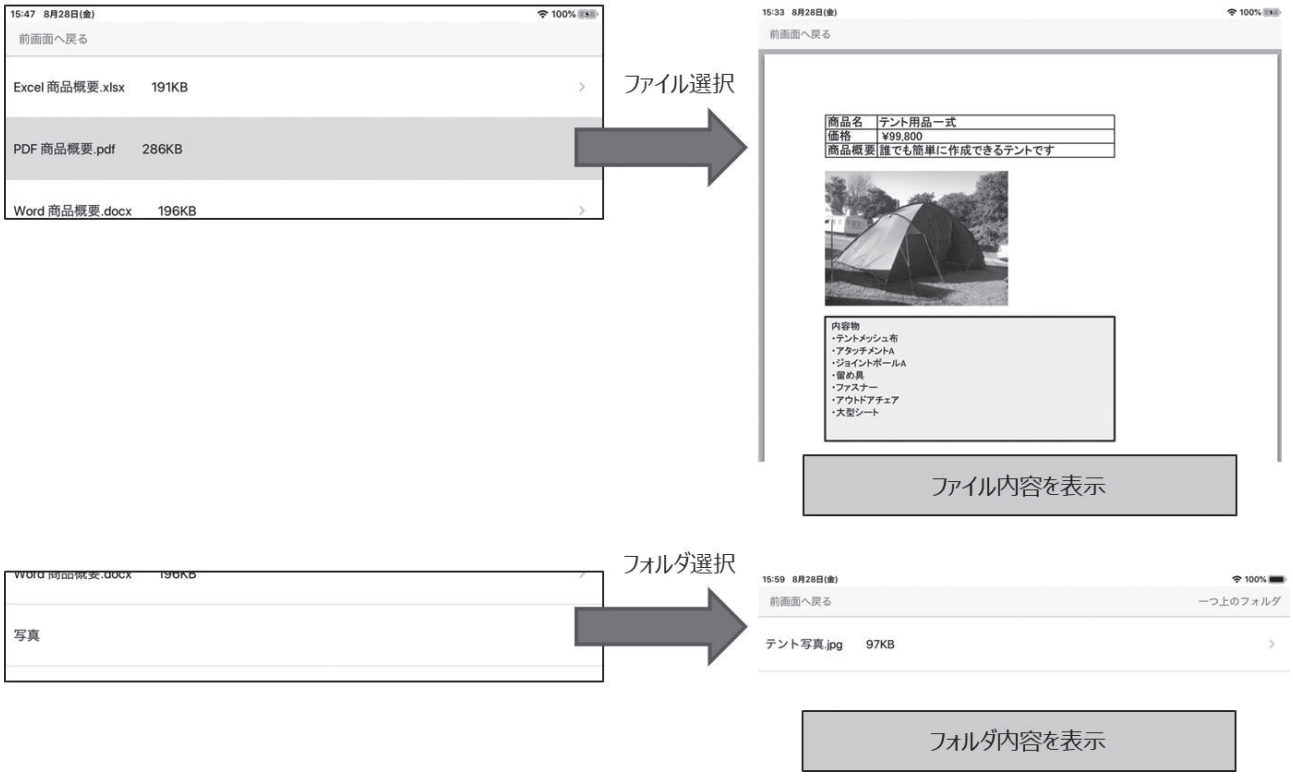
追加

ソース14

OnDestroy(画面破棄時処理)

```
{*****************************************************************************
  目的：画面破棄時処理
  引数：
  戻値：
*****************************************************************************}
procedure TForm1.FormDestroy(Sender: TObject);
var
  DirList: TStringDynArray;
  iLoop: Integer;
begin
  // 作成したtmpフォルダ以下のフォルダを削除
  DirList := TDirectory.GetDirectories(FTmpDir);
  for iLoop := 0 to Length(DirList) - 1 do
  begin
    TDirectory.Delete(DirList[iLoop], True);
  end;
end;
```

図20　ファイル閲覧画面



ソース15

private宣言部に追加

```
  private
    { private 宣言 }
    FCurrDir: String;                  // フォルダパスを保持

    FTmpDir: String;                   // 端末内保存用フォルダパス
    FRootDir: String;                  // 初期表示フォルダパス
    FDirPathCnt: Integer;              // 保存ファイル数カウント
    FBefoDir: TStringList;             // 順に選択したフォルダパスを保持

    FFullFileName: String;             // 表示ファイル名
    FPoint: TPointF;                   // タップ位置を保持

    procedure GetFileListData;         // ファイル一覧取得処理

    procedure GetFileDataDisp;         // ファイルデータ表示処理
  public
```

追加

GetFileDataDisp(ファイルデータ表示処理)

```
  (CustFileDataClient.FieldByName('FDATA') as TBlobField).SaveToFile(sFullFileNa

  // ファイル内容の表示
  wbFileData.Navigate('file://' + sFullFileName);

  // 内部保持
  FFullFileName := sFullFileName;
end;
```

追加

ソース16

uses節の追加

```
uses
  System.SysUtils, System.Types, System.UITypes, System.Classes, System.Va
  FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs, FMX.TabCo
  FMX.Controls.Presentation, FMX.StdCtrls, Data.DBXDataSnap, Data.DBXCommo
  IPPeerClient, FMX.ListView.Types, FMX.ListView.Appearances,
  FMX.ListView.Adapters.Base, FMX.ListView, Data.DB, Datasnap.DBClient,
  Datasnap.DSConnect, Data.SqlExpr, Data.FMTBcd, FMX.WebBrowser,
  System.IOUtils,
  iOSapi.Foundation, iOSapi.UIKit, Macapi.Helpers, FMX.Platform.Ios;
```

追加

OnTap、OnClick(ファイル保存ボタン押下時処理)

```
{*******************************************************************************
 目的: ファイル保存ボタン押下時処理
 引数:
 戻値:
*******************************************************************************}
procedure TForm1.btnFileSaveClick(Sender: TObject);
var
  URL: NSURL;
  FController : UIDocumentInteractionController;
  iTop, iLeft, iBottom, iRight: Integer;
begin
  FController := TUIDocumentInteractionController.Wrap(
  TUIDocumentInteractionController.Alloc.init);

  // ファイルパスを渡す
  URL := TNSUrl.Wrap(TNSUrl.OCClass.fileURLWithPath(StrToNSStr(FFullFileName)));
  FController.setURL(URL);

  // ダイアログ表示位置指定
  iTop    := Trunc(FPoint.Y);
  iLeft   := Trunc(FPoint.X);
  iBottom := iTop  + 10;
  iRight  := iLeft + 5;

  FController.presentOptionsMenuFromRect(
      RectToNSRect(TRect.Create(iLeft, iTop, iRight, iBottom)),
      WindowHandleToPlatform(self.Handle).View,
      true);
end;

{*******************************************************************************
 目的: ファイル保存ボタンタップ時処理
 引数:
 戻値:
*******************************************************************************}
procedure TForm1.btnFileSaveTap(Sender: TObject; const Point: TPointF);
begin
  FPoint := Point;
end;
```

図21　ファイル保存


18:02　8月28日(金)
100%
前画面へ戻る
ファイル保存
00002
PDF書類 · 293 KB
商品名　テント用品一式
価格　¥99,800
商品概要　誰でも簡単に作成できるテ
Skitchにコピー
SmartLibrary
ブックにコピー
その他
コピー
マークアップ
プリント
Skitch
"ファイル"に保存
アクションを編集...
内容物
・テントメッシュ布
・アタッチメントA
・ジョイントポールA
・留め具
・ファスナー
・アウトドアチェア
・大型シート
指定したアプリ内に保存可能
任意のフォルダなどに保存可能

佐田 雄一

株式会社ミガロ.

RAD事業部 技術支援課

[Delphi/400] Delphi 10シリーズ
VCLプログラム開発の最新トピックス

略歴
1985 年 12 月 6 日生まれ
2009 年 3 月 甲南大学 経営学部卒業
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
2019 年 4 月 RAD 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の経験を経て、現在は Delphi/400 のサポート業務を担当。

1. はじめに

2020 年 5 月に、Delphi の最新バージョンとなる「10.4 Sydney」がリリースされた。

Delphi および Delphi/400 では、コンポーネントをフォームに配置してプロパティを定義し、プログラムをコーディングしていく、「ビジュアルプログラミング」と呼ばれる開発手法でアプリケーションを作成する。本稿では、2015 年に登場した Delphi 10 シリーズの初版である「Delphi 10 Seattle」から、最新の「Delphi 10.4 Sydney」の間で進化したビジュアルプログラミングの手法を紹介する。

なお、本稿執筆（2020 年 9 月）時点で Delphi は「10.4 Sydney」が最新だが、IBM i に接続するためのミドルウェア「Delphi/400」は「10.2 Tokyo」が最新となっているため、今後のリリースにご期待いただきたい。

また、「Delphi 10 Seattle」より前の新機能については 2016 年発行のミガロ. テクニカルレポートにある『Delphi/400 最新プログラム文法の活用法』を参照いただきたい。

2. 最新Delphi活用 TIPS

本章では、10 Seattle 以降の新しい Delphi ならびに Delphi/400 を使って開発効率をさらに向上するための TIPS をいくつか紹介する。

なお、10 Seattle で Delphi/400 に正式対応となった「FireDAC 接続」、10 Seattle で追加された IoT や Beacon コンポーネントを使った開発テクニック、10.2 Tokyo で追加された各種クラウドサービスに接続する「Enterprise Connectors」については、それぞれ過去のテクニカルレポートで紹介しているのでそちらを参照いただきたい。

2-1. GetIt パッケージマネージャ（10 Seattle ～）

Delphi の統合開発環境（IDE）の 10 Seattle 以降では、「GetIt パッケージマネージャ」を使用して、Delphi の開発元であるエンバカデロ社が提供する追加コンポーネントや、サードパーティ製品の評価版などを簡単にインストール／アンインストールできる。

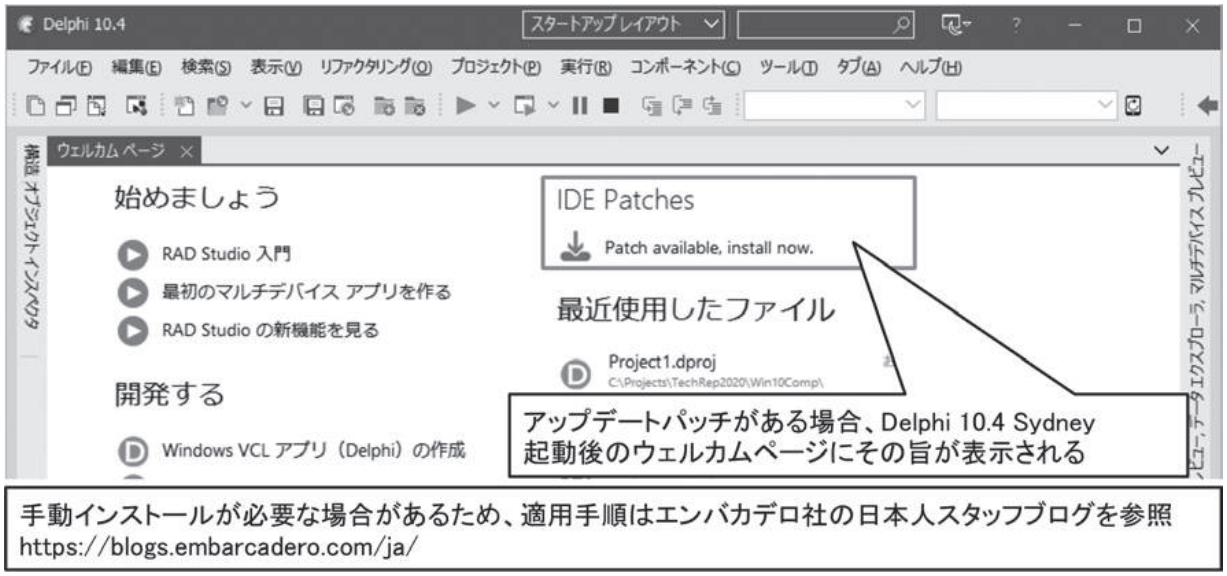
「ツール」メニューに追加された「GetIt パッケージマネージャ」を選択すると、【図 1】のような画面が起動するので、ここに表示されている一覧から目的の項目をインストール／アンインストールする。Delphi のバージョンによって、画面のレイアウトや提供ツールのラインナップは若干異なっている。

また Delphi 10.4 Sydney では、エンバカデロ社が製品の修正パッチを発行した際に、この GetIt パッケージマネージャでもダウンロード可能になっている。

この修正パッチはダウンロードしただけでは適用されずに手動インストールを必要とする場合がある。手元の Delphi 10.4 Sydney を起動した際に、ウェルカムページに【図 2】のような画

図1　GetItパッケージマネージャ

図2　製品アップデートパッチの確認



手動インストールが必要な場合があるため、適用手順はエンバカデロ社の日本人スタッフブログを参照
https://blogs.embarcadero.com/ja/

ソース1

インライン変数宣言①

```
// Delphi 10.4 Sydneyより前は、
// コード開始前のvarブロックで宣言する必要があった
procedure Test1;
var
  I, J: Integer;
begin
  I := 24;
  J := 26;
  ShowMessage(IntToStr(I + J));
end;

//---------------------------------------------------------------------------

// Delphi 10.4 Sydneyでは、
// コード内で直接変数を宣言できるようになった
procedure Test2;
begin
  var I: Integer := 24;
  var J: Integer;
  J := 26;
  ShowMessage(IntToStr(I + J));
end;
```

構文の途中でvar宣言や初期化が可能

面が表示された場合は、エンバカデロ社のホームページにある日本人スタッフブログを参照いただきたい。

2-2. インライン変数宣言（10.4 Sydney）

従来の Delphi のコーディングでは、伝統的な Pascal 言語の規約に従い、すべての変数は関数、プロシージャ、メソッドのコードの開始前に var ブロックで定義する必要があった。

しかし、さらなる柔軟性を提供する新しいインライン変数宣言のコーディングが可能となり、コードブロック内で直接変数を宣言できるようになった【ソース 1】。ここで宣言された変数はそのコードブロック内（「begin ～ end;」の中）でのみ有効で、その外側から参照しようとするとコンパイルエラーになる。【ソース 2】

コードブロック内でのインライン変数宣言がとくに有効となるのが、変数宣言を伴う for ループで、「for var I: Integer := 1 to 10 do ……」といった記述方法が可能である。10.4 Sydney でこのように記述した場合、ループの外側でループ変数を参照しようとするとコンパイルエラーになるので、意図しないループ変数の参照も防げる。

2-3. Language Server Protocol（10.4 Sydney）

Delphi の開発者であれば、IDE でコードを入力しているときに Ctrl ＋スペースキーを押してコード補完を使用したことがあるだろう。

このコード補完を表示する手法をコードインサイトと呼ぶ。Delphi 10.4 Sydney のコードインサイトでは、Language Server Protocol（以下、LSP）が採用されており、LSP はこのコード補完の結果を Delphi 自身ではなく別プロセスで計算する手法をとっている。

従来のバージョンでは Delphi 自身がコードインサイトを行っていたため、Ctrl ＋スペースキーを押してからコード補完が表示されるまでに、コンマ数秒～数十秒の待機時間が必要であった。

しかし Delphi 10.4 Sydney では、待機時間なしでコード補完が可能である（待機時間の間も Delphi を操作できる）。それに加えて、内部フィルタリングロジックの変更によって従来の前方一致だけでなく中間一致にも対応しており、プロパティやメソッドがより見つけやすくなっている。【図 3】

また、別プロセスでコードインサイトを行うもう 1 つの利点として、従来のバージョンでは実行できなかったデバッグ中のコード補完の表示も可能となる点が挙げられる。【図 4】

2-4. プロジェクトオプションの設定項目充実（10 Seattle ～）

IDE の「プロジェクト」メニューにある「オプション」を選択すると、開いているプロジェクトのコンパイル先、アイコン、バージョン情報といったさまざまな設定が可能である。10 Seattle 以降は、このプロジェクトオプションも目覚ましい進化を遂げている。【図 5】

＜高 DPI の認識＞

高 DPI は高精細液晶や高解像度液晶とも呼ばれており、Windows 10 のディスプレイ設定にある「拡大縮小とレイアウト」の「テキスト、アプリ、その他の項目のサイズを変更する」の項目を指す。

対応しているアプリケーションを起動すると、実行 PC のディスプレイの DPI の設定が 100％でない場合にフォームやフォーム内の各項目のサイズが自動拡張され、解像度の高いディスプレイでも画面が小さくなりすぎずに表示できる。【図 6】

Delphi では、10 Seattle からこの高 DPI 設定に対応しており、プロジェクトオプションの「アプリケーション」の項目から、マニフェストの設定を変更することで指定が可能となっている。バージョンごとに設定できる項目や項目名が少しずつ異なるので、バージョン間の差異については【図 7】に記載する。

＜管理者権限の有効化＞

コンパイルしたアプリケーションをダブルクリックなどで実行する場合に、実行時の権限を指定できる。

プロジェクトオプションの「アプリケーション」の項目から、前述の高 DPI と同様に、マニフェストの設定にある実行レベルを変更することで指定できる。【図 5】

デフォルトの「インボーカとして」を選択している場合は通常権限（親プロセスと同じ権限）で実行し、「使用可（最高）」を選択している場合はユーザーが取得できる最も高い権限で実行する。

そして「管理者権限が必要」を選択している場合は、アプリケーションを強制的に「管理者として実行」で実行させる。こちらも前述の高 DPI と同様、バージョンごとに設定できる項目や項目名が少しずつ異なるので、バージョン間の差異については【図 7】に記載する。

なお、「管理者権限が必要」の設定でコンパイルしたアプリケーションを Delphi 上でデバッグするには、Delphi 自身も管理者として起動しておく必要がある。【図 8】

＜カスタムスタイルの機能拡張＞（10.4 Sydney）

XE3 以降のバージョンでは、プロジェクトオプションでスタイルを設定することで、コンパイルされるアプリケーションの見た目を変更できる【図 9】。10.4 Sydney ではこのスタイル機能が進化し、同一アプリケーション内でのフォームやコントロール（Edit や Button など）単位でスタイルを変更可能になった。

プロジェクトオプションでチェックを入れたカスタムスタイルの中から、対象のフォームやコントロールの「StyleName」プロパティに、そのスタイル名を指定することで反映される。【図 10】

2-5. Windows 10 サポートと IDE の機能強化（10 Seattle ～）

Delphi 10 Seattle からは Windows 10 が正式にサポートされ、専用のユーザーインターフェースに対応したコンポーネントも追加されている。また、XE7 以前のバージョンと比較して IDE で利用可能なメモリが約 2 倍に拡張されており、大規模プログラムでも安定した IDE 環境で開発可能になっている。

加えて、非ビジュアルコンポーネントの表示切り替え機能が追加され、フォーム／ユニットの切り替えの右側に追加されたボタンを押すと、【図 11】のように Table や Query などの画面上に見えないコンポーネントの表示／非表示を切り替えられるようになった。これにより、画面項目の配置の微調整がさらに容易になっている。

ソース2

インライン変数宣言②

```
// コード内で直接宣言した変数は、そのbegin～end内でのみ有効
procedure Test3;
begin
  begin
    var I: Integer := 24;
    var J: Integer;
    J := 26;
  end;
  ShowMessage(IntToStr(I + J));
end;
```

これは変数宣言されたbegin～endの
外側にあるため、コンパイルエラーになる

図3　Language Server Protocol（LSP）

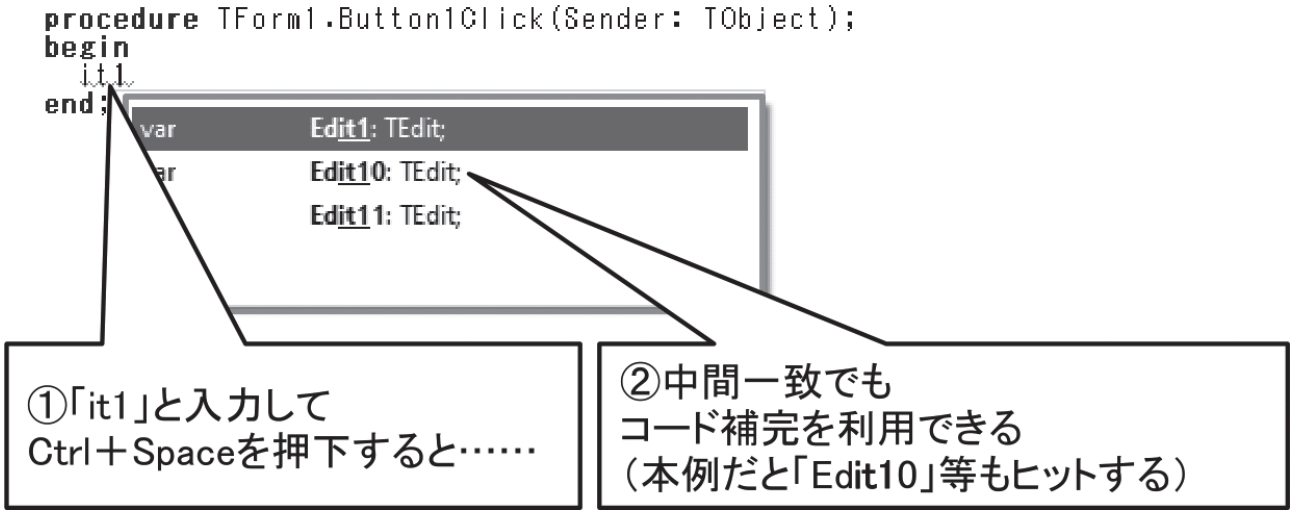
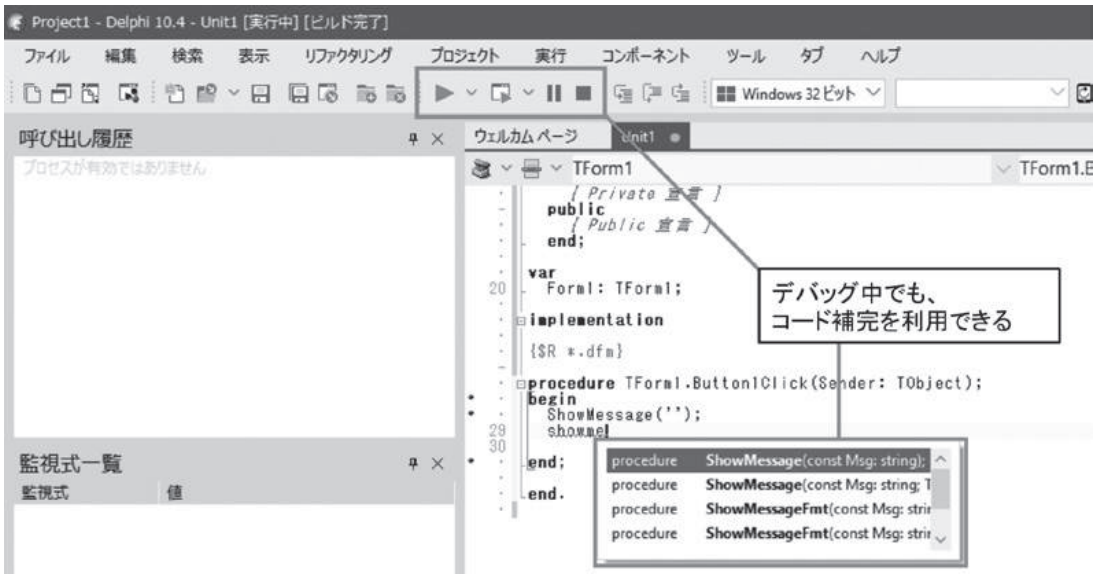


図4　Language Server Protocol（LSP）



2-6. Windows 10 ライクな新コンポーネント群（10 Seattle ～）

Windows 10 の正式サポートに伴い、その動作を模倣するコントロールがいくつか追加されている。これらのコンポーネント群はツールパレットの「Windows 10」パレットページに追加されているが、実際には Windows 10 ネイティブのコンポーネントではないので、古い Windows OS で実行しても動作できる。【図 12】

以下に、代表的なものをいくつか紹介する。【図 13】

・TSplitView

開いたり閉じたりできる他コントロールのコンテナで、FireMonkey の TMultiView と似た動作をする。Open メソッドおよび Close メソッドで、Windows 10 やスマートデバイスの画面のようにアニメーションでパネルが開閉する動作を再現できる。

・TSearchBox

TEdit の拡張で、Enter キーを押すか、虫眼鏡のボタンを押すことで「OnInvokeSearch」イベントが走り、検索画面への遷移といった処理を組み込める。

・TToggleSwitch

Windows 10 の設定画面などでよく見かける「オン」と「オフ」を切り替えるスイッチの動作を実現する。State プロパティを「tssOn」に設定すればオンに、「tssOff」に設定すればオフに切り替わる。

・TActivityIndicator

Windows 10 で各種処理中に表示されるインジケータ（ぐるぐる回るアイコン）を画面上の任意の位置に表示させる。

上記以外にも、Delphi 10.2 Tokyo 以降では Windows 10 ライクな日付時刻の表示や指定を行う TCalendarView ／ TDatePicker ／ TTimePicker ／ TCalendarPicker などが追加されている。

本項で紹介したコンポーネントはいずれも、【図 13】に記載した簡単なプロパティの設定や命令だけで扱えるので、ぜひ一度使い心地をご確認いただきたい。

2-7. TTitleBarPanel と CustomTitleBar（10.4 Sydney）

最新の Delphi 10.4 Sydney では、新しく追加されたコンポーネント「TTitleBarPanel」、および TForm に新たに追加されたプロパティ「CustomTitleBar」を利用することで、最新のオフィスソフトやブラウザのようにタイトルバーをカスタマイズできるようになった。

前項の Windows 10 ライクな各コンポーネントと比較すると少々複雑なので、使用手順を以下に解説する。

まず対象のフォームに TTitleBarPanel を配置し、タイトルバー内に表示させたい Button や Edit などを配置する。【図 14】

TTitleBarPanel の設定が完了したら、親となる TForm の設定を行う。CustomTitleBar プロパティを開き、「Control」に今作成した TTitleBarPanel を、「Enabled」に True を指定すると、フォームと TTitleBarPanel の紐付けが行われる。【図 15】【図 16】

アプリケーションをコンパイルして実行すると、【図 17】のような画面が表示され、タイトルバーがカスタマイズされていることが確認できる。

2-8. TEdgeBrowser（10.4 Sydney）

Delphi のフォーム上に TWebBrowser コンポーネントを配置すると、実行するアプリケーション内で Web ブラウザの画面を表示できる。しかし、従来この TWebBrowser 内で表示されるブラウザは IE パーソナリティ（Internet Explorer 7 ベース）として動作していた。

最新の Delphi 10.4 Sydney では新しく「TEdgeBrowser」コンポーネントが登場し、これを利用することで Edge パーソナリティによるブラウザを画面に表示できるようになった。【図 18】【ソース 3】

2020 年 9 月現在、Microsoft Edge は Windows の OS 標準コンポーネントになっていないため、TEdgeBrowser を動作させるには以下の 2 つのアイテムを PC にインストールする必要がある。

① Microsoft Edge Chromium ベースのブラウザ（Canary channel version）
② WebView2 コントロールの DLL

このうち①については、2020 年 9 月現在、通常の Edge の最新バージョンよりも新しい、Canary Channel Version（毎日更新されるデベロッパー向けビルド）の Edge を Microsoft のサイトからダウンロードし、インストールする必要がある。【図 19】

一方の②は、GetIt パッケージマネージャからインストールしたあと、【図 20】【図 21】のようなフォルダが開くので、対応 bit 数（コンパイルするアプリケーションが 32bit アプリケーションなら「win32」、64bit アプリケーションなら「win64」）のサブフォルダを開き、その中の「WebView2Loader.dll」を取得しておく。

IDE でアプリケーションを開発し、完成したアプリケーションと同じフォルダに「WebView2Loader.dll」をコピーすることで、【図 18】のように TEdgeBrowser でブラウザが画面に表示される。

なお、従来の TWebBrowser コンポーネントも機能拡張し、Edge パーソナリティに対応している。上記の TEdgeBrowser を動作させる①②のアイテムと「WebView2Loader.dll」が必要である点は変わらないが、その条件を満たしていれば、TWebBrowser コンポーネントに追加された SelectedEngine プロパティを、デフォルトの「IEOnly」から「EdgeOnly」に変更することで、Edge パーソナリティで TWebBrowser のブラウザを表示できる。【図 22】

また SelectedEngine プロパティを「EdgeIfAvailable」に変更すると、使用可能な場合のみ Edge パーソナリティを使用し、使用不可の場合は従来の IE パーソナリティでブラウザを表示する。どちらのブラウザが使用されているかは、TWebBrowser の ActiveEngine プロパティの値で確認できる。

注意点としては、TWebBrowser の各プロパティ・メソッド・イベントは従来の IE パーソナリティに準拠している点が挙げられる。Edge パーソナリティで使用している場合は一部のプロパティ・メソッド・イベントが互換してお

図5　Delphi 10.4 Sydneyのプロジェクトオプション

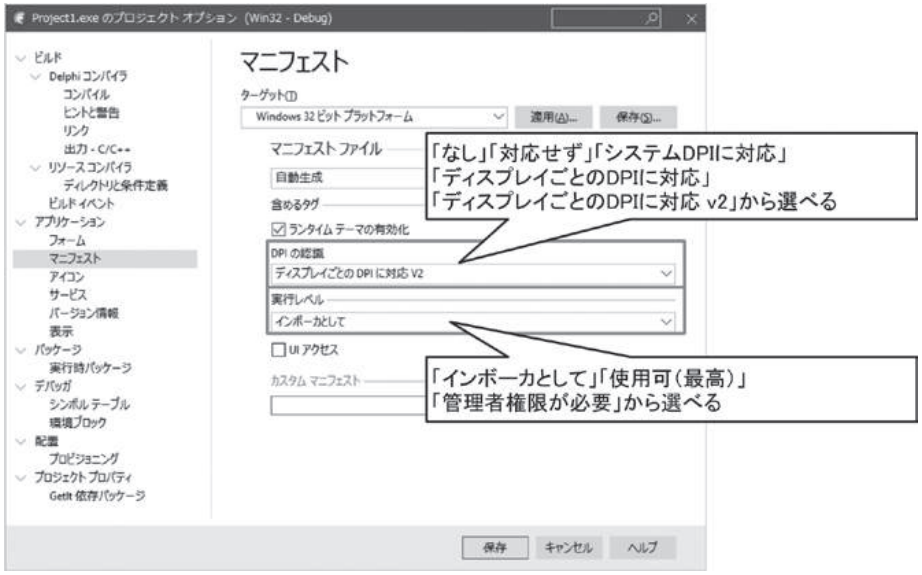


図6-1　高DPI設定対応

図6-2　高DPI設定対応

らず、プロパティの取得結果が変わったり、メソッドで例外が発生したりする場合がある。【図 23】

Delphi 10.4 Sydney を取得したあと、TWebBrowser から TEdgeBrowser への移行が可能な場合は、段階的に移行を検討いただきたい。

3. まとめ

株式会社ミガロ．では、開発元である仏 SystemObjects 社の間で日本総代理店契約を締結し、2000 年 9 月に Delphi/400 V5 の販売を開始してから今年で 20 周年を迎えた。

Delphi/400 は古いバージョンのまま互換の範囲で使い続けられることも魅力ではあるが、本稿で紹介した新機能を利用することで、より多彩な機能が活用できるようになる。

本稿の記述を参考に、最新バージョンの Delphi ならびに Delphi/400 に関心をもっていただければ幸いである。

M

図7　Delphi10シリーズの高DPI・実行レベルの設定

<高DPI対応の設定>

バージョン	XE7以前	10 Seattle	10.2 Tokyo	10.4 Sydney
プロジェクトオプション内の設定画面の場所	なし	アプリケーション → マニフェスト ファイル		アプリケーション → マニフェスト ファイル → DPI の認識
設定項目	設定不可	「高 DPIの有効化」チェックのオン／オフで切り替え（オンの場合「ディスプレイごとのDPIに対応」になる）		「なし」 「対応せず」 「システムDPIに対応」 「ディスプレイごとのDPIに対応」 「ディスプレイごとのDPIに対応 v2」から選択

※実行時の結果（実際にどのDPI設定になっているか）は、タスクマネージャの「詳細」タブから確認可能

<管理者権限の要求設定>

バージョン	XE7以前	10 Seattle	10.2 Tokyo	10.4 Sydney
プロジェクトオプション内の設定画面の場所	なし	アプリケーション → マニフェスト ファイル	アプリケーション → マニフェスト ファイル → 実行レベル	
設定項目	設定不可	「管理者権限の有効化」チェックのオン／オフで切り替え（オンの場合「管理者権限が必要」になる）	「インポーカとして」 「使用可（最高）」 「管理者権限が必要」から選択	

図8　管理者権限指定時の注意

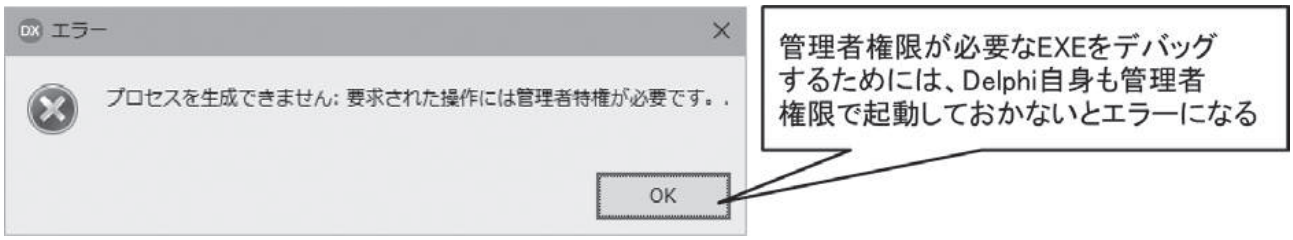


図9　カスタムスタイルの機能拡張

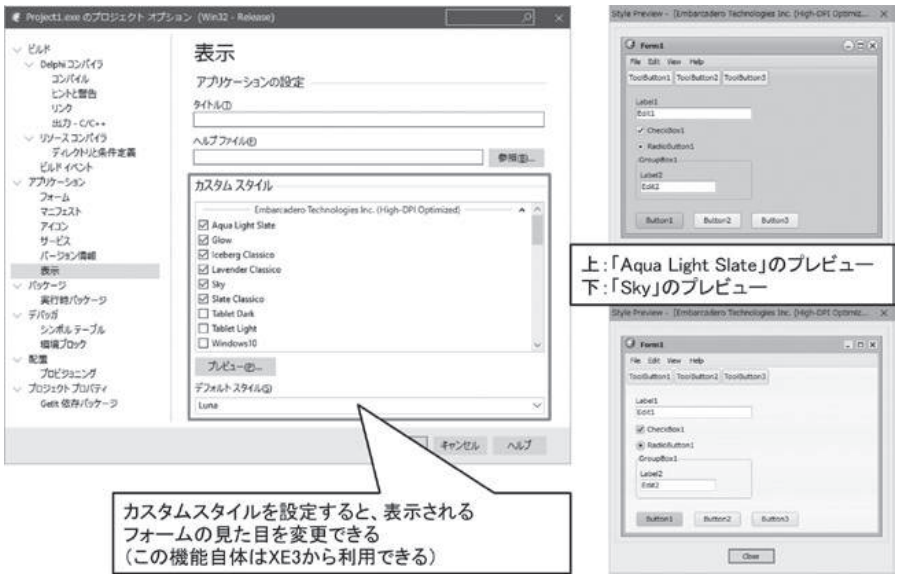


図10　カスタムスタイルの機能拡張

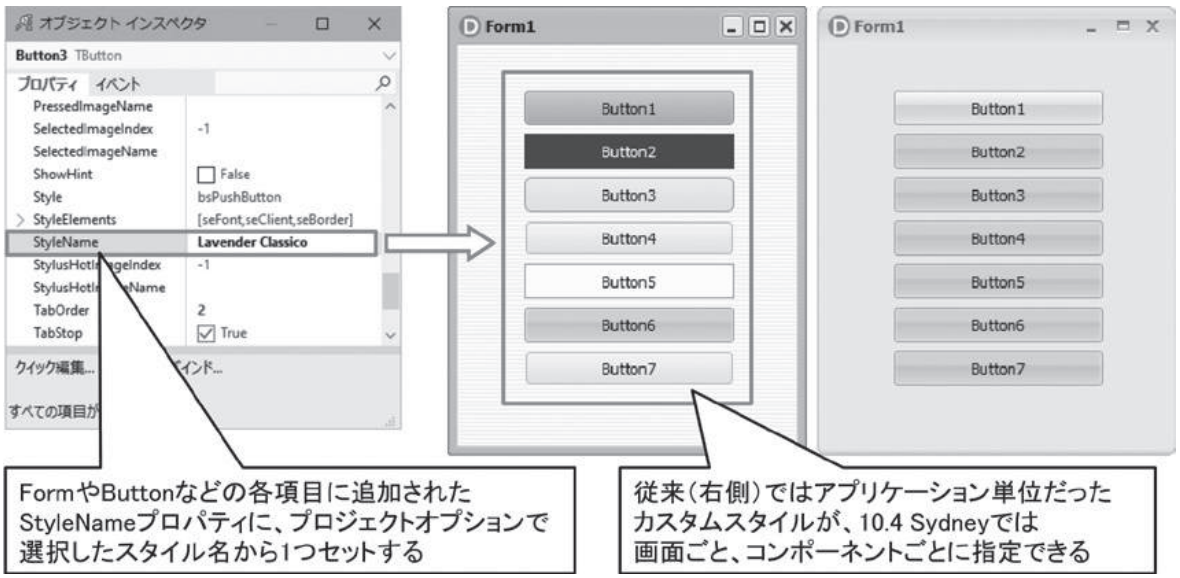


図11　非ビジュアルコンポーネントの表示切替

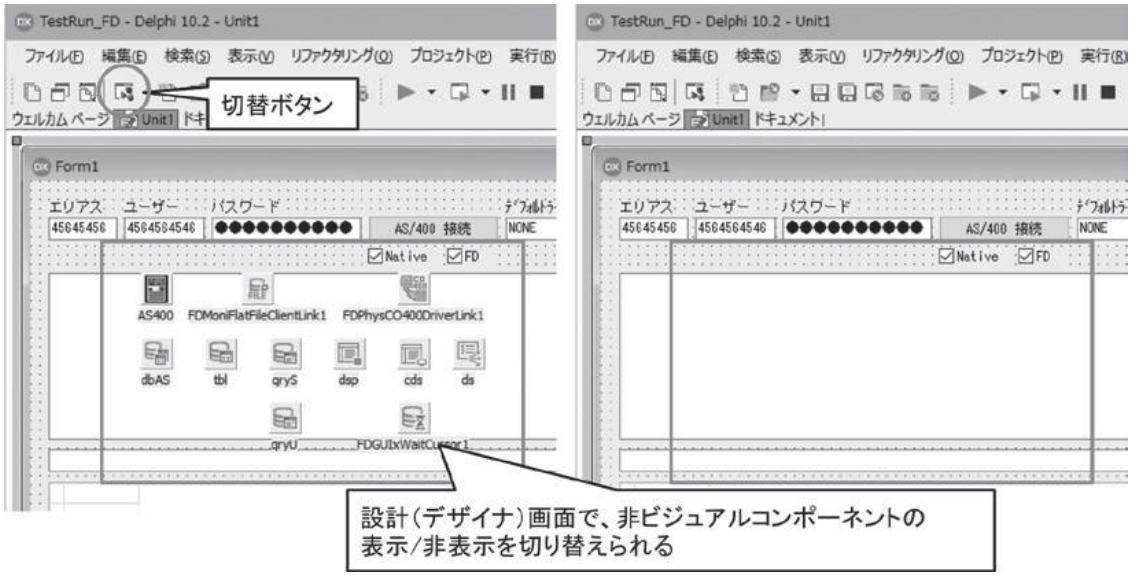


図12　Windows10の見た目を提供するコンポーネント

図13　Windows10の見た目を提供するコンポーネント

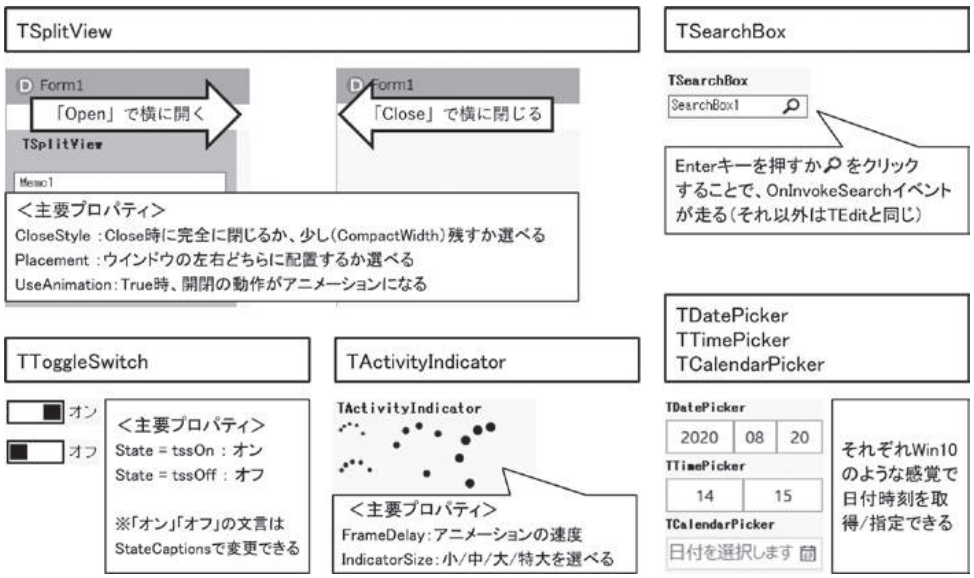


図14　TTitleBarPanelの設定①

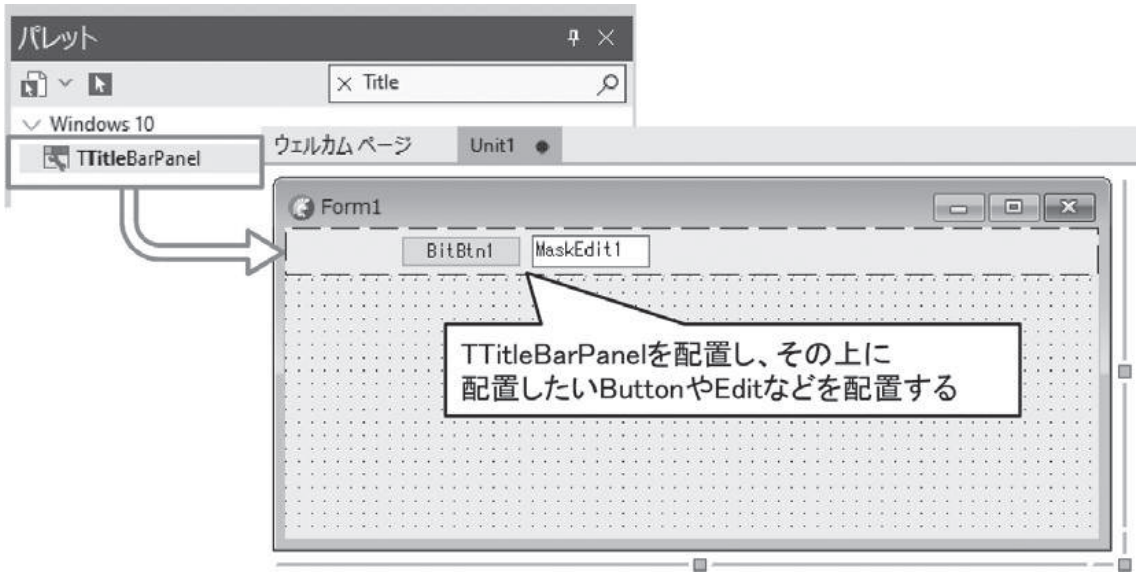


図15　フォームのCustomTitleBarプロパティの指定

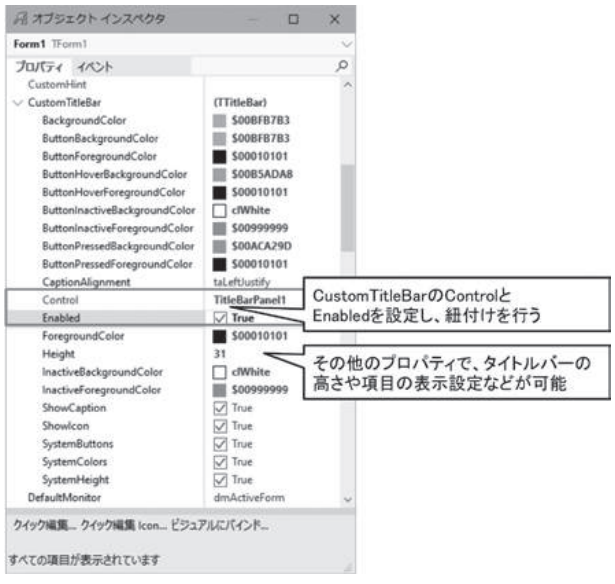


図16　TTitleBarPanelの設定②

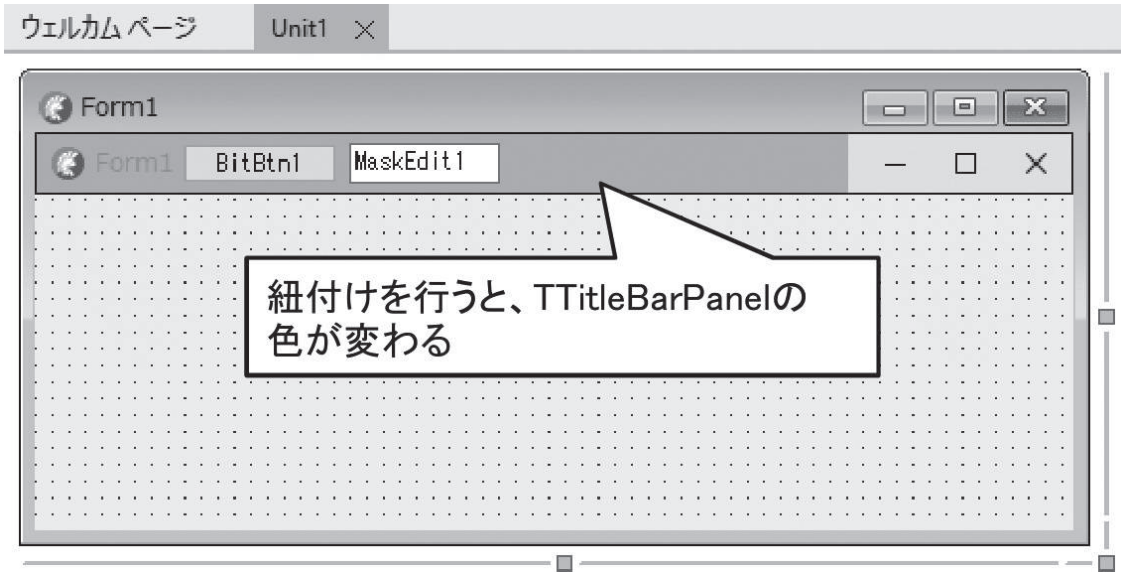


図17　TTitleBarPanelの設定～実行結果

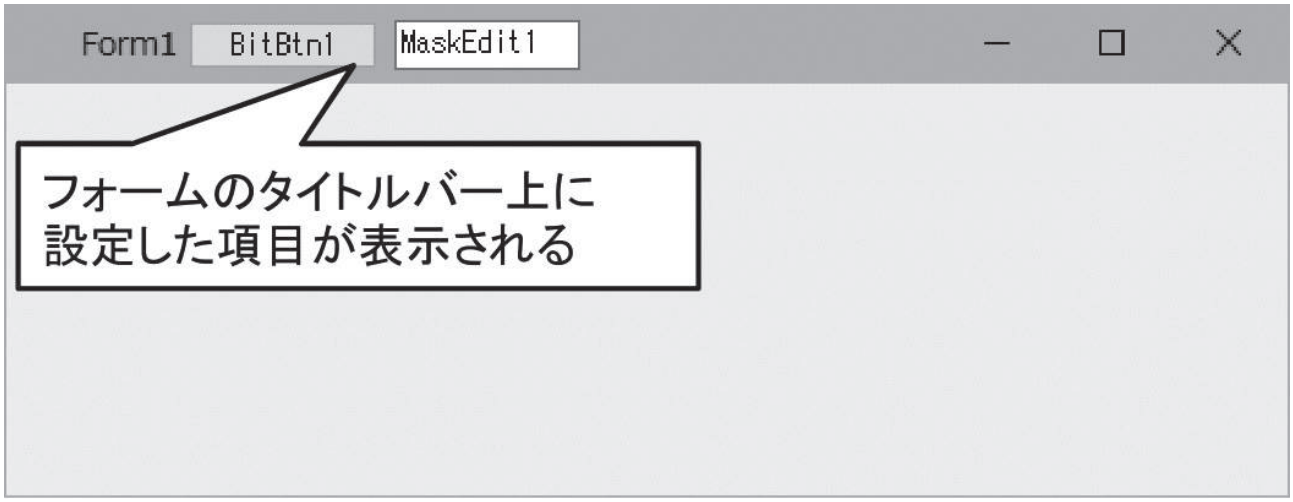
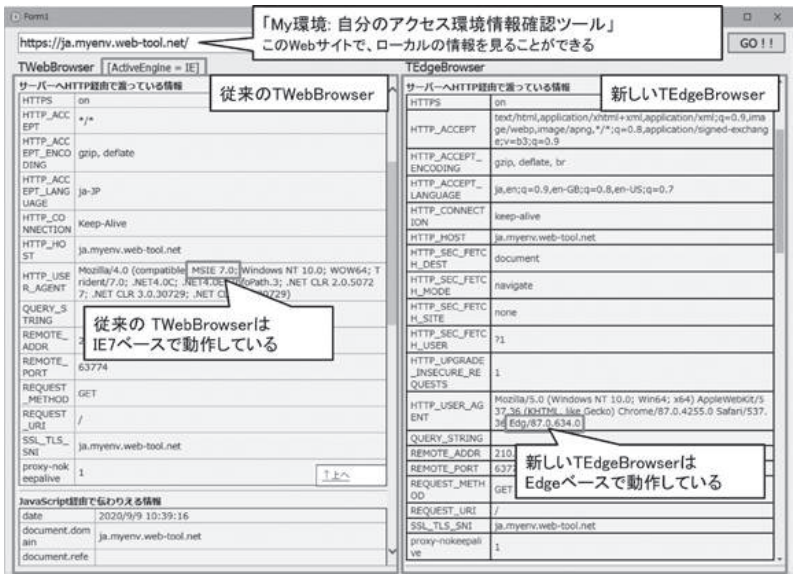


図18　新しいTEdgeBrowser



ソース3

TWebBrowser・TEdgeBrowserでURLを開く

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  // それぞれのコンポーネントでURLを開く
  EdgeBrowser1.Navigate(Edit1.Text);
  WebBrowser1.Navigate(Edit1.Text);
end;
```

どちらも「Navigate」メソッドで
URLにジャンプする

図19　TEdgeBrowserの使用準備①

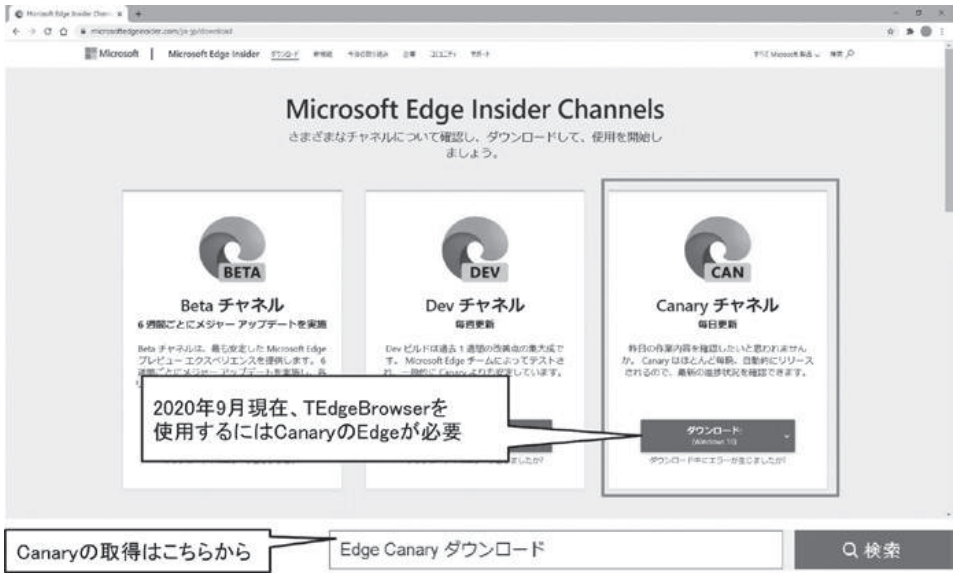


図20　TEdgeBrowserの使用準備②-1

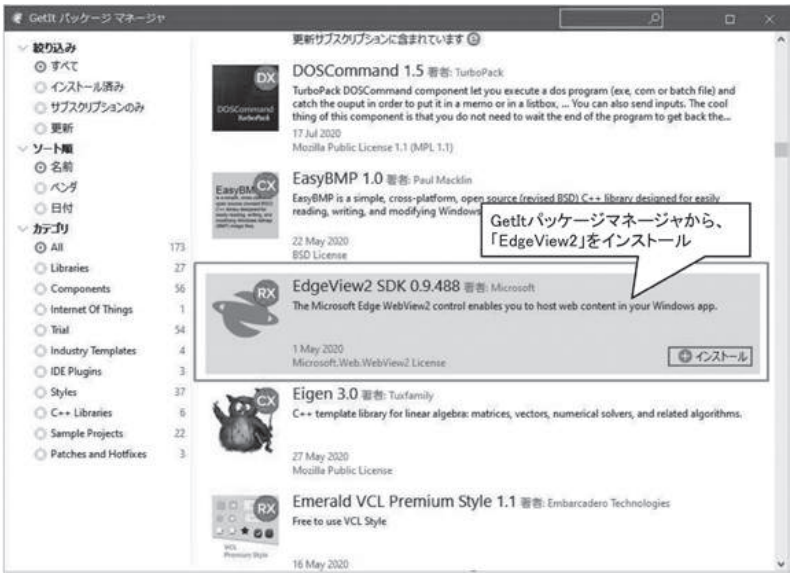


図21　TEdgeBrowserの使用準備②-2

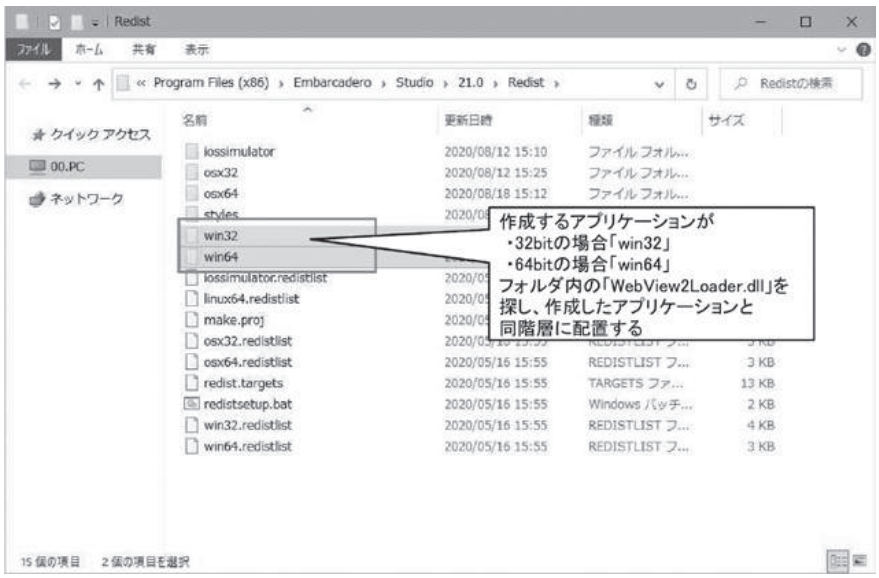


図22　TWebBrowserもEdgeベースで表示可能

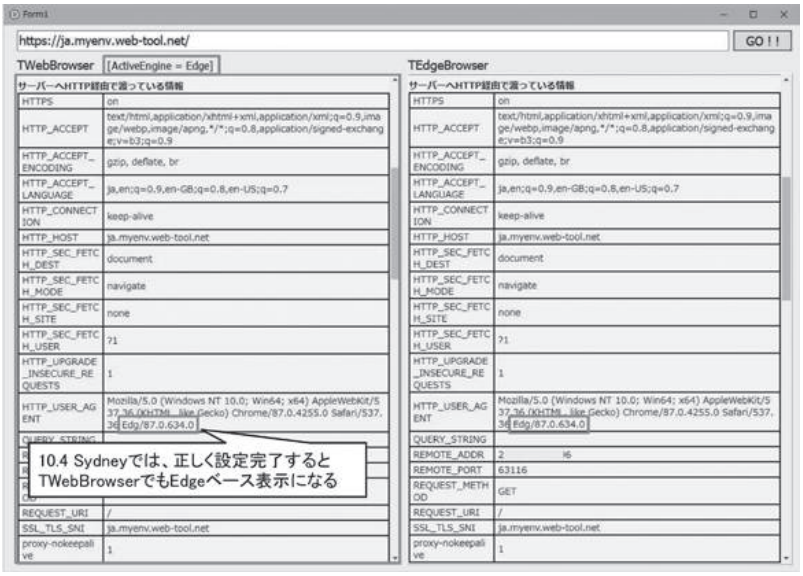


図23　TWebBrowserではEdge互換のない項目がある

國元 祐二

株式会社ミガロ.

RAD事業部 技術支援課

[SmartPad4i] Eclipse(Cobos4i)を使用したSmartPad4i開発術

略歴
1979年3月27日生まれ
2002年3月 追手門学院大学 文学部アジア文化学科卒業
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
SmartPad4i（JC/400）、Business4Mobile、Valenceの製品試験やサポート業務、導入支援などを行っている。

1. はじめに

オープン系の開発ではWebアプリケーションを作成する際に、開発言語としてJavaやPHPが選択されることが少なくない。JavaやPHPの開発では通常、開発を効率化するために統合開発環境を使用する。

統合開発環境にはさまざまな製品があるが、中でもオープンソースのEclipseが有名である。Eclipseは1990年代後半から開発され、2001年11月29日にバージョン1.0がリリースされた。2020年8月時点では、2020年6月17日にリリースされたバージョンEclipse IDE 2020-06が最新版で、長きにわたり使用されている統合開発環境である。

Eclipseの特徴は、機能拡張用のソフトウェア(プラグイン)を追加インストールすることで、さまざまな機能を組み込めるように設計されている点である。プラグインを導入することで、JavaやPHPだけでなく、さまざまな言語の開発環境として使用できる。

弊社では、Cobos4iという製品のリリースを予定している。Cobos4iとはEclipseのプラグインを活用し、SmartPad4iアプリケーションでの開発をより効率化する統合開発環境である。本稿では、Cobos4iの特徴やメリットを紹介する。

1.1. SmartPad4iでの開発手法

Cobos4iを解説する前に、まずはSmartPad4iについて簡単に説明しておこう。なお、詳しい説明については2014年発行のミガロ.テクニカルレポートに『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 - 2. JC/400、SmartPad4iにおける画面作成の基本』が紹介されているので、こちらも参考にしていただきたい。

SmartPad4iは、RPG／COBOLスキルでWebシステムを構築できるツールである。RPG／COBOLの資産や知識を活用して、HTML5やJavaScriptなどの最新Web技術を組み合わせた（Webまたはモバイルⅰ）アプリケーションが作成できる。

IBM iの5250アプリケーションでは、DSPF（画面ファイル）で画面を定義するが、SmartPad4iでは、HTMLで画面を定義する。SmartPad4iはHTMLで画面を作成できるので、グリーン画面での80桁×24行の制限がない。画像やCSS、JavaScriptを使用することで、わかりやすく、便利なインターフェースを作成できる。

HTMLには、入出力項目や要素へユニークなid名を設定するのがルールである。DSPFでのフィールドIDと同様、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。

作成したHTMLをSmartPad4i 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のスケルトンプログラムを配布する。配布された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

図1　SmartPad4iアプリケーション開発手法

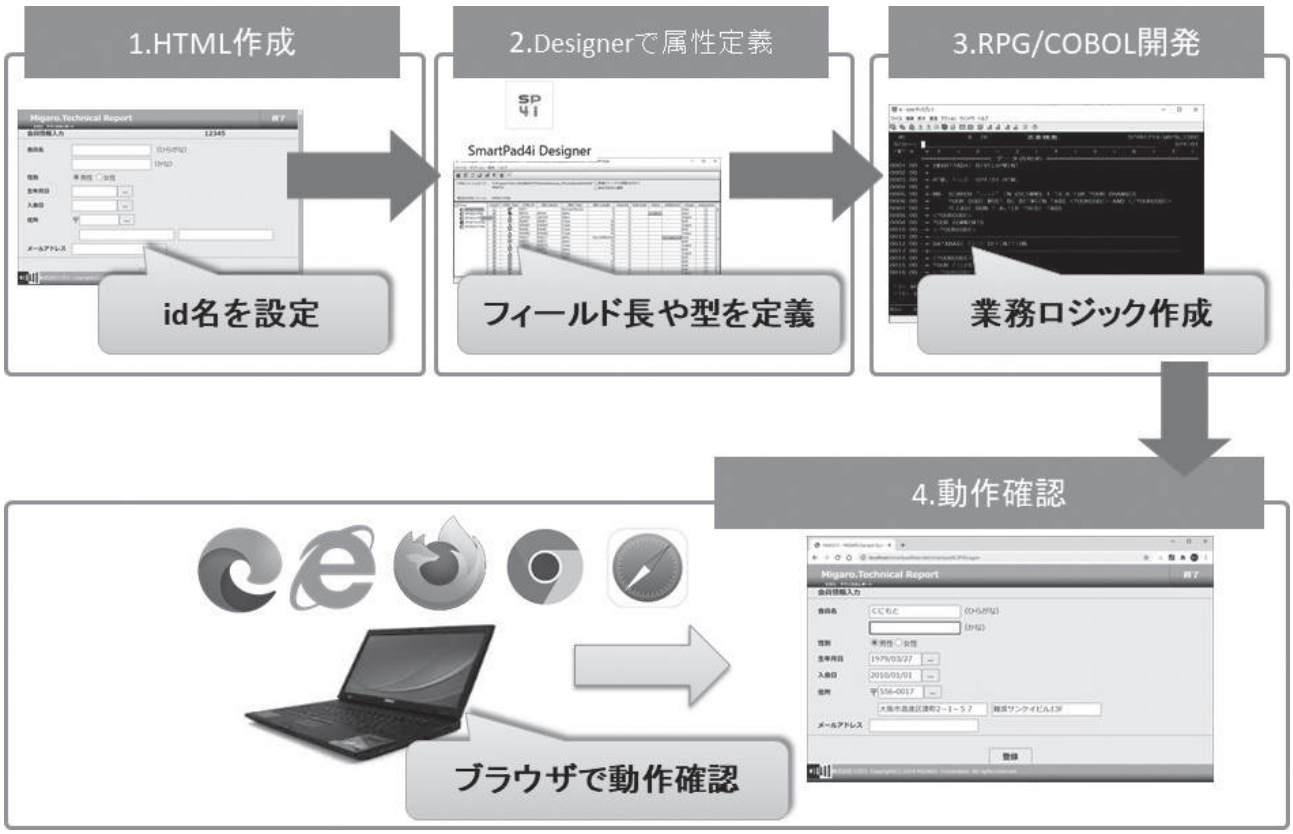
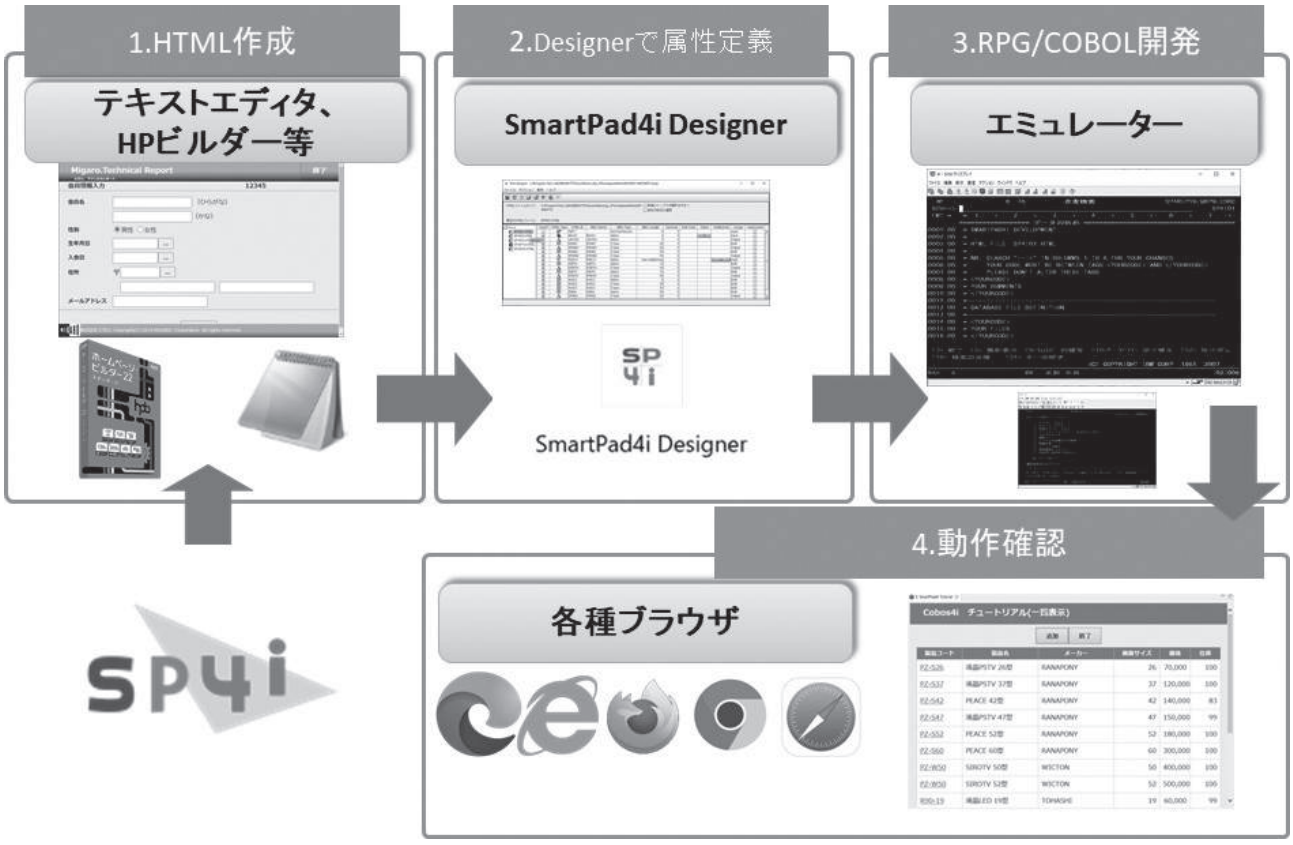


図2　SmartPad4i開発で使用するソフトウェア



成できる。【図 1】

前述のとおり、SmartPad4i を使用すると、容易に Web アプリケーションの開発が可能である。しかし Cobos4i を使用すると、より少ないステップで開発できる。Cobos4i については、次の 2. から詳しく説明していく。

2. Cobos4iとは?

Cobos4i は、Eclipse をベースとした SmartPad4i アプリケーションの統合開発環境である。以下のように 3 点の特徴やメリットがある。

①開発環境を簡単に導入できる

Apache Tomcat が採用されたことにより、環境構築を簡易化できる。

②開発効率が向上する

Eclipse 上で SmartPad4i アプリケーション開発の工程がすべて実行できる。開発に使用するソフトウェアを切り替える必要がないため、開発効率が向上する。

③ Eclipse のプラグインが使用できる

HTML の作成やソース管理などの機能がプラグインで使用可能になる。

Cobos4i の特徴やメリットについて、次で詳しく紹介する。

3. Cobos4iの特徴

3.1. 開発環境をすぐに導入

従来の SmartPad4i で開発環境を構築する場合には、WebSphere Application Server（以下、WAS）の導入が必要になる。

WAS は、IBM i または Windows 上に導入可能なアプリケーションサーバーである。IBM i に WAS を導入するには、累積 PTF 等の導入を必要とする場合が多く、IBM i に運用停止時間が発生するため、導入には細心の注意が必要となる。またインストールや環境構築の知識も必要なので、WAS 環境構築には時間を要する。

IBM i 環境に比べて比較的導入が容易な Windows 環境に WAS を導入する場合でも、少なくとも半日はインストール時間が必要となる。

それに対して Cobos4i は、WAS 環境構築と SmartPad4i のインストール作業が不要である。SmartPad4i の環境を作成する場合は WAS の知識が必須であったが、Cobos4i は Apache Tomcat が同梱されており、Windows 端末に配置すると SmartPad4i の開発環境が構築できる。

また Apache Tomcat はオープン系の開発でもよく使用されるため、インターネット上の情報量も多い。

従来は半日ほど必要だった環境構築が、Cobos4i では 1 時間以内で完了する。開発環境構築に時間がかからず、WAS と SmartPad4i の導入知識が不要なことも Cobos4i の魅力である。

3.2 Apache Tomcat のサポート

前項でも記載しているが、Cobos4i は、アプリケーションサーバーに Apache Tomcat 採用している。Apache Tomcat は、Java Servlet や Java Server Pages（JSP）を実行するための Web コンテナである。そのため、Java Servlet の技術で作成されている SmartPad4i アプリケーションを実行できる。さらに、Apache License 2.0 を採用したオープンソースソフトウェアなので、無償で利用可能である。

弊社では SmartPad4i の実行環境として、これまでは WAS のみをサポートしてきたが、Cobos4i のリリースに合わせて、Apache Tomcat 環境も追加する。

3.3. すべての開発手順を Eclipse 上で実施

SmartPad4i アプリケーションの作成には、それぞれの開発ステップで、別々のソフトウェアを使用する必要がある。【図 2】

たとえば HTML の作成では、ホームページビルダーや Dreamweaver、テキストエディタ等のソフトウェアを使用することになる。作成した HTML の属性を定義するには、Windows 環境で動作する SmartPad4i Designer を使用する。

また RPG／COBOL 開発では、IBM i Access Client Solutions（ACS）や IBM i Access for Windows（クライアントアクセス）等の 5250 エミュレータが必要である。さらに作成した SmartPad4i アプリケーションの動作を確認するには、各種ブラウザを使用する。このように従来の SmartPad4i アプリケーション開発では、さまざまなソフトウェアを使用する必要があった。

しかし Cobos4i では、開発手順のすべてのステップを Eclipse 環境で作業できる。すべての手順を Eclipse で行えるため、開発効率が非常によい。【図 3】

3.4. Cobos4i Designer

Cobos4i では、Eclipse のプラグインである Web ページ・エディターや HTML エディターを使用して HTML を作成できる。従来の SmartPad4i Designer で行っていた、入出力フィールドの型や長さを指定する HTML の属性定義については、Cobos4i Designer を利用する。

Cobos4i Designer で属性を定義し、IBM i に RPG／COBOL のスケルトンプログラムと I/O 定義ファイルを配布すると同時に、開発者のローカル PC にある Apache Tomcat サーバーへ定義ファイルを展開する。【図 4】

また Cobos4i Designer は、SmartPad4i Designer で作成した .jdp ファイル（プロジェクトファイル）を読み込めるので、過去に作成した SmartPad4i アプリケーションも Cobos4i 環境へ簡単に移行できる。

そのため従来の SmartPad4i で開発してきたユーザーも、Cobos4i での開発に移行しやすい。Cobos4i を使用することで開発効率が向上するので、ぜひ使用を検討いただきたい。

3.5. Cobos4i RPG／COBOL の開発とコンパイル

Cobos4i では RPG Editor／COBOL Editor を使用して、RPG／COBOL プログラムを開発できる。

まず IBM i へ配布した RPG／COBOL のスケルトンプログラムを、Cobos4i の機能を使用してローカル PC 上にダウンロードする。

次に、ダウンロードしたプログラムを Cobos4i に含まれる RPG Editor／COBOL Editor を使用して業務ロジックを追加する。

業務ロジックを追加後、Cobos4i の機能を使用してコンパイルする。Cobos4i では、5250 エミュレータを使用せずに、

図3　Cobos4iの開発はすべてEclipse上で作業可能

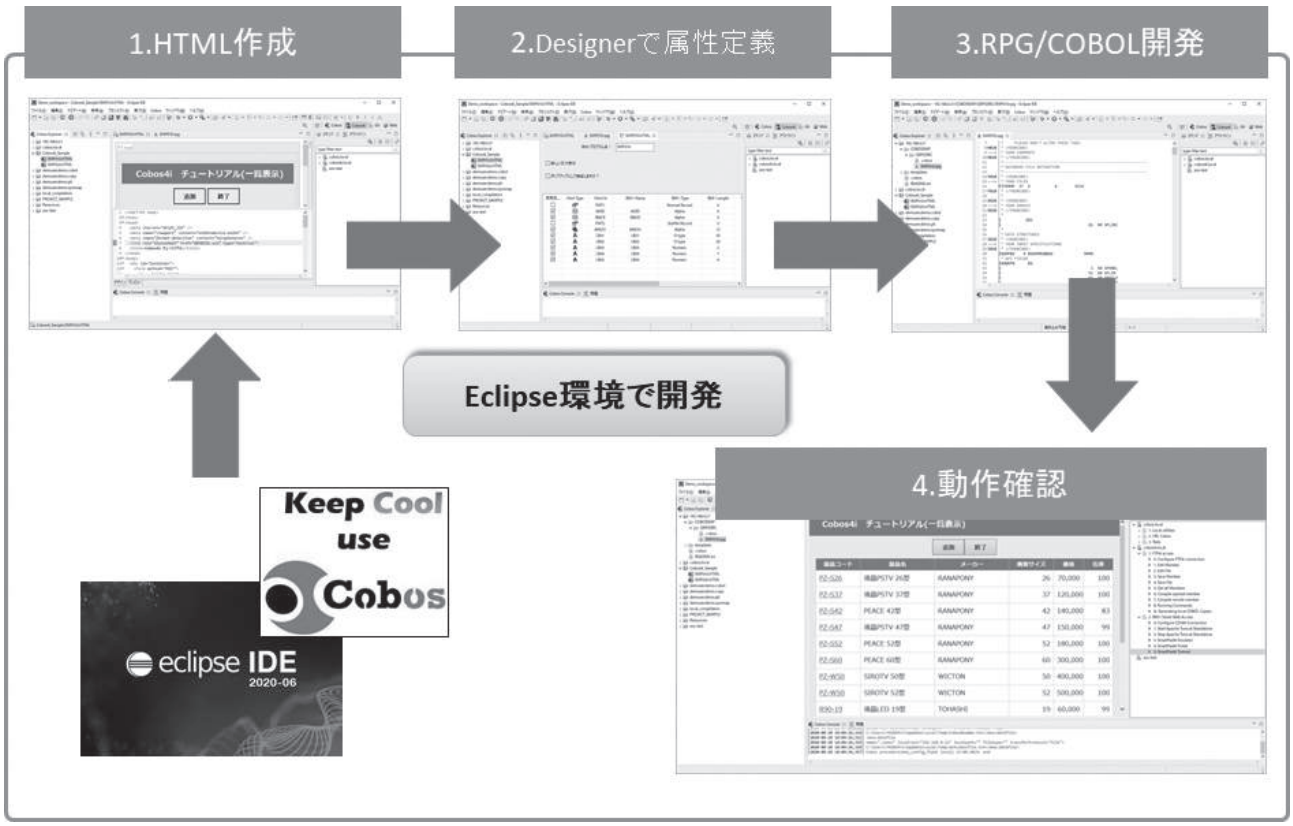
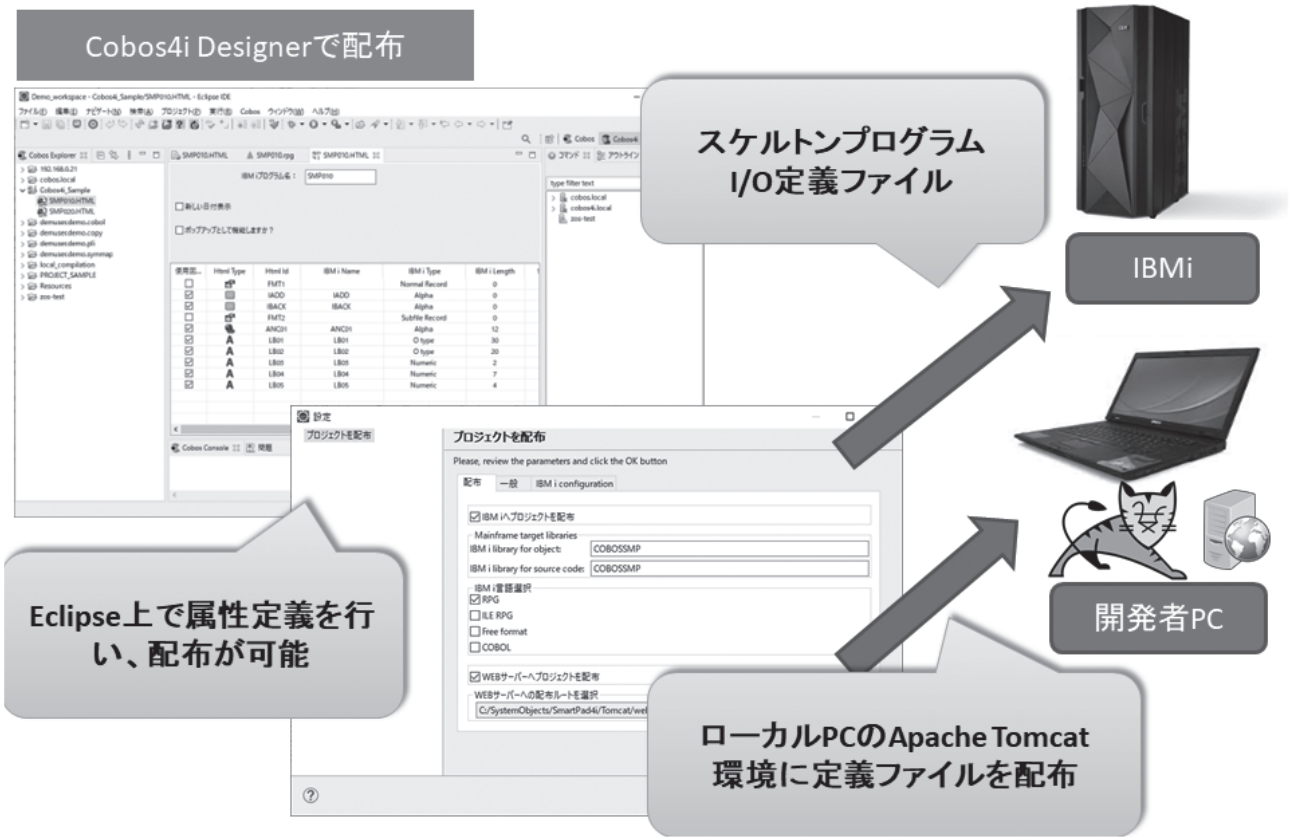


図4　Cobos4i Designer



Eclipse 環境からパラメータを指定するだけでコンパイルできる。【図 5】

3.6. Cobos4i SmartPad4i アプリケーションの実行

Cobos4i で作成した SmartPad4i アプリケーションは、ローカル PC 環境で動作確認できる。Eclipse から Apache Tomcat を起動後、SmartPad4i のプログラム実行画面を開き、実行するアプリケーションプログラムが配置されているライブラリーやプログラム名を入力することで、SmartPad4i アプリケーションをローカル PC 上の環境で動作させられる。【図 6】

従来の開発手法では、開発したアプリケーションの動作確認にはブラウザを起動させる必要があったが、Cobos4i では Eclipse 環境上で動作確認できるので非常に便利である。

4. Eclipseのプラグイン

4.1. Eclipse の Web ページ・エディター

Eclipse プラグインが利用できるのは、Cobos4i の大きな魅力である。たとえば HTML の開発には、Eclipse の Web ページ・エディターが使用できる。

Web ページ・エディターは、WYSIWYG（*1）およびエディターにより HTML を開発できるツールで、HTML 画面を表示しながら HTML ソースを確認できる。【図 7】

Web ページ・エディターは使用前に、パースペクティブやビューを設定すると使いやすい。Eclipse のパースペクティブは、Eclipse の見た目や見え方である。パースペクティブを変更することで、【図 7】のアウトラインを表示できる。

Web パースペクティブを開くには、Eclipse を起動後、上部メニューの「ウインドウ」>「パースペクティブ」>「パースペクティブを開く」>「その他」で、「パースペクティブを開く」のウインドウを表示後、「Web」を選択して、開くボタンを押下する。【図 8】

またパレットのビューを使用すると、ドラッグ＆ドロップで HTML の要素を作成できる。パレットのビューを開くには、上部メニューの「ウインドウ」>「ビューの表示」>「その他」で、「ビューの表示」のウインドウを表示後、「パレット」を選択して、開くボタンを押下する。【図 9】

Web ページ・エディターは HTML 作成時にとても便利なので、ぜひ活用いただきたい。

4.2. Eclipse の EGit

Cobos4i では、Eclipse のプラグイン EGit を使用してソースファイルを管理することも可能である。

ソースファイルを管理するソフトウェアは多数存在するが、シェアを伸ばしているオープンソースの Git を取り上げたい。

Git は分散型バージョン管理システムの 1 つで、Linux のソースコードを効果的に管理するために開発されたソフトウェアである。

バージョン管理システムでは、リポジトリと呼ぶ概念がある。リポジトリとは、ファイルやディレクトリの履歴を管理する場所を指す。Git には、「リモートリポジトリ」と「ローカルリポジトリ」という 2 つの概念がある。

リモートリポジトリはサーバーに配置され、複数人で共有する。ローカルリポジトリは、各ユーザーが自身の PC 内でバージョン管理するために、ローカル PC 上に配置する。Git は、各ユーザーがそれぞれのローカル PC 上にリポジトリをもてる点が最大の特徴である。

Git は、IBM i のライセンスプログラム 5733-OPS「オープンソース for IBM i」のオプション 6 に含まれているので、IBM i 上にインストールして環境を構築することも可能である。各ユーザーがローカル PC 上にリポジトリをもち、IBM i のリモートリポジトリで管理するような開発環境も作成できる。【図 10】

Cobos4i で Git を使用するには、Eclipse プラグインの EGit をインストールする必要がある。インストール手順としてはまず、Eclipse のメニュー「ヘルプ」>「Eclipse マーケットプレース」を選択後に表示される Eclipse マーケットプレースのダイアログにて、検索の入力欄へ EGit と入力する。

さらに【GO】ボタンをクリックすると EGit が検索されるので、【インストール】ボタンを押下し、次に表示される【選択されたフィーチャーの確認】では、デフォルトの設定で【確認】ボタンを押下する。【図 11】

するとライセンスのレビューが表示されるので、使用条件の条項に同意のラジオボタンを選択後、「完了」ボタンを押下すると、EGit のインストールが開始される。インストール完了後、「今すぐ再起動」をクリックして Eclipse を再起動すれば、EGit が使用可能となる。【図 12】

EGit は、Git のパースペクティブを開くことで使用できる。Git パースペクティブを開くには、上部メニューの「ウインドウ」>「パースペクティブ」>「パースペクティブを開く」>「その他」で、「パースペクティブを開く」のウインドウを表示後、「Git」を選択して、開くボタンを押下する。すると Git のパースペクティブが開き、EGit が使用可能になる。【図 13】

次は、リモートリポジトリからローカルリポジトリを作成する。リモートリポジトリをコピーしてローカルリポジトリを作成することを、Git ではクローンと呼ぶ。以下にクローンを実行して、IBM i 上のリモートリポジトリからローカルリポジトリを作成する手順を説明する。

まず Git パースペクティブを選択して、「Git リポジトリ」から「Clone a Git repository」を選択し、「ソース Git リポジトリ」の「URI」に、リモートリポジトリの URL を設定する。弊社環境のリモートリポジトリは HTTP 経由でアクセスできるように設定されているため、【図 14】のように設定した。

次に、リモートリポジトリへアクセスするための認証情報（IBM i のユーザープロファイル、パスワード）を設定する。「セキュア・ストアに保管」をチェックすると、認証をたびたび要求されることがないので、チェックすることを推奨する。

リモートリポジトリの設定完了後、リモートリポジトリのブランチを選択する。ブランチはプロジェクトから分岐させることにより、プロジェクト本体に影響を与えずに開発できる機能である。つまり、作業履歴を枝分かれさせて記録していくために存在する。

*1 WYSIWYG は「What You See Is What You Get」の略で、編集画面での表示内容と同じものが完成形（HTML）として得られる機能を指す。

図5　Cobos4i IBMiプログラムの開発とコンパイル

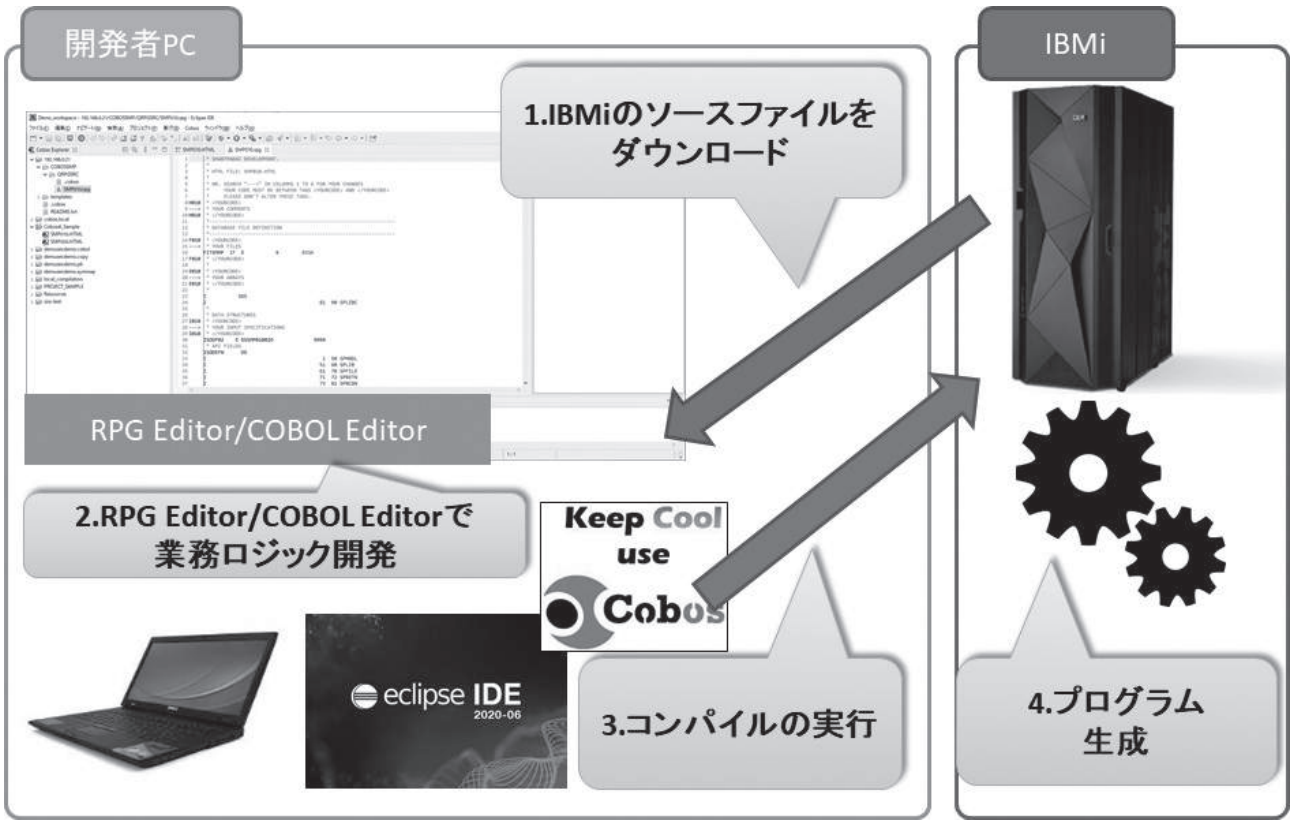
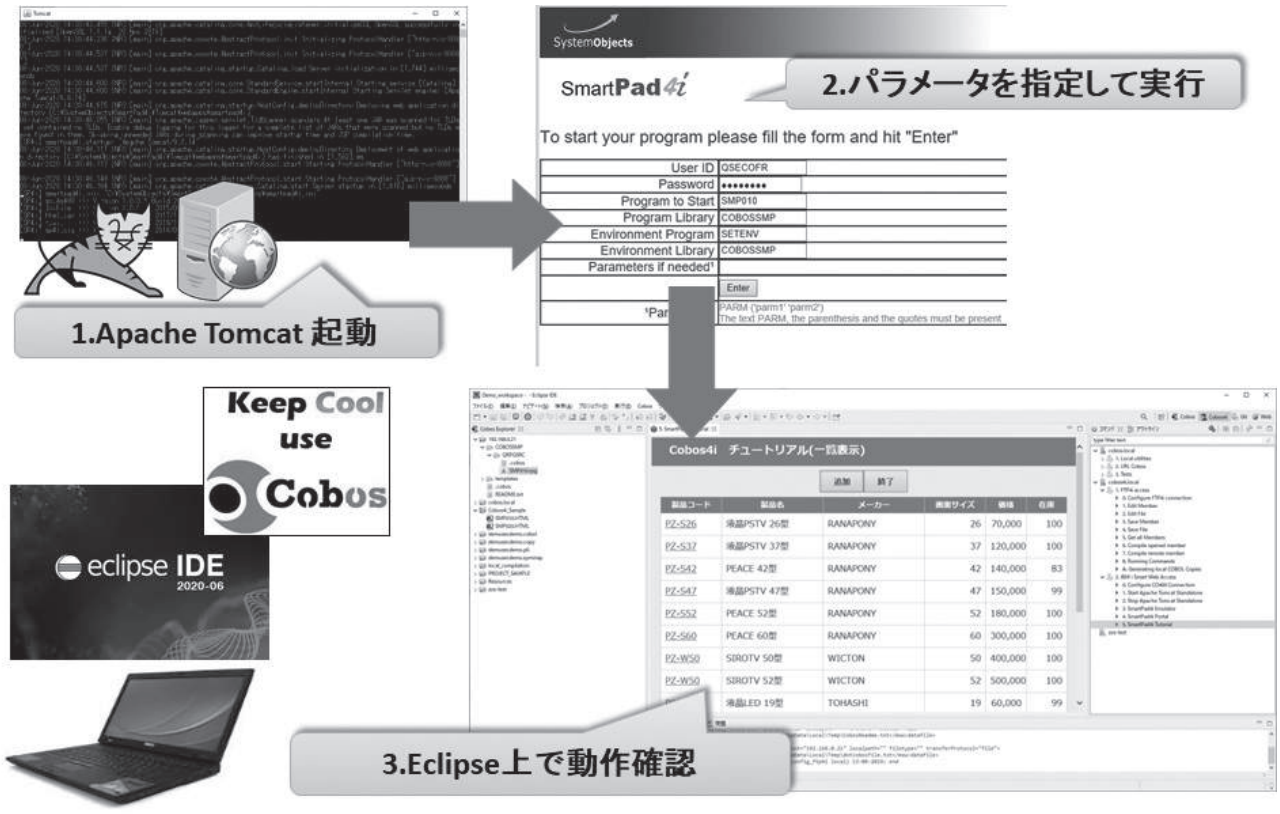


図6　Cobos4i アプリケーションの実行



ブランチが複数ある場合はクローンの際に、どのブランチをコピーするかを、【図 15】の画面で選択できる。コピーしたいブランチを選択して、「次へ」ボタンをクリックすると、ローカル保管場所を指定する画面が表示される。ローカルの作業ディレクトリを選択後、「完了」ボタンをクリックすると、ローカルリポジトリが作成される。

以上の操作により、リモートリポジトリがクローンされ、Eclipse プロジェクトとして追加された。

ソース管理下の RPG プログラムを更新する場合には、Cobos Explorer から対象の RPG プログラムソースを選択後、右クリック＞「チーム」＞「コミット」を選択する。【図 16】

すると Git ステージングのビューが開くので、ステージされた変更に RPG プログラムがリストされていることを確認後、「コミット」または「コミットおよびプッシュ」を行う。

コミットはローカルリポジトリのみを更新すること、プッシュはリモートリポジトリを更新することを意味する。【図 17】

また Git パースペクティブを開き、ヒストリーを確認することで、変更履歴を確認できる。さらに Git 参照ログでは、詳細なソースコードの変更点も確認できる。【図 18】

EGit を使用すると、ソースの変更履歴や変更箇所が管理できるため、開発時には非常に便利である。

5. おわりに

Cobos4i を使用することで、SmartPad4i アプリケーションを効率的に開発できることが理解いただけたと思う。新規に SmartPad4i アプリケーションを作成する場合は、Cobos4i での開発を推奨する。

オープンソースで広く使用される Eclipse のプラグインを、Cobos4i での開発に使用できるのは、大きなメリットである。Cobos4i によって、SmartPad4i アプリケーションの開発が劇的に変わると確信している。

M

図7　Webページ・エディター

図8　Webページ・エディター パースペクティブ

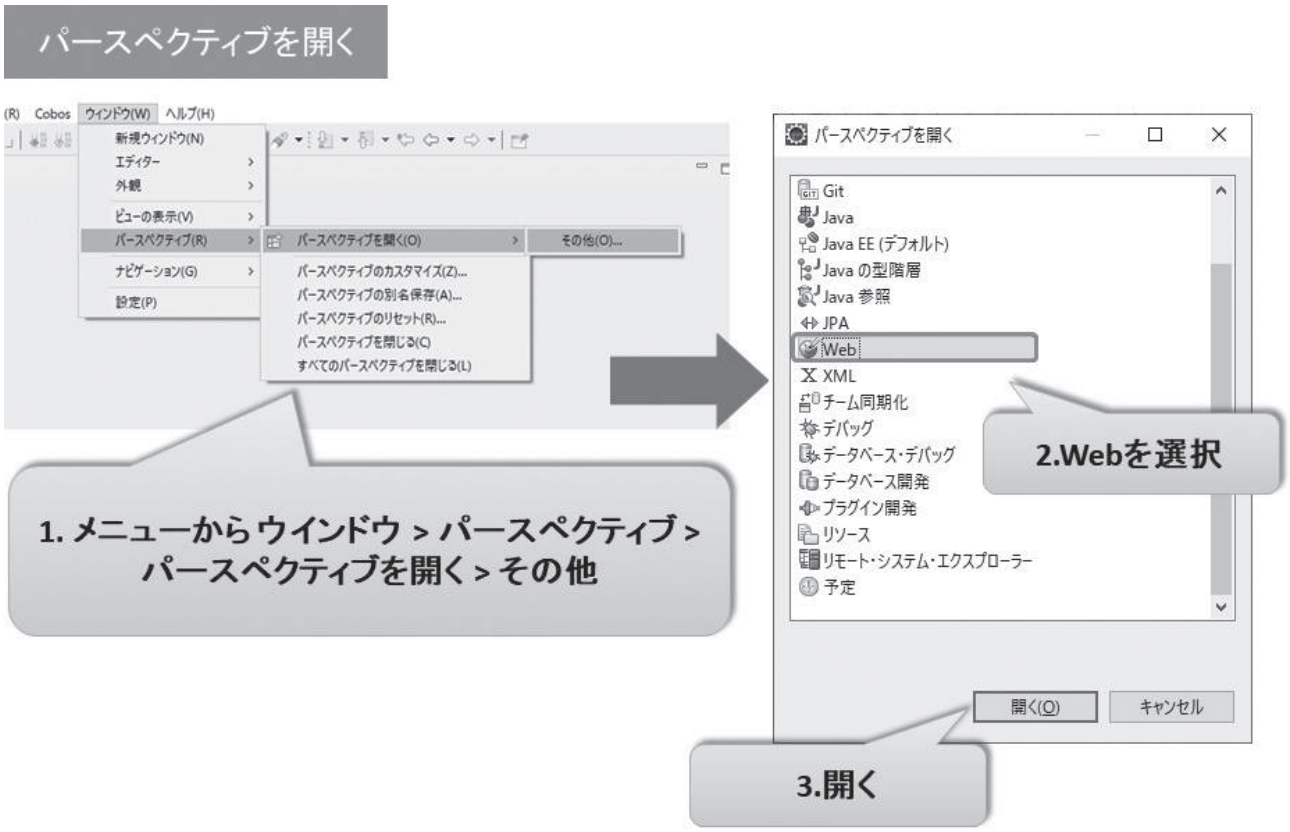


図9　Webページ・エディター ビューを開く

図10　IBMiにGitを構築

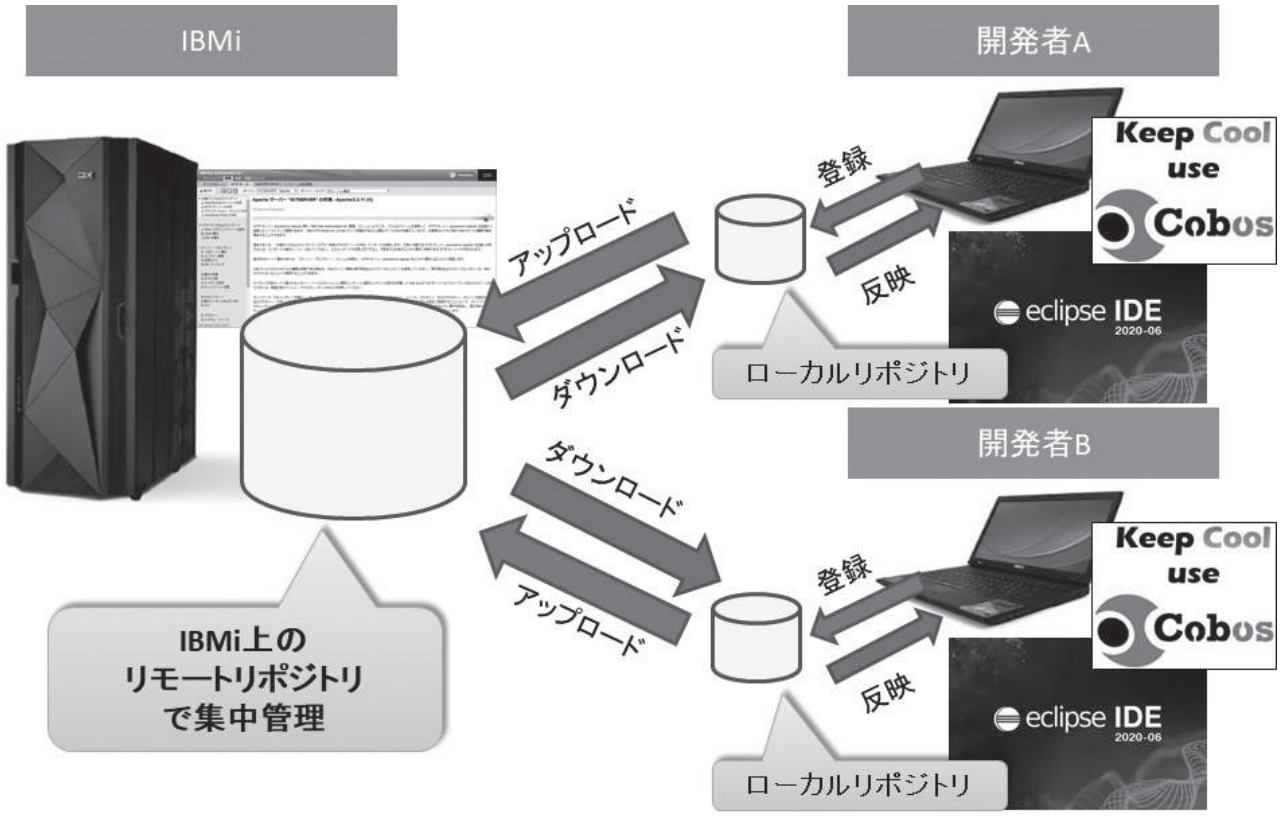


図11　EGitのインストール1

図12　EGitのインストール2

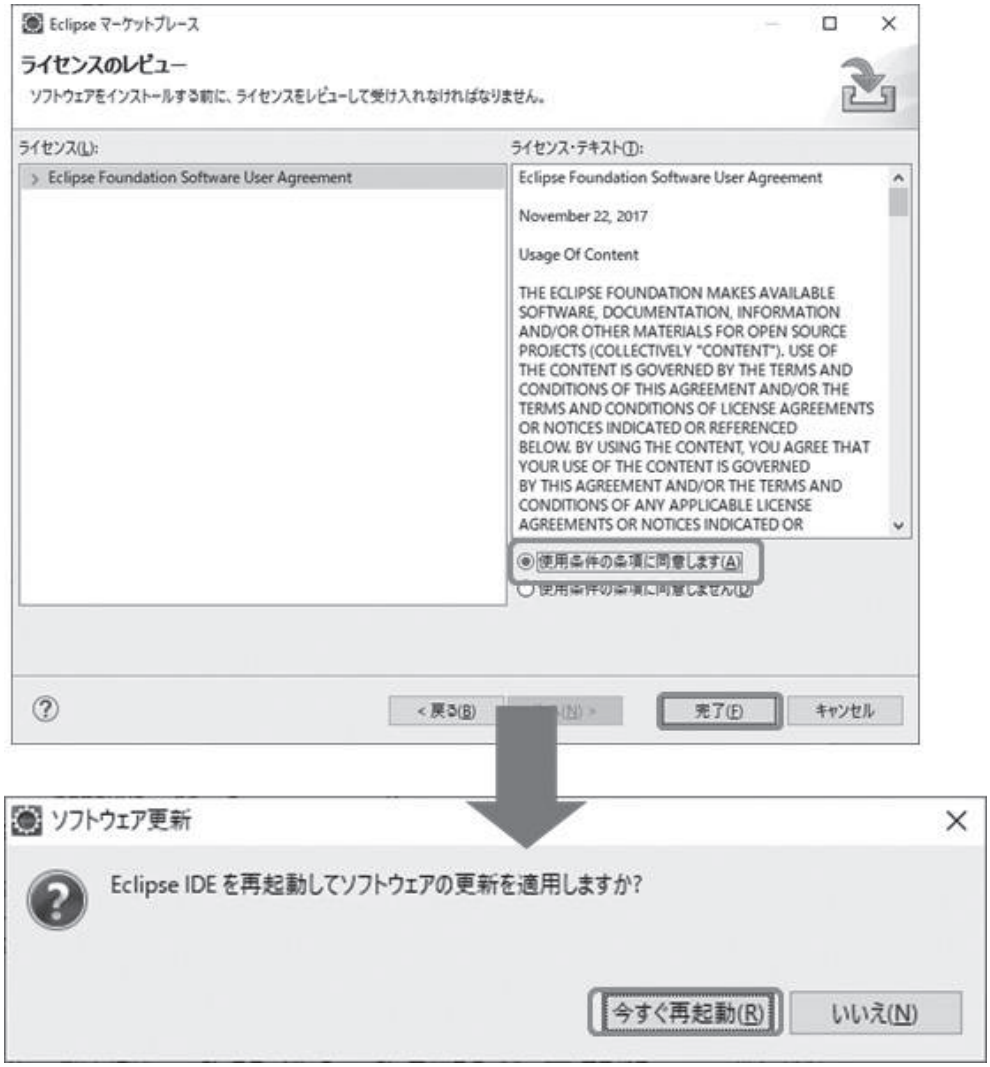


図13　EGitのパースペクティブを開く

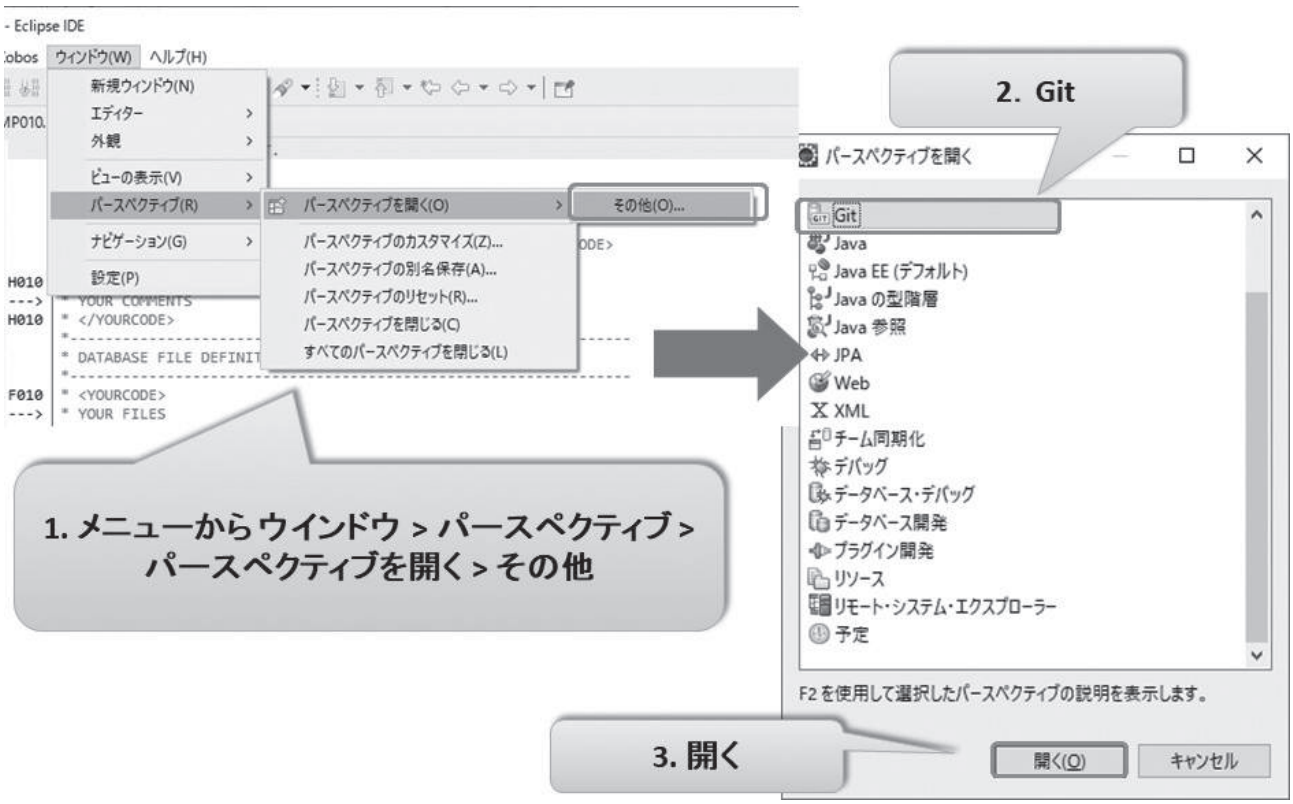


図14　Gitリポジトリのクローン1

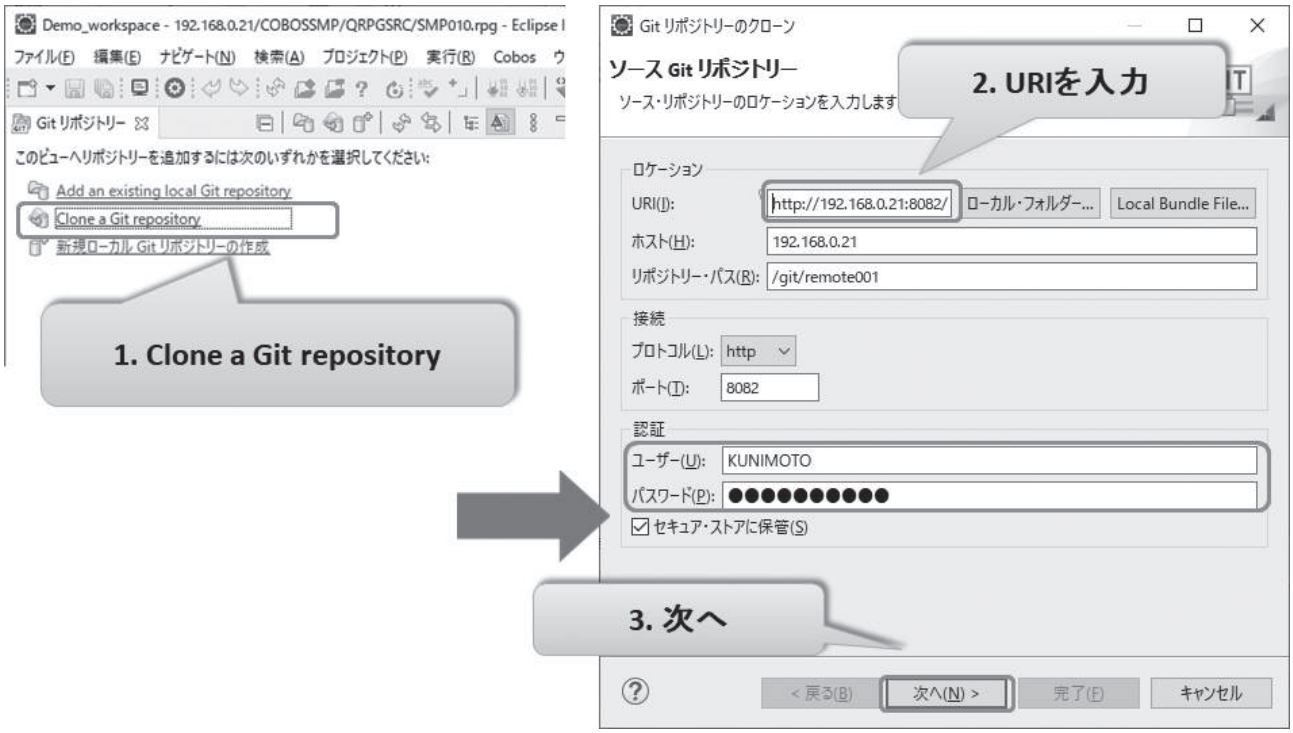


図15　Gitリポジトリのクローン2

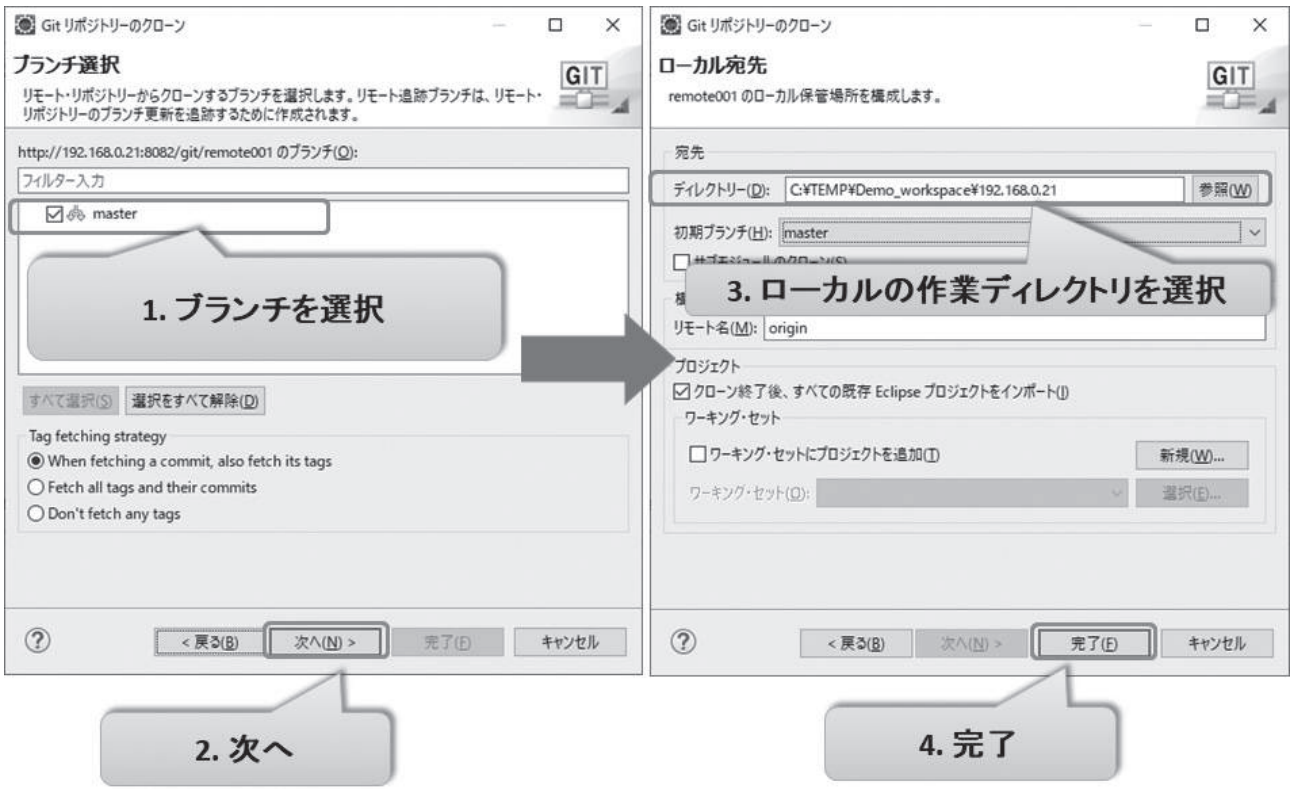


図16　Gitコミットとプッシュ1

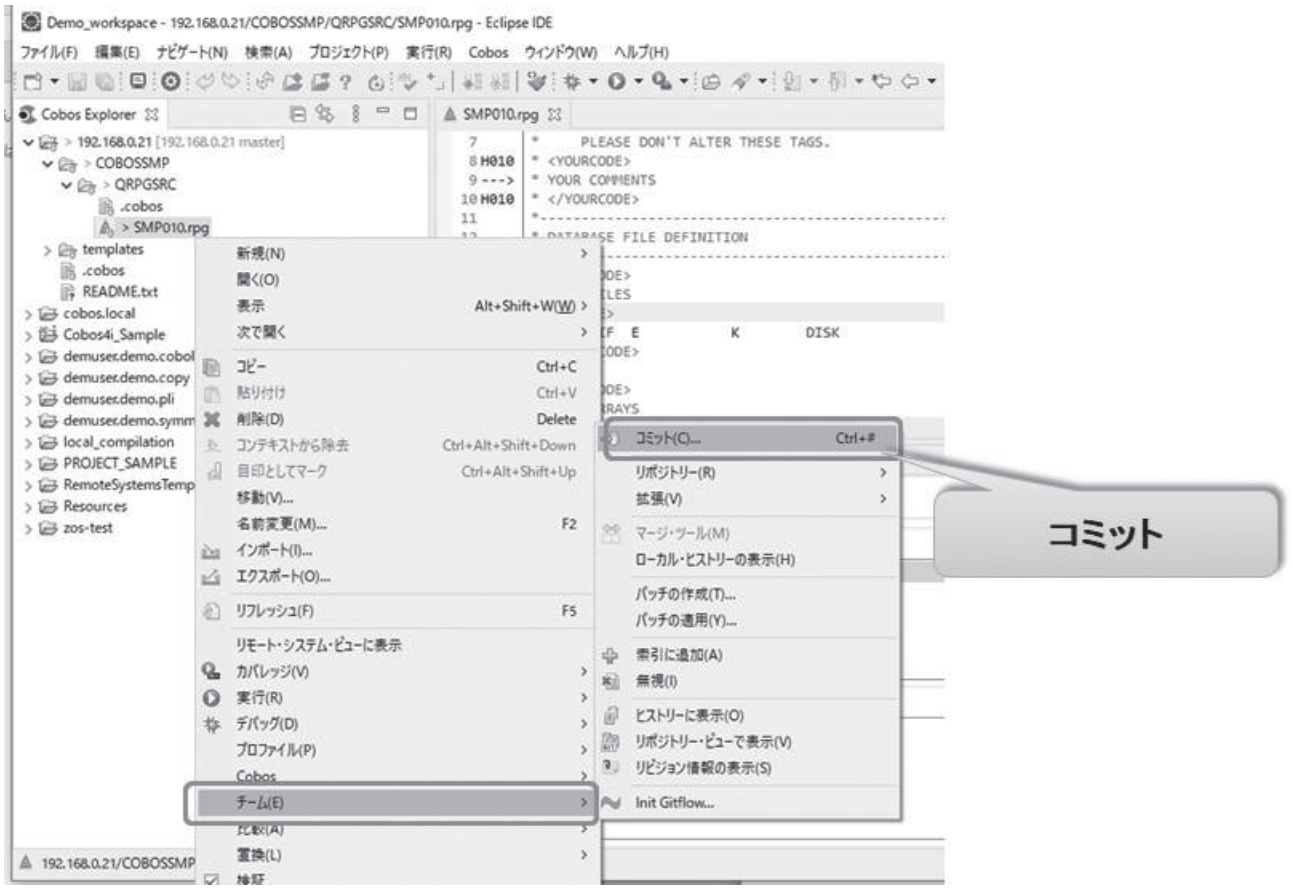


図17　Gitコミットとプッシュ2

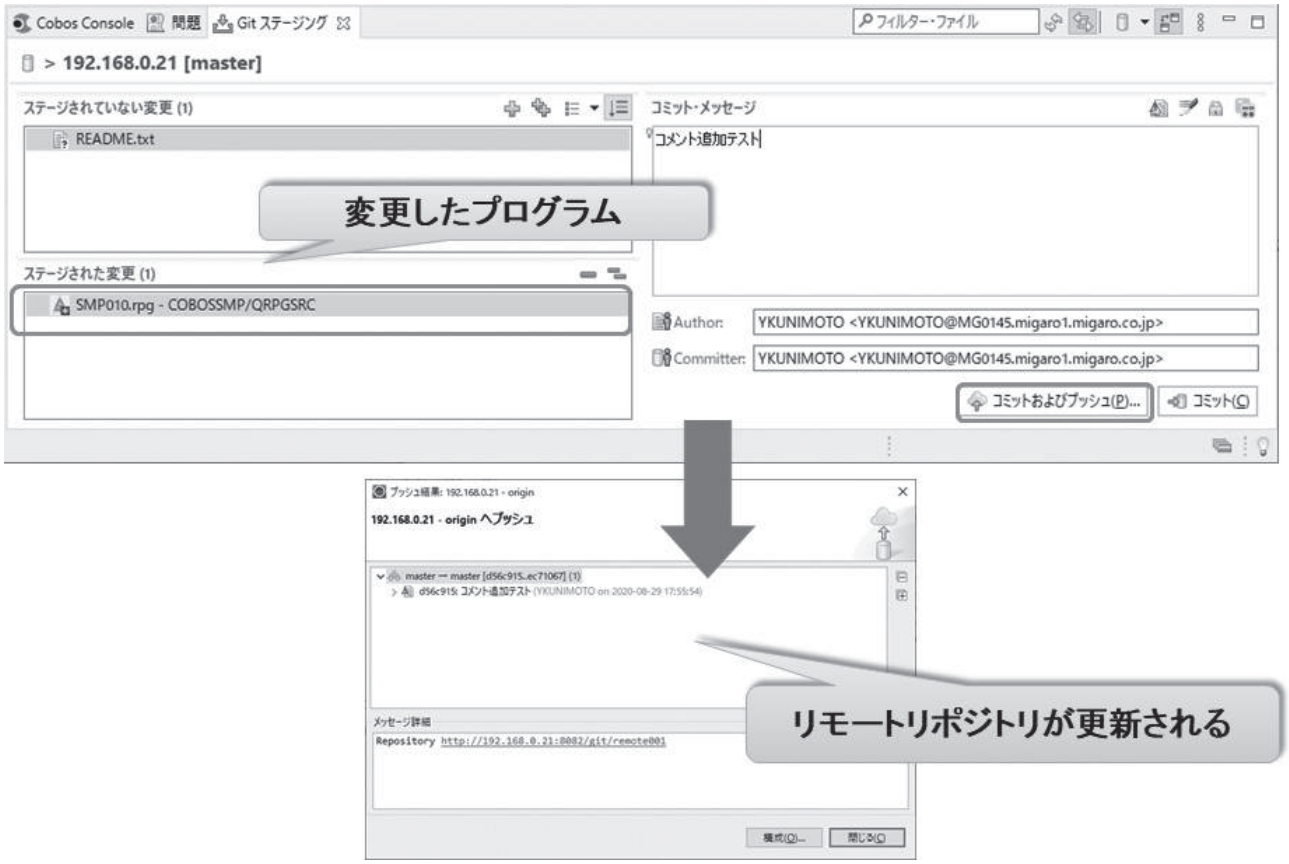
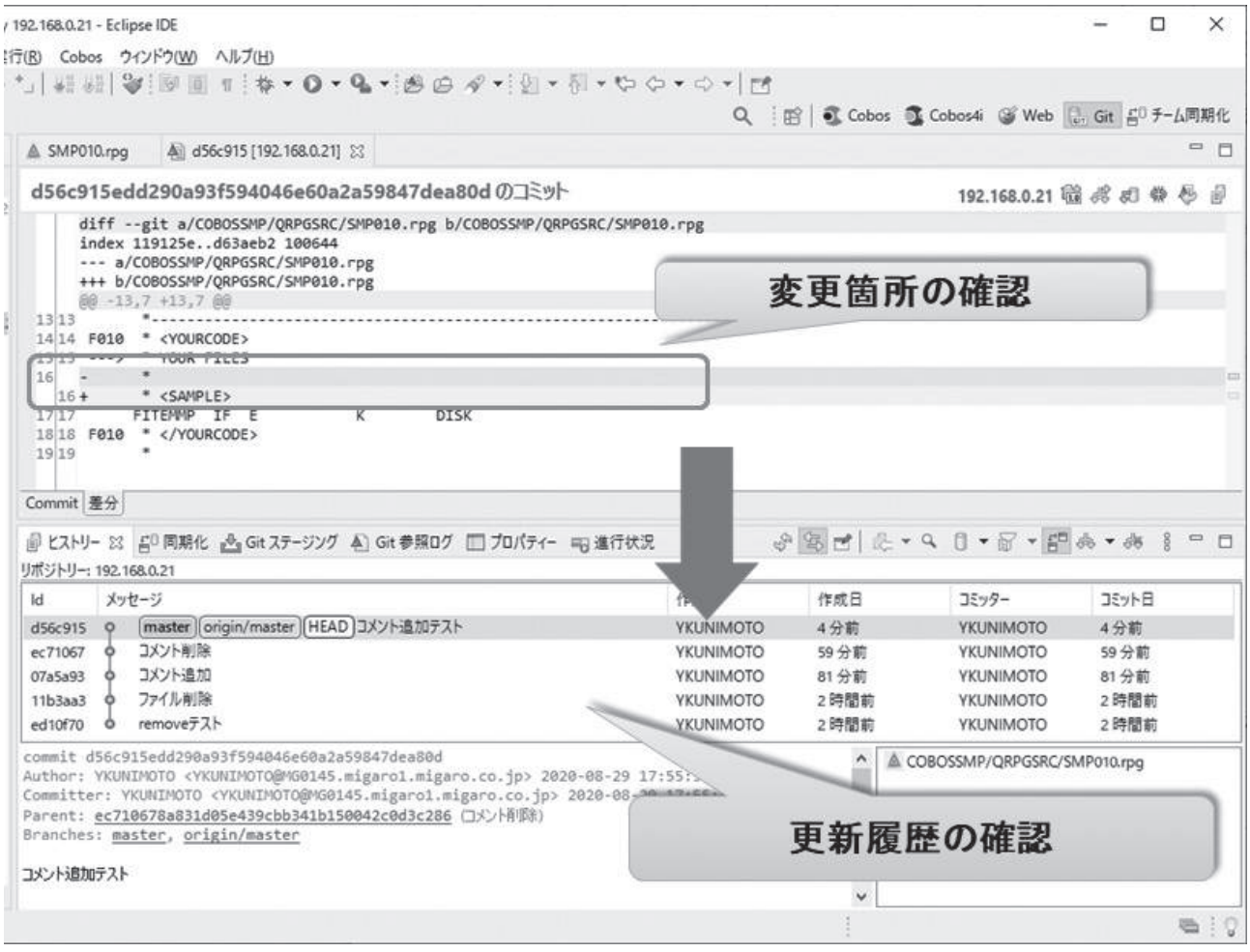


図18　Gitでソース管理



尾崎 浩司

株式会社ミガロ.

RAD事業部 技術支援課

[Valence]

Valence 最新バージョン進化のポイント

略歴
1973 年 8 月 16 日生まれ
1996 年 3 月 三重大学 工学部卒業
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
Delphi/400 を中心としたテクニカルサポート対応や製品セミナーの講師などを担当している。

1. はじめに

Valence は、IBM i を Web 環境で活用できるモダナイゼーション開発・運用ツールである。Valence での Web アプリ開発は、以前は Sencha と呼ばれる JavaScript フレームワークを使用する方法のみであった。

しかし 2018 年 8 月に登場した「Valence 5.2」にて、ローコード開発ツール「Valence App Builder」が追加されたことにより、アプリの開発効率が飛躍的に向上した。

ウィザードを使用したシンプルな 3 ステップの作成手法でアプリ開発ができるので、簡単なものであれば、ものの数分でアプリを作成可能である。

App Builder は基本的にノンコーディングで簡単にアプリを作成できるが、複雑な業務ロジックを組み込みたい場合は、RPG を組み合わせることもできる。

App Builder の概要および RPG 連携テクニックについては、2019 年発行のテクニカルレポートにある『Valence App Builder RPG 連携テクニック』で詳しく紹介しているので、参考にしていただきたい。

この Valence だが、2020 年 6 月にバージョンアップし、Valence5.2+（プラス）となった。さらに 2020 年 9 月の本稿執筆時点では、正式発表前ではあるが、まもなく Valence6.0 のリリースも予定している。

本稿では、Valenec5.2+ 以降の最新バージョンにおける製品進化ポイントならびに開発ツール App Builder の強化されたアプリ開発テクニックを紹介する。

2. Valence最新バージョンの進化点

Valence 最新バージョンの主な進化点は、以下のとおりである。

(1) Valence Portal の機能強化
(2) Web エミュレータ Fusion5250 の追加
(3) App Builder アプリ開発機能の強化

(1) については、Valence Portal のユーザー認証機能や権限制御が強化された。具体的には、「2 ファクタ認証」によるセキュリティ強化と、「NAB 権限」による App Builder の作成権限制御の強化である。

2 ファクタ認証とは従来のパスワードによる認証に加え、利用者個人が所持するスマートフォン等の認証アプリを使用したデジタル認証コードを組み合わせることで、セキュリティを向上させる仕組みである。Google や Amazon 等でも使用されているセキュリティ強化の一般的なアプローチと同じである。【図 1】

2 ファクタ認証を有効にすれば、たとえ Valence にサインオンするユーザー ID とパスワードの情報が外部に漏洩したとしても、ユーザー本人所持のスマートフォン上で、都度発行される確認コードを正しく入力しない限り、Valence にサインオンできなくなる。

Valence で機密性の高いデータを扱う

図1　Valence 2ファクタ認証

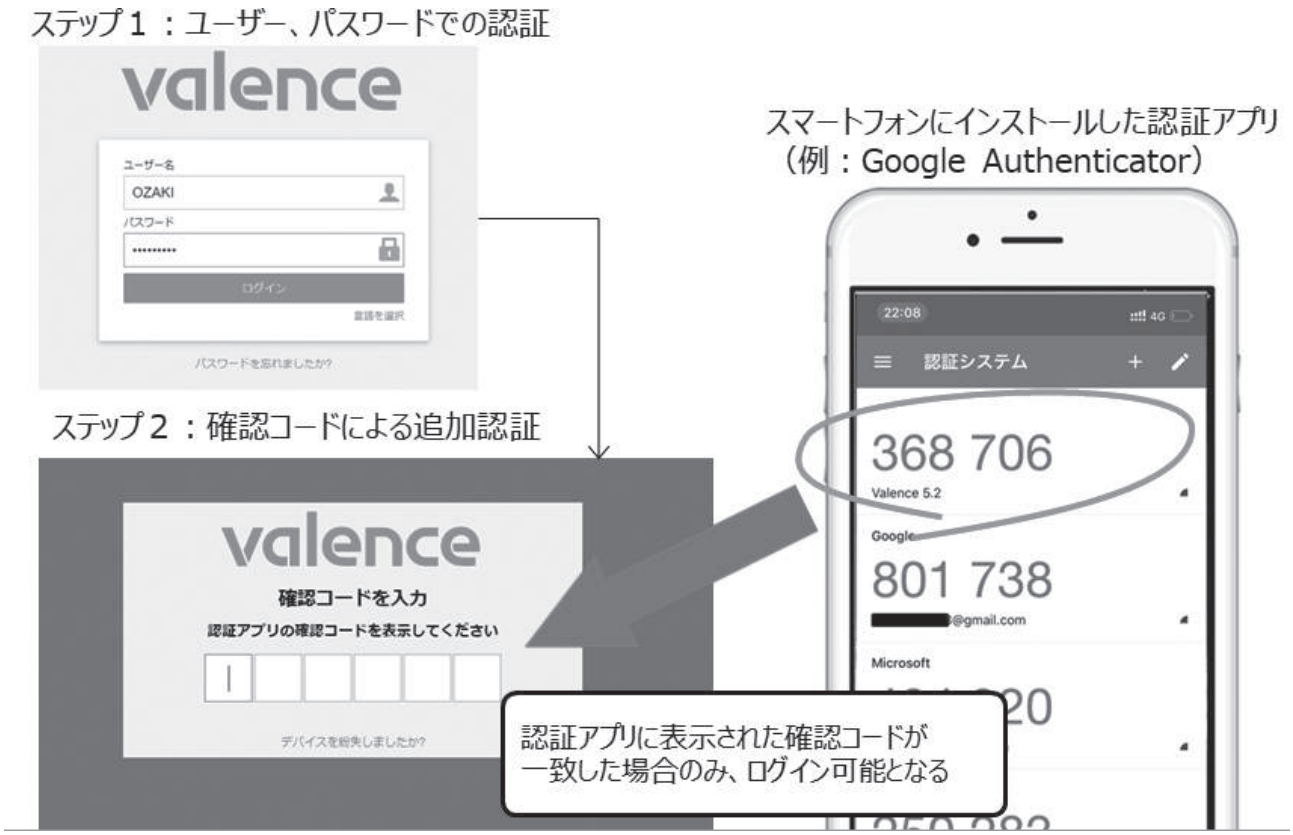


図2　Valence NAB権限

必要がある場合、この2ファクタ認証を有効にしておくと効果的である。

次にNAB権限だが、これはValenceのユーザー登録に追加された機能である。

これまではValenceのグループ設定により、Valence PortalからApp Builderメニューを表示する・しないを制御することで、アプリ開発が可能なユーザーを制限することは可能であった。

しかし今回、NAB権限が追加されたことで、ユーザーごとにApp Builderの作成権限レベルを制御できるようになった。

たとえば、管理者があらかじめ用意したデータソースを一般ユーザーに提供し、一般ユーザーにはウィジェットやアプリケーションの作成だけを許可するといった使い方が可能である。あるいはウィジェットの作成権限はユーザーに付与するが、データを更新するためのウィジェット（Edit Grid）は不許可にすることで、データの参照目的のみでApp Builderの作成を一般ユーザーに開放するといった使い方もできる。【図2】

(2) のWebエミュレータFusion5250は、Valence5.2+で追加された新機能である。Valence Portalのメニューから Fusion5250アイコンをクリックすると、ブラウザ上で5250エミュレータを実行できる。

これは従来のクライアントアクセスのような専用ソフトに依存せず、ブラウザだけで実行できるので、Windowsクライアントだけでなく、たとえばMacクライアントでもエミュレータが利用できる。【図3】

Fusion5250は単にブラウザで動作するエミュレータというだけでなく、Fusion5250という名前が示すとおり、他のValence機能との融合がポイントである（詳細は3.で紹介する）。

(3) は、ローコード開発ツールApp Builderの強化である。Valence最新バージョンでは、新たに2つのウィジェットが追加された。「タイル」ウィジェットと「タイムライン」ウィジェットである。

タイルウィジェットは、データソース上のデータをタイル形式のパネルに表示する部品である。会社の受付システム等で見られるタッチパネルを使って、社員を選択するような画面が作成できる。【図4】

一方のタイムラインウィジェットは、データソースのデータを時系列に並べて表示する部品である。たとえば、売上日ごとにどんな商品がよく売れるのか、季節ごとにどんな商品が売れ筋なのか、など横軸の時系列に対してデータをプロットするようなインタラクティブなタイムラインを表示できる。【図5】

また最新バージョンでは、新たにアプリケーション内で扱えるアプリ変数機能が追加された。アプリケーション内の複数ウィジェット間で変数値を共有でき、さらにRPGで取得した値をクライアント側で保持することも可能になった。【図6】

ほかにもセキュリティ機能の強化や、スクリプト実行機能、RPGの新しいテンプレートなど多彩な機能強化がなされている（詳細は4.で紹介する）。

3. Fusion5250を使用したアプリの融合

PC5250エミュレータを日本語環境で使用する場合、ホストコードページに［930: 日本語（カタカタ）］、あるいは［939: 日本語（拡張ローマ字）］を使用するのが一般的である。ValenceのWebエミュレータであるFusion5250でも、同様にコードページが指定できる。【図7】

Valence Portalより［ポータル管理］を起動し、サブメニューより［言語］を開くと、言語ごとにFusion5250で使用するコードページ設定が表示されるので、［930-Japan Katakana Extended］、あるいは［939-Japan Latin Extended］を選択すればよい。

Fusion5250は、デフォルトでは一般的なエミュレータと同様に、黒背景・緑文字が基調のいわゆるグリーンスクリーンで表示される。IBM iユーザーにとってはお馴染みだが、Valenceのほかの画面と比べると異質に感じるかもしれない。

Fusion5250には、テーマ設定が用意されている。3種類から選択できるが、このうちの［Valenceテーマ］は他のValenceアプリとの親和性がよいので、こちらを選択するとよい。【図8】

Fusion5250の特徴の1つがマクロ機能である。これは、エミュレータ上でのユーザー操作をスクリプトとして定義しておき、コマンド操作を自動化する仕組みである。

マクロはJSON形式で記述し、Valenceインスタンス内のIFS上（/resources/fusionmacros/）に保管すればよい。ここではマクロの例として、IBM i上のCLプログラムをCALLし、特定のアプリを起動する例を紹介する。ここで使用するマクロファイルは、【ソース1】である。

マクロファイルは、「note」「showProgress」「steps」という3つの要素で構成される。noteは単なるメモで、動作に影響は与えない。showProgressは、マクロ実行中の途中画面を表示する、表示しないを切り替える。falseを指定すると、すべてのマクロが実行されたあとに画面が表示される。stepsは、実行する個々の処理ステップである。

【ソース1】のマクロは、エミュレータの7桁目、20行目、つまりエミュレータのコマンド入力ラインにカーソルをセットし、CLコマンドをタイプしたあと、［実行］（ENTER）キーを押下するという動作を表している。

Fusion5250からマクロを実行する方法は、Valence Portalの［ポータル管理］→［アプリ］より、新規アプリを追加し、パス欄に以下の指定を行えばよい。【図9】

/build/production/Fusion/index.html？macro＝［マクロファイル名］（※拡張子.jsonは記述不要）

以上で完了である。これでValence Portal上に、新しいメニューが追加される。このメニューをクリックすると、Fusion5250が立ち上がると同時にマクロが実行され、既存のPC5250アプリがブラウザ上に立ち上がる。Valence PortalとPC5250アプリがシームレスに融合していることがわかる。【図10】

なお、今回はマクロをJSONファイルで直接作成する方法を紹介したが、マクロの記録機能を使用すると、Fusion5250上の操作を自動記録し、そのままValence Portalに新たなアプリとして追加できる。Fusion5250画面左下にある［REC］ボタンをクリックするだけで記録できるので、試してほしい。

図3　Fusion5250

図4　タイルウィジェット

またマクロを使用したFusion5250は、Valence Portalからだけでなく、App Builderで作成するアプリからも呼び出せる。アプリケーション作成時に、「動作内容」画面で定義するアクションとして、Fusion5250を指定する。もちろんマクロを含められるのだが、その際にパラメータとして、ウィジェット上のフィールド値やアプリ変数の値を渡すこともできる。【図11】

つまりウィジェット上で選択したレコードのキーをパラメータにして、マクロを実行することにより、PC5250アプリ上のキー項目に値を自動入力したり、RPG側にパラメータとして値を渡したりできる。

この仕組みを使用すれば、App BuilderとFusion5250とのシームレスなアプリ融合が可能になる。【図12】

4. App Builderの強化されたアプリ開発機能

App BuilderとRPGとの連携例として、商品マスタメンテナンスの登録画面の作成方法を2019年発行のテクニカルレポートで紹介した。【図13】

ここでは、ボタンをクリックしたときに実行されるRPGプログラムを紹介した。ボタンクリックは、VALENCE52/QRPGLESRC（EXNABBTN）のテンプレートを使用する。【図13】のサンプルで作成したプログラムが、【ソース2】である。

このEXNABBTNプログラムは、Valence5.2+以降で記述方法が一部変更された。ポイントは、【ソース2】の1-①～③の部分である。

以前は、1-①のようにd仕様書としてプロシージャー宣言が必要だったが、これは不要になった。以前のバージョンで作成されたソースを再コンパイルする際には、この部分を削除する必要があるので注意してほしい。

1-②、1-③の部分は、レスポンスをクライアント側に返却する処理であるが、以前は、vvOut_toJsonPairというValenceの汎用APIを直接記述する必要があった。最新バージョンでは、EXNABBTN用の専用APIが追加されており、その中には、レスポンスを返却するSetResponseというAPIが追加されている。【図14】

これにより【ソース3】のように、クライアントへのレスポンスの返却がよりシンプルに記述できるようになった（ただし、従来の記述方法でも正しく動作する）。

さらに専用APIになったことで、レスポンス結果として、クライアント上の特定機能を使用不可に変更するような制御が可能になった。

次に、ファイルのアップロードについてである。旧バージョンのValence5.2でもRPG Toolkitを活用することで、CSVファイルやExcelファイルのダウンロードを実装できたが、これまでファイルのアップロードはサポートされていなかった。

今回の最新バージョンでは、アップロードできるようになり、画像データの登録やデータの一括登録などが可能になった。今回紹介する例は、商品マスタメンテナンスにおけるCSVデータ取り込み処理の追加である。【図15】

この例では、ファイルのアップロードはApp Builderにて動作設定を定義し、RPGプログラムを組み合わせて作成する。App Builderでは「RPGプログラム呼び出し」を追加し、設定画面を開く。

そしてプログラム実行前処理として、「Prompt For User Information」を選択し、パラメータを追加する。そしてパラメータ名にfile、タイプにUploadを選択すればよい。もしアップロードするファイルの拡張子を制限する場合は、さらにValid Extensionsに許可する拡張子を指定する。【図16】

そしてRPG側では、Valenceに用意されたRPG Toolkitの中にあるvvInを使用する。【ソース4】

アップロード先は、vvIn.pathにIFSディレクトリを指定する。任意のディレクトリを指定可能だが、今回は、2-①のようにValenceが保持しているテンポラリーディレクトリ（TEMP_PATH）を指定している。2-②のように、vvIn_fileを使用するとファイルがアップロードされる。なお第2引数は、常に“*NULL”を渡す。

これでIFS上にファイルがアップロードされるわけだが、今回はこのCSVファイルを使用して、物理ファイルである商品マスタ（MSYHIN）の内容を置き換える。IBM iには、CPYFRMIMPFというコマンドがあるので、今回はこのコマンドをRPGから実行することとした。

2-③のように“QCMDEXEC”を使用すると、RPGからCLコマンドが実行できる。

以上でCSVアップロード処理は完成である。CSVファイルを使用したデータの一括登録は利用頻度が高いと思うので、今回の内容をもとに活用してほしい。

次に、Valenceの新しいセキュリティ機能を紹介する。これを使用すれば、作成したアプリの中の一部機能を特定ユーザーのみ実行可能にできる。たとえば、先ほどのCSVデータ取り込み処理を、特定のグループ（今回は「経理グループ」（ID：1010）を例とする）の場合のみ使用できるように権限設定する。【図17】

セキュリティ機能も、App Builderの設定とRPGとの組み合わせで実装する。

まずApp Builderだが、アプリケーション作成画面に新たに追加された「セキュリティ」ボタンをクリックする。セキュリティ設定一覧が表示され、制御したい機能に対し任意の名前（機能名）を付与し、あとは実行したいRPGのプログラムIDを指定すればよい。【図18】

【図18】で指定したプログラムは、VALENCE52/QRPGLESRC（EXNABSTART）というテンプレートを使用してRPGロジックを作成する。App Builderで設定した機能名を使用不可にする場合は、disableFeatureを呼び出せばよい。【ソース5】

このサンプルでは、3-①のようにRPG Toolkitに用意されたvvSecurity APIを使用して、実行中のユーザーが指定したグループに含まれるかを判断し、含まれない場合にdisableFeatureを実行している。

なお今回のサンプルでは使用していないが、現在実行中のユーザーはこのテンプレートプログラム内で、gUserという変数に格納されているので、これを使用することも可能である。

次に、入力画面等で使用するFormウィジェットの新機能を紹介する。ヘル

図5　タイムラインウィジェット

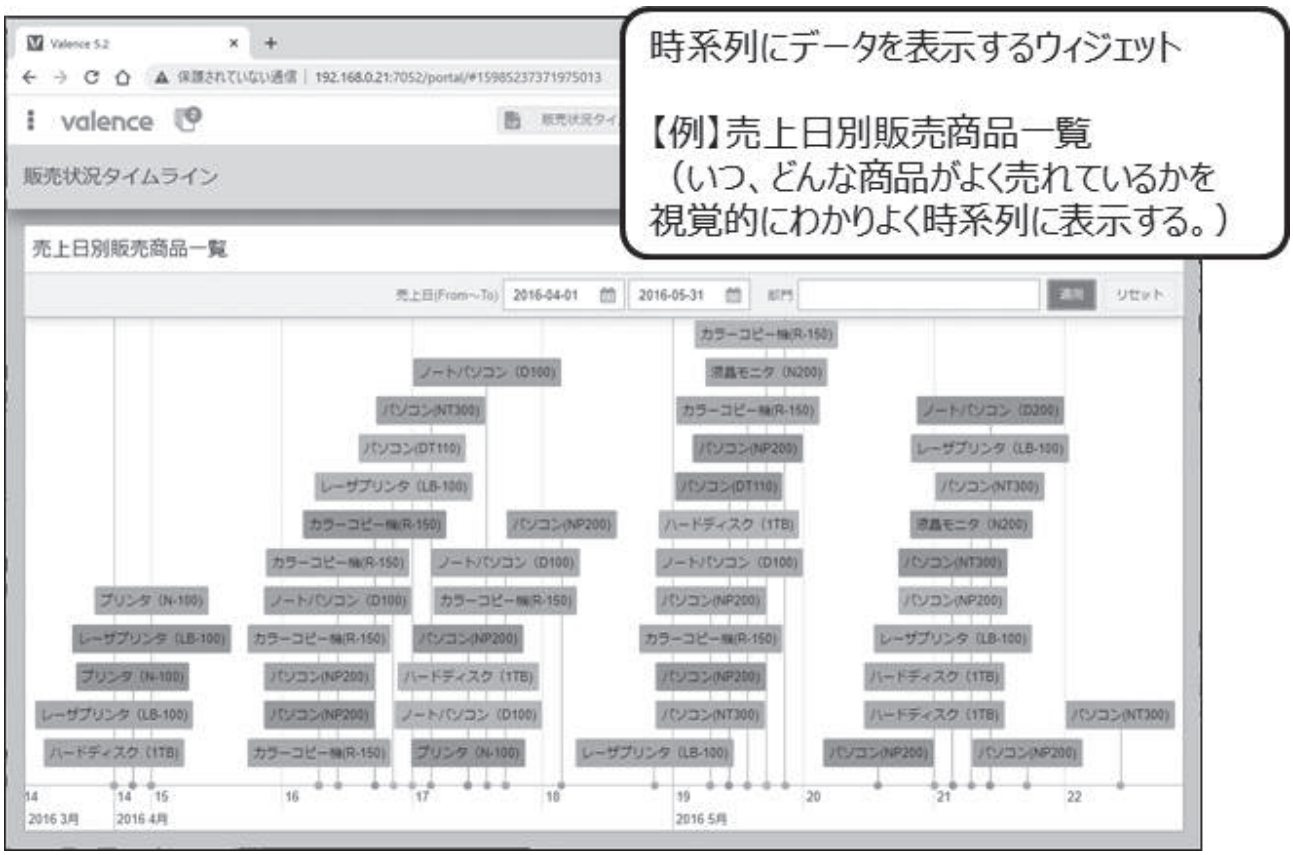


図6　アプリ変数

アプリケーション作成画面

パープログラムと呼ばれる機能である。Valence5.2では、ボタン等をFormウィジェットに配置し、そのボタンをクリックすることでRPGプログラムを実行する処理だけがテンプレートで用意されていた。しかし最新バージョンで追加されたヘルパープログラムを使用すると、リアルタイムな応答性を備えるインタラクティブな画面が作成できる。

今回紹介する例は、入庫入力を行う画面である。アプリ起動時に入庫日欄には、システム日付が初期セットされる。商品コードの値を変更したタイミングで、自動的に商品コードの存在チェックを行い、商品マスタに当該レコードが存在する場合は商品名を表示し、存在しない場合はエラーを表示する。【図19】

ヘルパープログラムも、App BuilderとRPGの組み合わせで実装する。Formウィジェットの設定画面に追加されたヘルパープログラム欄に、実行したいRPGプログラムのIDを指定する。【図20】

あわせてヘルパープログラムが呼び出させるタイミングを指定する。設定はこれだけである。実際の処理はすべてRPGで記述する。ヘルパープログラムは、VALENCE52/QRPGLESRC（EXNABFHLP）というテンプレートをもとに作成する。今回作成したTEC20PG30のソースは、【ソース6】である。

このプログラムには、4-①、4-②の2つの処理がある。4-①は、初期表示時の処理であり、実行モードを示すgMode変数が“formRender”のとき（4-③）に、起動時を表している。今回は、4-④のようにSetValueを使用することで、フォーム上に初期値をセットしている。

gModeがブランクの場合はフィールド変更時（4-②）を表しており、4-⑤のようにgField変数に対象のフィールド名がセットされる。今回は、4-⑥のようにvvIn_Charを使用して、フォーム上の値を取得し、商品マスタの存在チェックを行い、存在する場合は取得した商品名をフォームにセットする。存在しない場合は、4-⑦のようにSetErrorを使用して、エラーメッセージをフォームにセットしている。

またフォーカスを商品CD欄に再セットしている。このようにヘルパープログラムの追加により、Formウィジェット上の画面制御が大幅に向上したことがわかる。

最後に、App Builderにおけるスクリプト機能を紹介する。最新バージョンでは、RPGによるロジック追加に加え、クライアント側のロジック追加としてスクリプトが記述できるようになった。

これにより、RPGに依存しないような画面上の制御も簡単に実装できる。スクリプトを追加する際は、動作内容の中からスクリプトアクションを追加すればよい。

【図21】の枠内にスクリプトを記述すればよいのだが、スクリプトの実装例として、先ほどの入庫入力画面にQRコード読み込み機能を追加する。Valenceは、PCを使用したブラウザでの実行に加え、専用アプリを使用したモバイルアプリとしての実行も可能である。モバイルアプリの特徴は、デバイスの機能を活用できることである。今回は、デバイスのカメラ機能を活用したQRコード／バーコード読み込みの追加となる。実装例は【ソース7】となる。

このスクリプトの追加により、モバイルアプリで実行したときだけ、カメラが起動するようになり、QRコードやバーコードが読み取れる。【図22】

なお、PCブラウザからボタンをクリックして実行しようとした場合は、エラーメッセージが表示されるようになっている。

スクリプトソースのうち、取得したコードの値をForm上のどのフィールドにセットするかの部分（下線部）だけを変更すれば、他のプログラムにもそのまま組み込めるのでぜひ活用してほしい。

5. さいごに

本稿では、バージョンアップした最新版Valenceの進化点を紹介してきた。

これまでは、既存のPC5250アプリを専用エミュレータ上で実行し、App Builderで作成したWebアプリをブラウザ上で実行するといったように、それぞれの仕組みを個別に使用するしかなかった。

今回のバージョンアップにより、これら異なる種類のアプリが融合され、1つのブラウザ上でシームレスに使用できるようになったので、ユーザーの利便性が大幅に向上した。

またApp Builderについても、従来からのシンプルな開発手法という特徴はそのままに、大幅に機能拡張されたので、より本格的なアプリケーションの作成にも対応できる。ぜひ本稿を参考に、さらなるValenceの活用をご検討いただきたい。

M

図7　Fusion5250 ホストコードページ

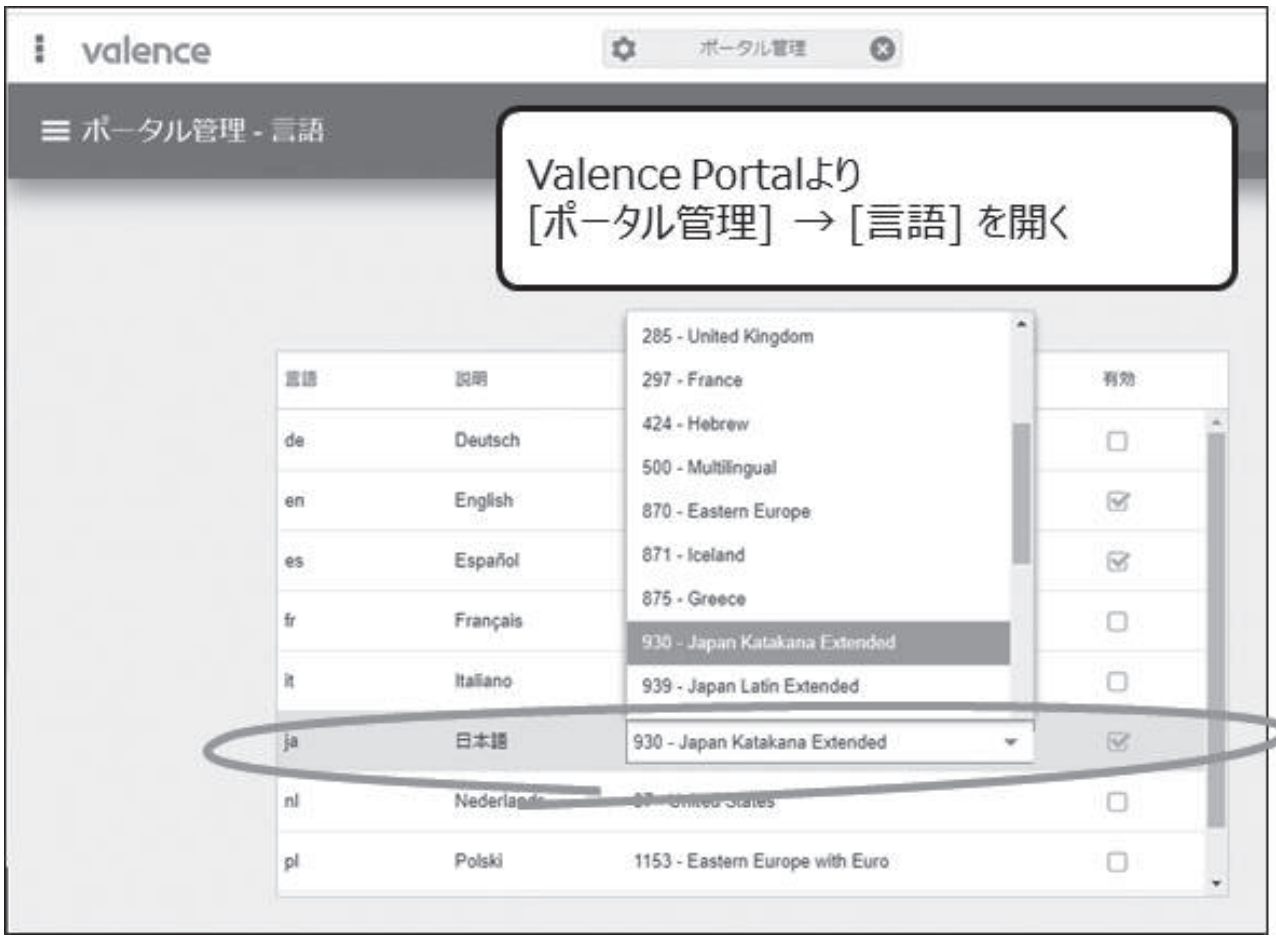


図8　Fusion5250 テーマの設定

ソース1　Fusion5250 マクロ 記述例

{} 5250SampleApp.json ●

V: > resources > fusionmacros > {} 5250SampleApp.json > ...

```
{
    "note": "iSite給与メニュー起動",
    "showProgress":false,
    "steps": [
        {
            "col":"7",
            "row":"20",
            "type":"CALL PXOLIBSO/START"
        },
        {
            "action":"ENTER",
            "control":false,
            "shift":false
        }
    ]
}
```

図9　Fusion5250 マクロの呼び出し

図10　Fusion5250 マクロ実行

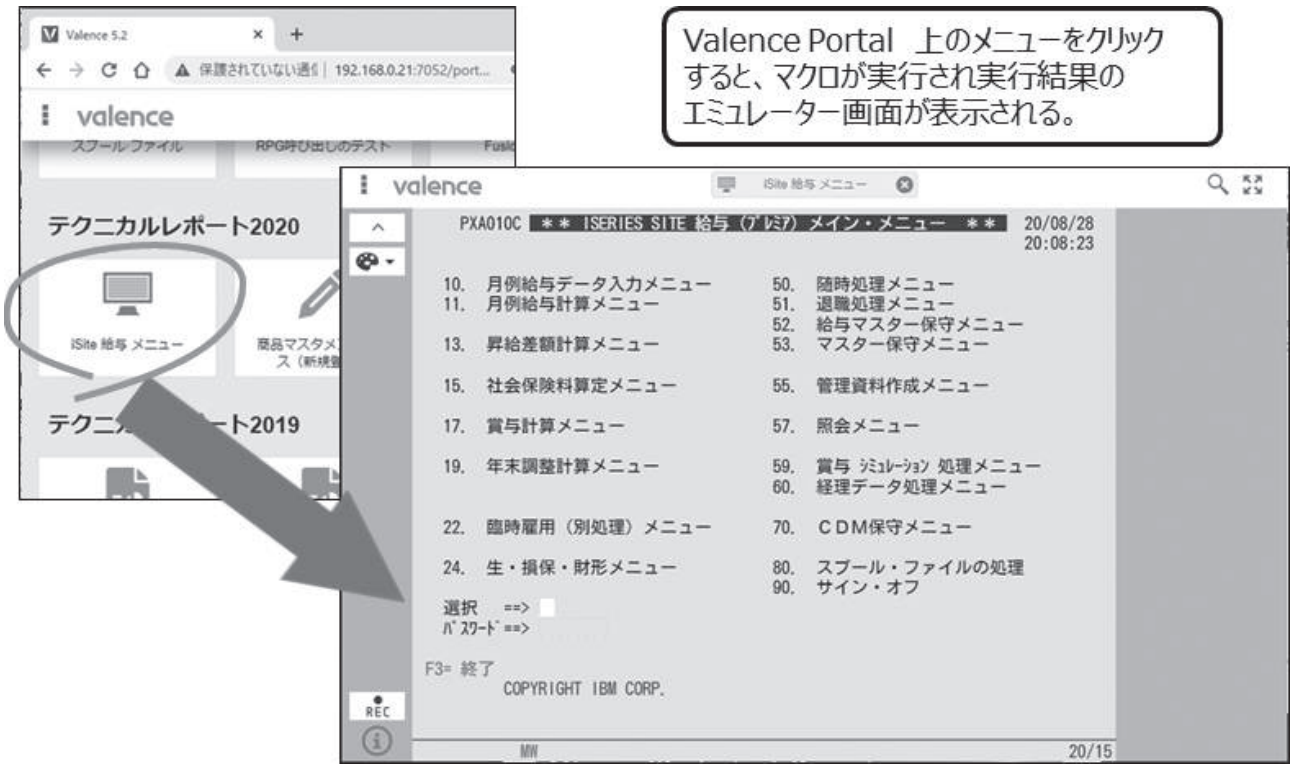


図11　Fusion5250 マクロパラメータ受け渡し

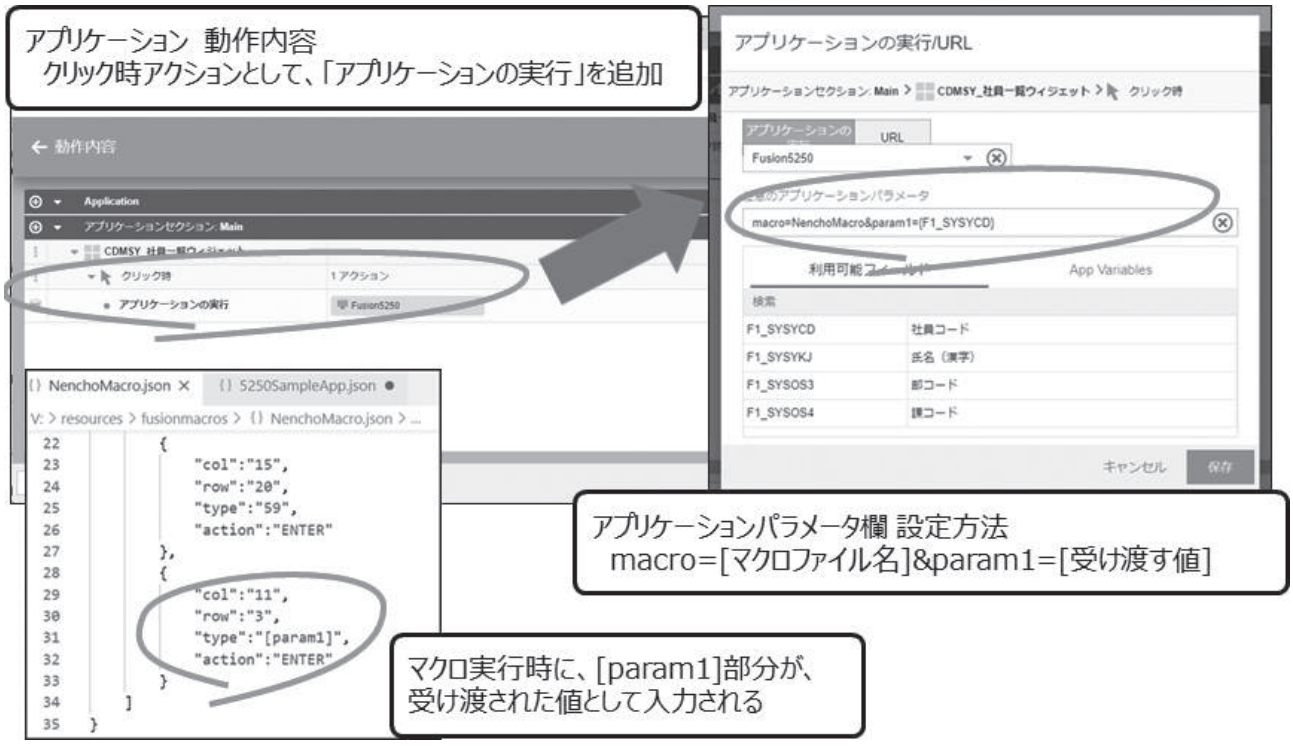
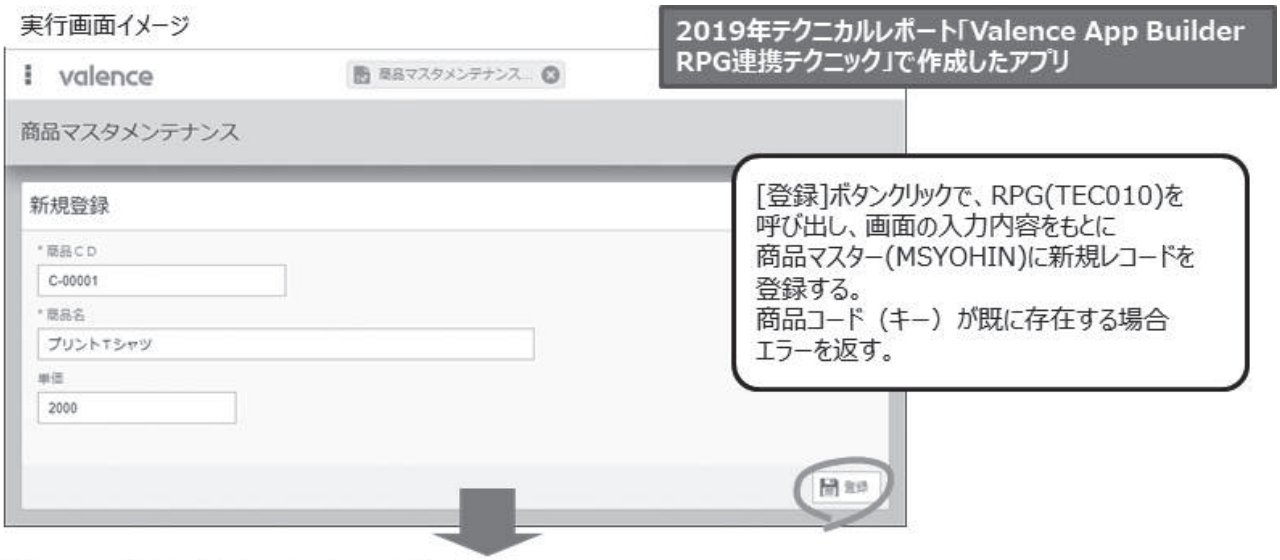


図12　App BuilderとFusion5250の融合

図13　ボタンクリック サンプルプログラム

実行画面イメージ



重複エラーの場合、ポップアップメッセージを表示

登録が正常終了した場合、画面下部にお知らせメッセージ表示

ソース2　商品マスタメンテナンス新規登録(TEC010)

```
0001.00      /copy qcpylesrc,vvHspec
0002.00     ** ----------------------------------------------------------
0003.00     ** TEC010:商品マスタメンテナンス新規登録
0004.00     ** ----------------------------------------------------------
0005.00     F* ----------------------------------------------------------
0006.00     F* ファイル定義
0007.00     F* ----------------------------------------------------------
0008.00     F*<商品マスタ>
0009.00     FMSYOHIN   UF A E           K DISK
0010.00     F*
0011.00     d TEC010          pr
0012.00     d TEC010          pi
0013.00      /define nabButton
0014.00      /include qcpylesrc,vvNabTmpl
0015.00     ** ----------------------------------------------------------
0016.00     ** program start
0017.00     ** ----------------------------------------------------------
0018.00      /free
0019.00       Initialize();
0020.00       Process();
0021.00       CleanUp();
0022.00       *inlr=*on;
0023.00      /end-free
0024.00     ** ----------------------------------------------------------
0025.00     p Process         b
0026.00     d                 pi
0027.00     D VSYHNCD         S              10A
0028.00     D VSYHNNM         S              32A
0029.00     D VTANKA          S               9  0
0030.00     D*
0031.00      /free
0032.00       //フォーム上の値を取得
0033.00       VSYHNCD = GetFormChar('F1_SYHNCD');          //商品ＣＤ
0034.00       VSYHNNM = GetFormChar('F1_SYHNNM':'0');      //商品名
0035.00       VTANKA  = GetFormNum('F1_TANKA');            //単価
0036.00      /end-free
0037.00     C*
0038.00     C*-----キー重複チェック
0039.00     C     VSYHNCD       CHAIN     MSYOHIR                          81
0040.00     C     *IN81         IFEQ      *OFF
0041.00     C*
0042.00      /free
0043.00        //エラーメッセージを送信
0044.00        vvOut_toJsonPair('success:false,'
0045.00          + 'msg:入力した商品ＣＤは既に存在しています');
0046.00      /end-free
0047.00     C*
0048.00     C                   ELSE
0049.00     C*-----新規レコード登録
0050.00     C                   MOVEL     VSYHNCD       SYHNCD
0051.00     C                   MOVEL     VSYHNNM       SYHNNM
0052.00     C                   Z-ADD     VTANKA        TANKA
0053.00     C*
0054.00     C                   WRITE     MSYOHIR
0055.00     C*
0056.00      /free
0057.00        //正常終了メッセージを送信
0058.00        vvOut_toJsonPair('success:true,refresh:true,'
0059.00          + 'info:登録が終了しました');
0060.00      /end-free
0061.00     C*
0062.00     C                   END
0063.00     p                   e
0064.00      /include qcpylesrc,vvNabTmpl
```

1-①　削除

1-②

1-③

図14　ボタンクリック（EXNABBTN）新API

新API一覧

API	I/O	概要
GetAppVar	I	アプリ変数を取得
SetAppVar	O	アプリ変数をセット
SetResponse	O	レスポンスを返却

SetResponse パラメーター一覧

第1パラメータ	概要	記述例
success	処理の成否をセット	SetResponse('success':'true');
info	画面下部にトーストメッセージを表示	SetResponse('info':'終了しました。');
clearSelection	Grid行選択をクリア	SetResponse('clearSelection':'true');
refresh	データソース再読込	SetResponse('refresh':'true');
applyData	ウィジェットのフィールドに値をセット	SetResponse('applyData':'F1_NAME','尾崎　浩司');
disableFeature	対象機能を使用不可にする	SetResponse('disableFeature':'FuncSave');
enableFeature	対象機能を利用可能にする	SetResponse('enableFeature':'FuncSave');

ソース3　SetResponseメソッド（TEC010を変更）

```
0038.00     C     *IN81          IFEQ      *OFF                    1-②
0039.00     C*
0040.00     /free
0041.00       //エラーメッセージを送信
0042.00       SetResponse('success':'false');
0043.00       SetResponse('msg':'入力した商品ＣＤは既に存在しています');
0044.00     /end-free
0045.00     C*
0046.00     C                    ELSE
0047.00     C*-----新規レコード登録
0048.00     C                    MOVEL     VSYHNCD        SYHNCD
0049.00     C                    MOVEL     VSYHNNM        SYHNNM
0050.00     C                    Z-ADD     VTANKA         TANKA
0051.00     C*
0052.00     C                    WRITE     MSYOHIR                 1-③
0053.00     C*
0054.00     /free
0055.00       //正常終了メッセージを送信
0056.00       SetResponse('success':'true');
0057.00       SetResponse('refresh':'true');
0058.00       SetResponse('info':'登録が終了しました');
0059.00     /end-free
0060.00     C*
0061.00     C                    END
```

図15　ファイルアップロード サンプルプログラム

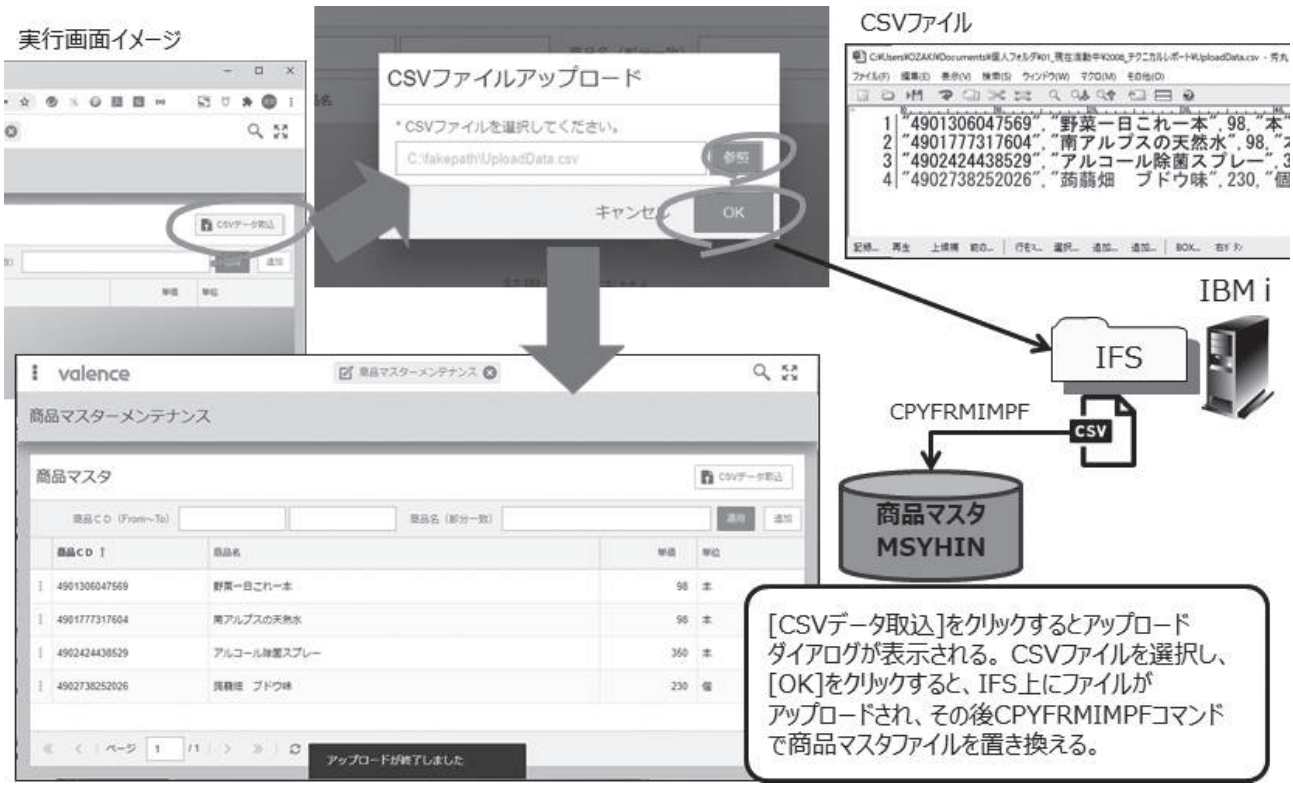
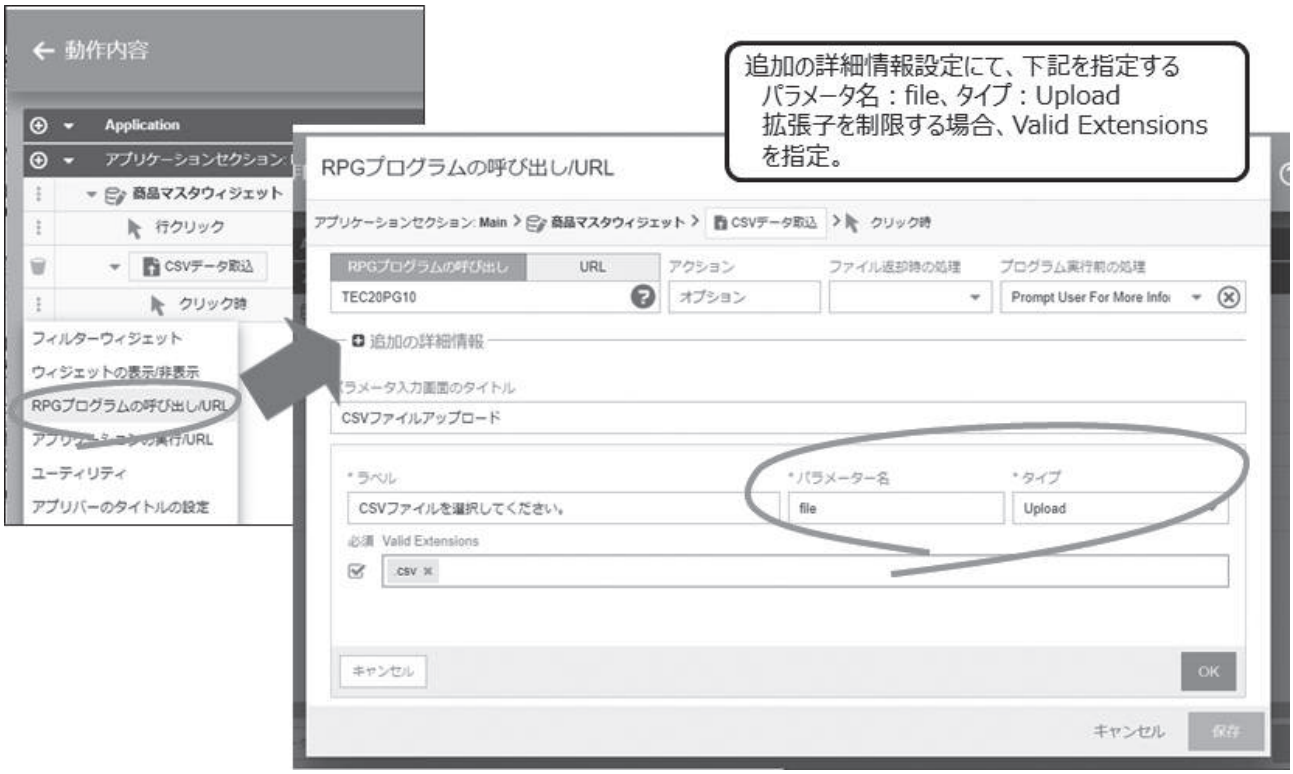


図16　ファイルアップロード RPGプログラム呼び出し設定



ソース4　CSVファイルアップロード(TEC20PG10)

```
0001.00      /copy qcpylesrc,vvHspec
0002.00     ** ------------------------------------------------------------
0003.00     ** TEC20PG10：ＣＳＶファイルアップロード
0004.00     ** ------------------------------------------------------------
0005.00     ** ------------------------------------------------------------
0006.00      /define nabButton
0007.00      /include qcpylesrc,vvNabTmpl
0008.00     ** ------------------------------------------------------------
0009.00     ** program start
0010.00     ** ------------------------------------------------------------
0011.00      /free
0012.00       Initialize();
0013.00       Process();
0014.00       CleanUp();
0015.00       *inlr=*on;
0016.00      /end-free
0017.00     ** ------------------------------------------------------------
0018.00     p Process         b
0019.00     d                 pi
0020.00     D TMPPATH         S             60A
0021.00     D FILENM          S             60A
0022.00     D CMDSTR          S            255A
0023.00     D*
0024.00      /free                                              2-①
0025.00        //ファイル保管先ＩＦＳディレクトリの指定
0026.00        TMPPATH = vvUtility_getValenceSetting('TEMP_PATH');
0027.00
0028.00        //ＩＦＳへファイルアップロードを実施
0029.00        vvIn.path = %trim(TMPPATH);                      2-②
0030.00        vvIn.ccsid = 943;
0031.00        vvIn_file(vvIN: '*NULL');
0032.00
0033.00        //アップロードしたファイル名を取得
0034.00        FILENM = %trim(TMPPATH) + %trim(vvIn.fileName);
0035.00                                                         2-③
0036.00      /end-free
0037.00     C*----- CPYFRMIMPFコマンドで物理ファイルMSYHINに転送
0038.00     C*----- CPYFRMIMPFコマンド文字列の作成
0039.00     C                   EVAL      CMDSTR =
0040.00     C                               'CPYFRMIMPF '
0041.00     C                             + 'FROMSTMF(''' + %trim(FILENM) + ''') '
0042.00     C                             + 'TOFILE(MSYHIN) '
0043.00     C                             + 'MBROPT(*REPLACE) FROMCCSID(943) '
0044.00     C                             + 'RCDDLM(*CRLF) ERRRCDOPT(*REPLACE) '
0045.00     C
0046.00     C*----- CPYFRMIMPFコマンドを実行
0047.00     C                   CALL      'QCMDEXC'
0048.00     C                   PARM                    CMDSTR
0049.00     C                   PARM      255           CMDLEN          15 5
0050.00     C*
0051.00      /free
0052.00        //-----アップロードしたＣＳＶファイルを削除
0053.00        vvIfs_deleteFile(%trim(FILENM));
0054.00
0055.00        //-----レスポンスを返す
0056.00        SetResponse('success':'true');
0057.00        SetResponse('info':'アップロードが終了しました');
0058.00      /end-free
0059.00     p                 e
0060.00      /include qcpylesrc,vvNabTmpl
```

図17　セキュリティ機能 サンプルプログラム

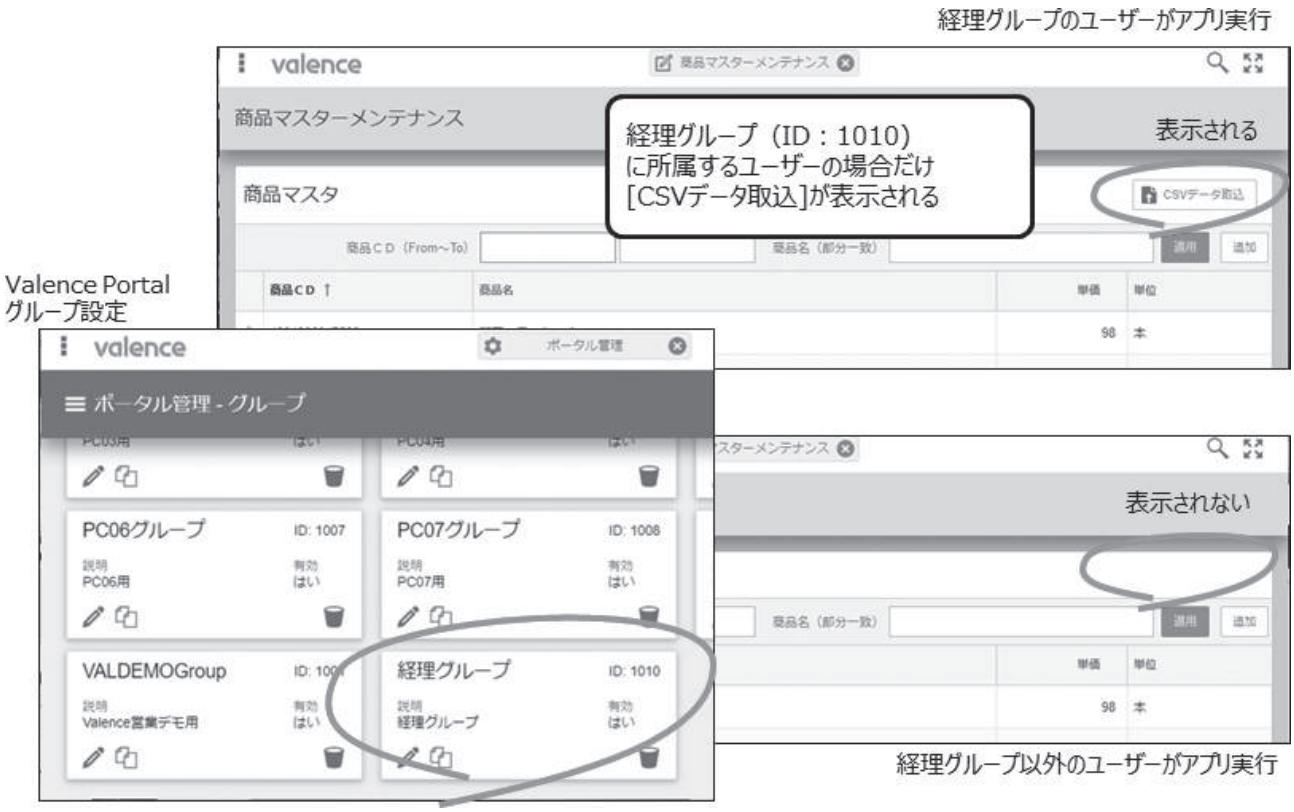


図18　セキュリティ機能 設定画面

ソース5　スタートアッププログラム（TEC20PG20）

```
0001.00      /copy qcpylesrc,vvHspec
0002.00     ** ------------------------------------------------------------
0003.00     ** TEC20PG20：スタートアッププログラム
0004.00     ** ------------------------------------------------------------
0005.00     ** ------------------------------------------------------------
0006.00      /define nabStartup
0007.00      /include qcpylesrc,vvNabTmpl
0008.00     ** ------------------------------------------------------------
0009.00     ** program start
0010.00     ** ------------------------------------------------------------
0011.00      /free
0012.00       Initialize();
0013.00       Process();
0014.00       CleanUp();
0015.00       *inlr=*on;
0016.00      /end-free
0017.00     ** ------------------------------------------------------------
0018.00     p Process         b
0019.00     d                 pi
0020.00      /free
0021.00       //------"経理グループ"（グループID:1010）で無い場合
0022.00       if not vvSecure_isMember(1010);
0023.00         //----- CSVデータ取込　ボタン利用不可
0024.00         disableFeature('UploadData');
0025.00       endif;
0026.00      /end-free
0027.00     p                 e
0028.00      /include qcpylesrc,vvNabTmpl
```

3-①

図19　Formウィジェット ヘルパープログラム サンプル

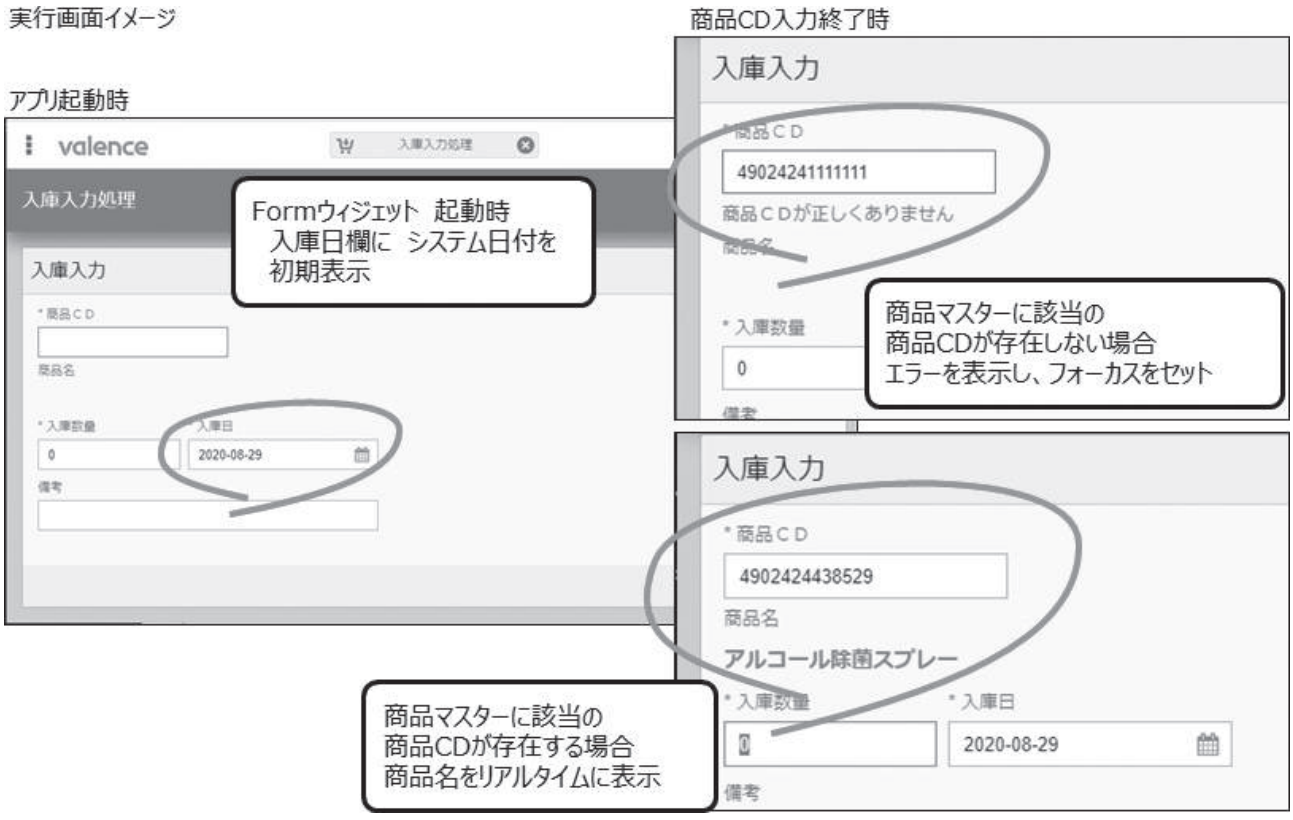
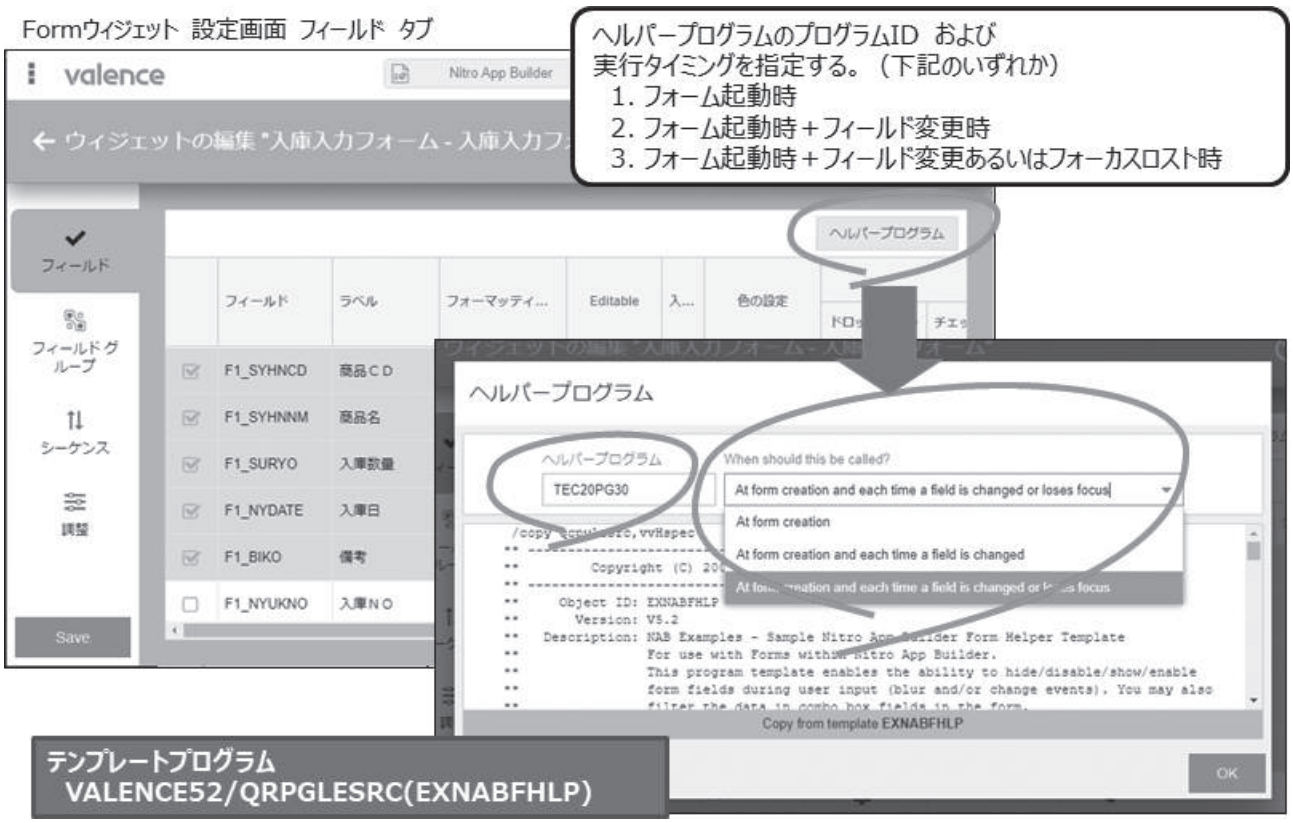


図20　Formウィジェット ヘルパープログラム 設定



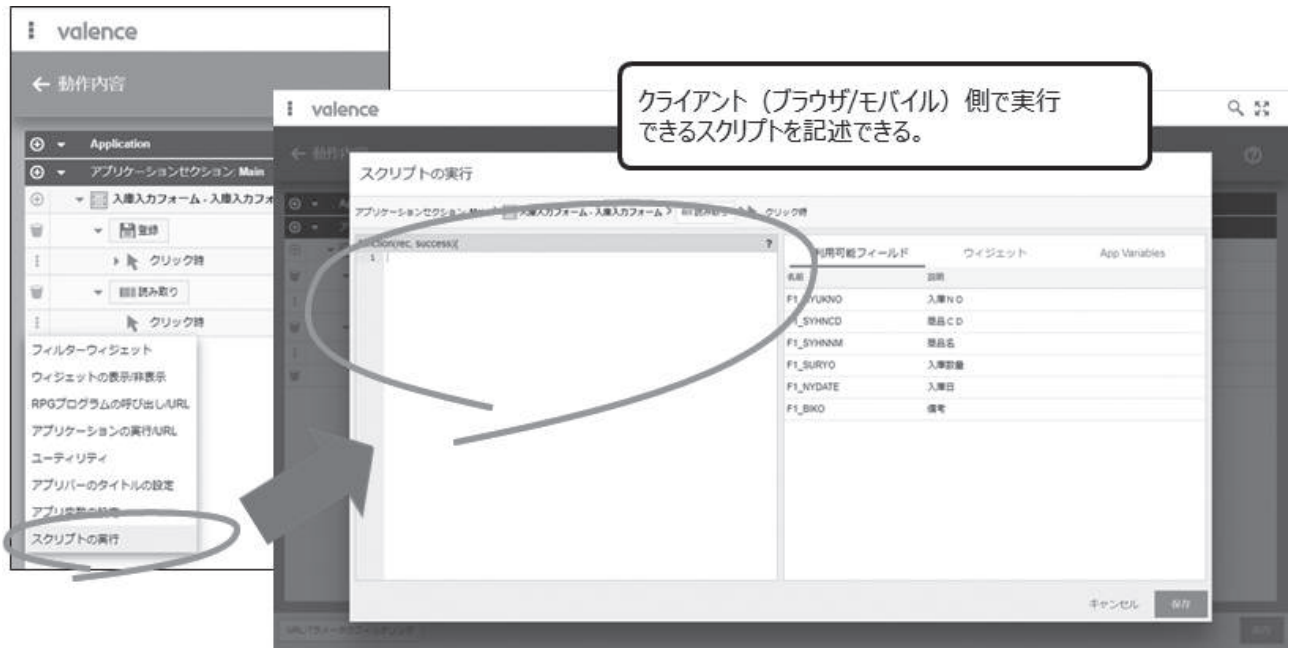
ソース6　ヘルパープログラム(TEC20PG30)

```
0001.00      /copy qcpylesrc,vvHspec
0002.00     ** ------------------------------------------------------------
0003.00     ** TEC20PG30：フォームヘルパープログラム
0004.00     ** ------------------------------------------------------------
0005.00     F*<商品マスタ>
0006.00     FMSYHIN    IF   E           K DISK
0007.00     ** ------------------------------------------------------------
0008.00      /define nabFormHelper
0009.00      /include qcpylesrc,vvNabTmpl
0010.00     ** ------------------------------------------------------------
0011.00     ** program start
0012.00     ** ------------------------------------------------------------
0013.00      /free
0014.00       Initialize();
0015.00       Process();
0016.00       CleanUp();
0017.00       *inlr=*on;
0018.00      /end-free
0019.00     ** ------------------------------------------------------------
0020.00     p Process         b
0021.00     d                 pi
0022.00     D VSYCD           S             20A
0023.00     D VSYNM           S             60A
0024.00     D VERFG           S              1A
0025.00     C*-----初期化
0026.00     C                   MOVEL     *BLANK         VSYNM
0027.00     C                   MOVEL     *BLANK         VERFG
0028.00     C*
0029.00     C*-----フォーム作成時処理                          4-③
0030.00     C     sMode         IFEQ      'formRender'
0031.00      /free
0032.00        //-----入庫日欄にシステム日付を初期セット
0033.00        SetValue('F1_NYDATE':%CHAR(%DATE()));
0034.00      /end-free                                         4-④
0035.00     C                   ENDIF
0036.00     C*
0037.00     C*-----フィールド変更時処理
0038.00     C     sField        IFEQ      'F1_SYHNCD'          4-⑤
0039.00      /free
0040.00        //----- 画面上の商品ＣＤ取得
0041.00        VSYCD = vvIn_char('F1_SYHNCD');                 4-⑥
0042.00      /end-free
0043.00     C*-----商品マスタに該当商品ＣＤが存在するか確認する
0044.00     C     VSYCD         IFNE      *BLANK
0045.00     C     VSYCD         CHAIN     MSYHIR                             91
0046.00     C     *IN91         IFEQ      *OFF
0047.00     C                   MOVEL     SYSYNM         VSYNM
0048.00     C                   ELSE
0049.00     C                   MOVEL     '1'            VERFG
0050.00     C                   ENDIF
0051.00     C                   ENDIF
0052.00      /free
0053.00        //-----結果を画面上の商品名欄にセットする
0054.00        SetValue('F1_SYHNNM':VSYNM);
0055.00
0056.00        //-----商品ＣＤが存在しない場合エラーを返す
0057.00        if VERFG <> *BLANK;
0058.00         SetError('F1_SYHNCD':'商品ＣＤが正しくありません');
0059.00         FocusField('F1_SYHNCD');
0060.00        endif;                                          4-⑦
0061.00      /end-free
0062.00     C                   ENDIF
0063.00     C*
0064.00     p                 e
0065.00      /include qcpylesrc,vvNabTmpl
```

4-①

4-②

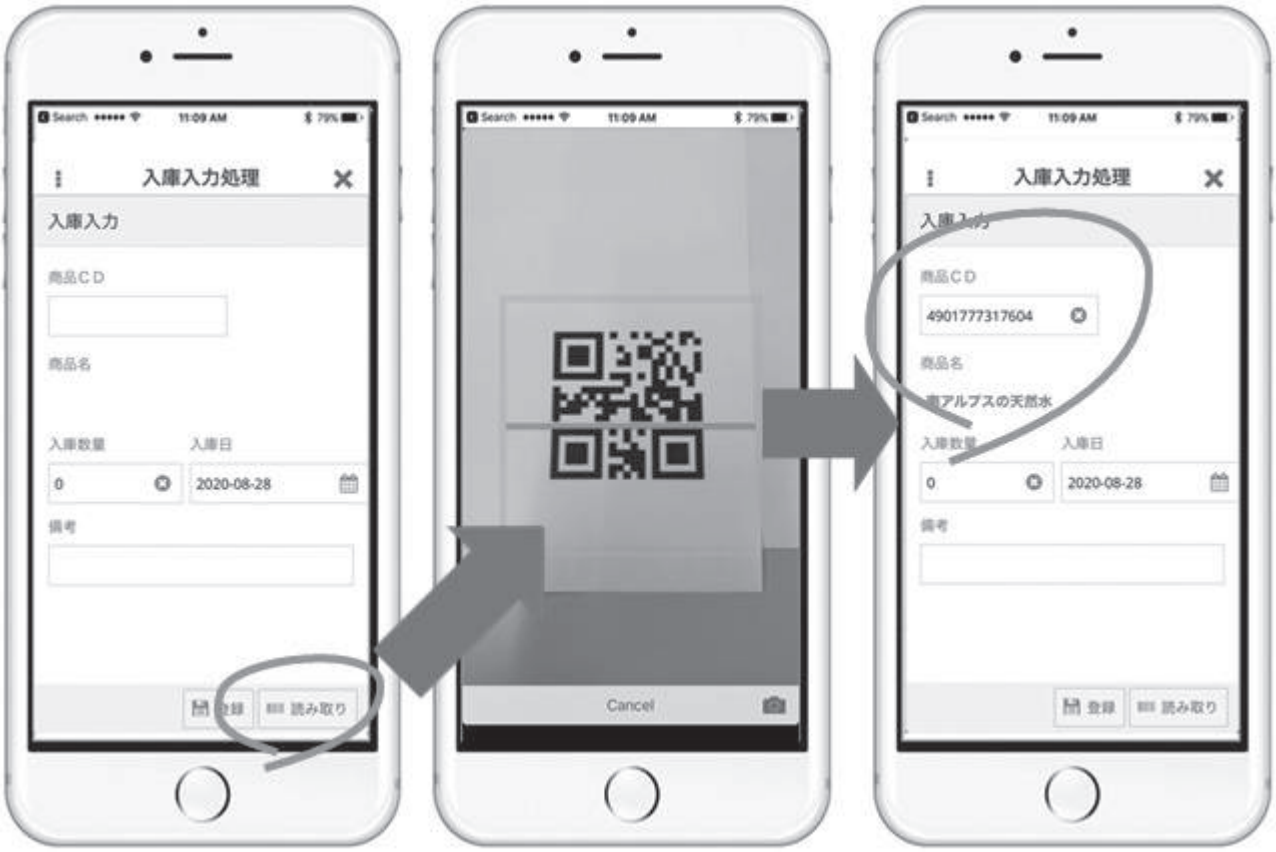
図21　スクリプトの実行設定



ソース7　スクリプト例（QR／バーコードリーダー連携）

```
function(rec, success){
if (Valence.mobile.Access.isNativePortal()) {
    Valence.mobile.Barcode.scan({
        callback : function (response) {
            if (Ext.isEmpty(response)) {
                success(); return;
            }
            if (response.success) {
                if (!response.data.cancelled) {
                    cmp.setValues({
                        F1_SYHNCD : response.data.text
                    });
                    success();

                } else {
                    Ext.Msg.alert('バーコードスキャン', 'キャンセル');
                    success();
                }
            }
        }

    });
} else {
    Ext.Msg.alert('バーコードスキャン', 'モバイルアプリの場合のみ使用可能です');
    success();
}

```

図22　バーコード／QRコード読み取り



Migaro. Technical Report
既刊号バックナンバー

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

No.1　2008 年秋

お客様受賞論文

●最優秀賞

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

●ゴールド賞

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

●シルバー賞

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

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

●優秀賞

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

技研化成の新基幹システム再構築
藤田 健治 様／技研化成株式会社

SE 論文

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

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

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

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

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

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

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

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

No.2　2009 年秋

お客様受賞論文

●最優秀賞

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

●ゴールド賞

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

●シルバー賞

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

●優秀賞

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

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

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

SE 論文

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

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

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

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

RPG でパフォーマンスを制御
松尾 悦郎／システム事業部 システム１課

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

No.3　2010 年秋

お客様受賞論文

●最優秀賞

建物のクレーム情報管理システム「アフターサービス DB」について

大橋 良之 様／東レ建設株式会社

●ゴールド賞

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

寒河江 幸喜 様／日綜産業株式会社

●シルバー賞

Delphi/400 で鉄鋼受発注業務を統一し 鉄鋼 EDI も実現

柿本 直樹 様／合鐵産業株式会社

●優秀賞

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

小島 栄一 様／西川計測株式会社

イントラでの PHP-Delphi-RPG 連携

仲井 学 様／西川リビング株式会社

Delphi/400 を使った取引先管理システム

大崎 貴昭 様／森定興商株式会社

SE 論文

Delphi/400 ローカルキャッシュ活用術

中嶋 祥子／ RAD 事業部 技術支援課

Delphi/400 帳票開発ノウハウ公開

尾崎 浩司／システム事業部 システム３課

Delphi/400 でドラッグ&ドロップを制御

辻林 涼子／システム事業部 システム２課

Delphi/400 のモジュールバージョン管理手法

前田 和寛／システム事業部 システム２課

Delphi/400　Web からの PDF 出力

福井 和彦・清水 孝将／システム事業部システム３課・システム２課

Delphi/400 で Flash 動画の実装

吉原 泰介／ RAD 事業部 技術支援課 顧客サポート

No.4　2011 年秋［創立 20 周年記念号］

お客様受賞論文

●最優秀賞

全社の経費処理業務を効率化した「e 総務システム」

鈴木 英明 様／阪和興業株式会社

●ゴールド賞

「Web 進捗管理システム」でリアルタイム性を実現

堀内 一弘 様／エスケーロジ株式会社

●シルバー賞

「営業奨励金申請書」をたった２日間で開発

簑島 宏明 様／株式会社ケーユーホールディングス

液体輸送における「配車支援システム」の構築

桂 哲 様／ライオン流通サービス株式会社

SE 論文

グラフ活用リファレンス

中嶋 祥子／ RAD 事業部 技術支援課

Web サービスを利用して機能 UP ！

福井 和彦・畑中 侑／システム事業部 システム２課

OpenOffice 実践活用

吉原 泰介／ RAD 事業部 技術支援課 顧客サポート

VCL for the Web 活用 TIPS 紹介

尾崎 浩司／システム事業部 プロジェクト推進室

JC/400 で JavaScript 活用

清水 孝将／システム事業部 システム１課

jQuery 連携で機能拡張

國元 祐二／ RAD 事業部 技術支援課 顧客サポート

No.5　2012 年秋［創刊 5 周年記念］

お客様受賞論文

【部門 1】

●最優秀賞

JC/400 による取引先との Web-EDI システム構築

久保田 佳裕 様／極東産機株式会社

●ゴールド賞

Delphi と Excel を使用した帳票コストの削減

大久保 治高 様／合鐵産業株式会社

もっと見やすく、もっと使いやすい画面を

新谷 直正 様／株式会社アダル

【部門 2】

●優秀賞

Delphi/400 で確認業務の効率化

為国 順子 様／ベネトンジャパン株式会社

取引先申請システムでの稟議書作成ワークフロー

大崎 貴昭 様／森定興商株式会社

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

島根 英行 様／シルフ

SE 論文

InstallAware を使った Delphi/400 運用環境の構築

中嶋 祥子／ RAD 事業部 技術支援課 顧客サポート

カスタマイズコンポーネント入門 Delphi/400 開発効率向上

前田 和寛／システム事業部 システム 2 課

Delphi/400 スマートデバイスアプリケーション開発

吉原 泰介／ RAD 事業部 技術支援課 顧客サポート

DataSnap を使用した 3 層アプリケーション構築技法

尾崎 浩司／システム事業部 プロジェクト推進室

JC/400 でポップアップウィンドウの制御＆活用ノウハウ

清水 孝将・伊地知 聖貴／システム事業部 システム 1 課

【創刊 5 周年記念】

ミガロ.SE 座談会―お客様と共に歩む、お客様への熱い思い

No.6　2013 年秋

お客様受賞論文

【部門 1】

●最優秀賞

自社用開発フレームワークの構築

駒田 純也 様／ユサコ株式会社

●ゴールド賞

Delphi/400 で CTI 開発および関連機能組み込み

仲井 正人 様／株式会社スマイル・ジャパン

●シルバー賞

IBM WebFacing から JC/400 への移行・リニューアル手法

八木 秀樹 様／極東産機株式会社

Delphi/400 と Delphi を利用した IBM i 資源の有効活用

小山 祐二 様／澁谷工業株式会社

発注システムを VB から Delphi へ移植しリニューアル

川島 寛 様／株式会社タツミヤ

【部門 2】

●優秀賞

5250 画面を使用せずに AS/400 スプールファイルをコントロールする

白井 昌哉 様／太陽セメント工業株式会社

Delphi/400 を利用した 承認フロー導入による IT 内部統制構築

塚本 圭一 様／ライオン流通サービス株式会社

SE 論文

FastReport を使用した帳票作成入門

尾崎 浩司／ RAD 事業部 営業推進課

Delphi/400 で開発する 64bit アプリケーション

吉原 泰介／ RAD 事業部 技術支援課 顧客サポート

Web コンポーネントのカスタマイズ入門

佐田 雄一／システム事業部 システム 1 課

Indy を利用したメール送信機能開発

辻野 健・前坂 誠二／システム事業部 システム 2 課

Windows テキストファイル操作ノウハウ

小杉 智昭／システム事業部 プロジェクト推進室

JC/400 Web アプリケーションのユーザー管理・メニュー管理活用術

吉原 泰介・國元 裕二／ RAD 事業部 技術支援課 顧客サポート

No.7　2014 年秋

お客様受賞論文

【部門 1】

●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

●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

小山 祐二 様／澁谷工業株式会社

●シルバー賞

IBM i と Delphi/400 のコラボレーション

新谷 直正 様／株式会社アダル

●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

【部門 2】

●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

●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

SE 論文

iOS/Android ネイティブアプリケーション入門

吉原 泰介／ RAD 事業部 技術支援課

ファイル加工プログラミングテクニック

小杉 智昭／システム事業部 プロジェクト推進室

FastReport を使用した帳票作成テクニック

前坂 誠二／システム事業部

大量データ処理テクニック

佐田 雄一／システム事業部

スマートデバイス WEB アプリケーション入門

尾崎 浩司／ RAD 事業部 営業推進課

國元 祐二／ RAD 事業部 技術支援課

No.8　2015 年秋

お客様受賞論文

【部門 1】

●最優秀賞

iPod Touch の業務利用開発と検証

石井 裕昭 様／豊鋼材工業株式会社

●ゴールド賞

ブランク加工図管理システムの構築

小山 祐二 様／澁谷工業株式会社

●シルバー賞

Delphi/400 でスプールファイル管理（WRKSPLF コマンドの活用）

三好 誠 様／ユサコ株式会社

●シルバー賞

予算管理システムの構築

川島 寛 様／株式会社タツミヤ

●シルバー賞

送状データ送信システムの Web 化について

仲井 学 様／西川リビング株式会社

【部門 2】

●優秀賞

繰り返し DB 参照時の ClientDataSet の First 機能について

牛嶋 信之 様／株式会社佐賀鉄工所

●優秀賞

IBM i のカレンダーを基準に他のシステムを稼働

福島 利昭 様／株式会社ランドコンピュータ

SE 論文

フレームを利用した開発手法

前坂 誠二／システム事業部 システム 2 課

Windows タブレット用にカスタムソフトウェアキーボードを実装

福井 和彦／システム事業部 プロジェクト推進室

マルチスレッドを使用したレスポンスタイム向上

尾崎 浩司／ RAD 事業部 営業・営業推進課

Android アプリケーションの NFC 機能活用

吉原 泰介／ RAD 事業部 技術支援課 顧客サポート

スマートデバイス開発で役立つ画面拡張テクニック

國元 祐二／ RAD 事業部 技術支援課 顧客サポート

No.9　2016 年秋

お客様受賞論文

【部門 1】

●最優秀賞

IBM i の見える化で実現するアジャイル開発

吉岡 延泰 様／日本調理機株式会社社

●ゴールド賞

Windows Like 5250 への道のり

小山 祐二 様／澁谷工業株式会社

●シルバー賞

Delphi プログラム管理ソフトの開発

牛嶋 信之 様／株式会社佐賀鉄工所

【部門 2】

●優秀賞

Delphi/400 を利用した定型業務の PDF 化

佐藤 岳 様／ライオン流通サービス株式会社

●優秀賞

ちょい足しモバイル

仲井 正人 様／株式会社スマイル・ジャパン

●優秀賞

AS/400 の受注データを Web で社員に公開

福島 利昭 様／株式会社ランドコンピュータ

SE 論文

iOS モバイルアプリ開発のデザイニングテクニック

前坂 誠二／システム事業部 システム 2 課

新データベースエンジン FireDAC を使ってみよう！

福井 和彦／システム事業部 プロジェクト推進室

Delphi/400 最新プログラム文法の活用法

尾崎 浩司／ RAD 事業部 営業・営業推進課

FastReport を活用した電子帳票作成テクニック

宮坂 優大／システム事業部 システム 1 課

Beacon 技術による IoT 活用の第一歩

吉原 泰介／ RAD 事業部 技術支援課 顧客サポート

Web & ハイブリッドアプリ開発で役立つ IBM i & ブラウザデバッグテクニック

國元 祐二／ RAD 事業部 技術支援課 顧客サポート

No.10　2017 年秋

Migaro.Technical Report 創刊 10 周年記念

パートナー様からの祝辞

武藤 和博 様／日本アイ・ビー・エム株式会社

藤井 等 様／エンバカデロ・テクノロジーズ日本法人

Serge Charbit 様／ SystemObjects Corporation

飯田 恭子 様／アイマガジン株式会社

お客様からの祝辞・お客様の声

石井 裕昭 様／豊鋼材工業株式会社

牛嶋 信之 様／株式会社佐賀鉄工所

大崎 貴昭 様／森定興商株式会社

川島 寛 様／株式会社タツミヤ

久保田 佳裕 様／極東産機株式会社

駒田 純也 様／ユサコ株式会社

小山 祐二 様／澁谷工業株式会社

寒河江 幸喜 様／日綜産業株式会社

佐々木 仁志 様／株式会社ジャストオートリーシング

佐藤 岳 様／ライオン流通サービス株式会社

白井 昌哉 様／太陽エコブロックス株式会社

仲井 学 様／西川リビング株式会社

福島 利昭 様／株式会社ランドコンピュータ

お客様座談会

石井 裕昭 様／豊鋼材工業株式会社

駒田 純也 様／ユサコ株式会社

寒河江 幸喜 様／日綜産業株式会社

仲井 学 様／西川リビング株式会社

上甲 將隆／株式会社ミガロ.

司会　飯田 恭子 様／アイマガジン株式会社

お客様受賞論文

【部門 1】

●最優秀賞

貸金庫と鍵のマッチング業務を Delphi/400 で実現
―文字認識データと基幹システムデータを統合

佐藤 正 様／株式会社富士精工本社

●ゴールド賞

Windows タブレット導入による工作部材料受入業務改革

小山 祐二 様／澁谷工業株式会社

【部門 2】

●優秀賞

Delphi/400 を利用した各拠点 PING コマンド簡素化

松垣 秀昭 様／ライオン流通サービス株式会社

汎用的な帳票出力画面

牛嶋 信之 様／株式会社佐賀鉄工所

バーコードリーダー読み取り後、次の入力位置にカーソルを自動遷移させる技術

上総 龍央 様／キョーラクシステムクリエート株式会社

IBM i のスプールファイル参照機能を Web で構築

福島 利昭 様／株式会社ランドコンピュータ

SE 論文

デスクトップアプリケーション開発でも
役立つ FireMonkey 活用入門

尾崎 浩司／ RAD 事業部 営業・営業推進課

Delphi/400 バージョンアップに伴う文字コードの違いと制御

宮坂 優大／システム事業部 システム 1 課

FastReport への効率的な帳票レイアウトコンバート

畑中 侑／システム事業部 システム 2 課

IBM i トリガー機能を活かしたセキュリティログ対応

八木沼 幸一／システム事業部 プロジェクト推進室

アプリケーションテザリングを利用した PC &モバイルアプリケーション連携

吉原 泰介／ RAD 事業部 技術支援課 顧客サポート

SmartPad4i の運用で役立つ WEB サーバー機能

國元 祐二／ RAD 事業部 技術支援課 顧客サポート

No.11　2018 年秋

お客様受賞論文

【部門 1】

●最優秀賞

Excel テンプレートを使用した帳票出力機能の開発

駒田 純也 様／ユサコ株式会社

●ゴールド賞

SP4i の活用による製品検査チェックシステムの構築

八木 秀樹 様／極東産機株式会社

【部門 2】

●優秀賞

配車支援システムを Delphi/400 で再構築

村上 稔明 様／ライオン流通サービス株式会社

一般シール受注入力業務の Delphi/400 化

寺西 健一 様／大阪シーリング印刷株式会社

Delphi/400 による無線ハンディターミナルのデータ集約の仕組みの実装

寺西 健一 様／大阪シーリング印刷株式会社

SE 論文

OLE を利用した Excel 出力のパフォーマンス向上手法

薬師 尚之／システム事業部 システム 2 課

FireDAC 実践プログラミングテクニック

佐田 雄一／システム事業部 システム 1 課

REST による Web サービスを活用した機能拡張テクニック

尾崎 浩司／ RAD 事業部 営業・営業推進課

Google Maps Platform を使用したアプリケーション開発テクニック

福井 和彦・小杉 智昭／システム事業部 プロジェクト推進室

RAD Server を使った新しい多層アプリケーション構築

吉原 泰介／ RAD 事業部 技術支援課

JC/400 から SP4i へのマイグレーションノウハウ

吉原 泰介・國元 祐二／ RAD 事業部 技術支援課

No.12　2019 年秋

お客様受賞論文

【部門 1】

●最優秀賞

Delphi/400 による電子帳簿保存法スキャナ保存制度への挑戦

石井 裕昭 様／豊鋼材工業株式会社

●ゴールド賞

出荷業務従事者のモチベーションアップ大作戦ー
Delphi/400 で基幹データの見える化

池田 純子 様／錦城護謨株式会社

●ゴールド賞

iPhone を利用した宝飾品の販売系システムを SP4i で構築

島本 佳昭 様／株式会社大月真珠

【部門 2】

●優秀賞

SmartPad4i による、導入後の改修を意識した設計

西村 直也 様／株式会社ジャストオートリーシング

●優秀賞

Valence を使用した 温度・湿度ログ表示
ーサーバー室内温度・湿度変化の見える化ー

中谷 佳史 様／株式会社保健科学西日本

●優秀賞

RPG ソースコードを Valence File Editor で改修

福島 利昭 様／株式会社ランドコンピュータ

特別寄稿論文

統合開発環境 Cobos のご紹介ー
RPG・SP4i の効率的な開発に

SE 論文

IBM i データベースへの FTP 送信手法の紹介

田村 洋一郎・宮坂 優大・都地 奈津美／システム事業部 システム 1 課

FireMonkey の活用　カメラコンポーネントを使ったアプリ

畑中 侑／システム事業部 システム 2 課

Subversion を使用した Delphi ソース管理

福井 和彦／システム事業部 プロジェクト推進室

Enterprise Connectors を利用した
クラウド連携テクニック

佐田 雄一／ RAD 事業部 技術支援課

SmartPad4i のインターフェース機能拡張

國元 祐二／ RAD 事業部 技術支援課

Valence App Builder　RPG 連携テクニック

尾崎 浩司／ RAD 事業部 技術支援課

MIGARO.
TECHNICAL
REPORT

Migaro.Technical Report
No.13　2020 年秋
ミガロ.テクニカルレポート

2020 年 12 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
https://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.

https://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

東京事業所
〒100-0013
東京都千代田区霞が関3-7-1
霞が関東急ビル 2F
TEL:03(5510)5701
FAX:03(5510)5702