尾崎 浩司

株式会社ミガロ.

システム事業部 プロジェクト推進室

Delphi/400:VCL for the Web活用TIPS紹介

Web アプリケーション開発に役立つ 4 つの TIPS。「VCL for the Web」を用いることで GUI 開発と同様、生産性の高い開発が実現できる。

略歴
1973 年 08 月 16 日生
1996 年三重大学工学部卒
1999 年 10 月株式会社ミガロ . 入社
1999 年 10 月システム事業部配属

現在の仕事内容
ミガロ . 入社以来、主に Delphi/400 を利用した受託開発を担当している。

- VCL for the Webとは
- Ajaxを使用したアプリケーション構築方法
- 動的に生成したファイルのダウンロード
- コネクションプーリングを使用した Delphi/400アプリ開発
- スマートフォン対応ページの作成方法
- まとめ

1. VCL for the Webとは

「VCL for the Web（旧 IntraWeb)」とは、Delphi/400 に付属する Web アプリケーション開発ツールのことである。

この VCL for the Web は、通常の VCL フォームアプリケーションと同様にフォーム上へコンポーネントを貼り付け、必要に応じプロパティやイベントを設定することで開発を行えることが特徴である。

ミガロ . では、Web アプリケーション開発において積極的にこの VCL for the Web を使用しており、Delphi/400 における Web アプリケーション開発のノウハウも蓄積されつつある。

本稿では、これらノウハウの中から役に立つであろう 4 つの TIPS について、具体例を交えながら紹介しようと思う。

※ VCL for the Web をはじめとする Delphi/400 における Web アプリケーションの基本的な開発手順については、ミガロ . のホームページに詳しく紹介しているのでそちらを参照していただきたい。

【Delphi/400 の技術情報（インターネット）に関する Tips】
http://www.migaro.co.jp/contents/products/delphi400/tips/web/index.html

2.Ajaxを使用したアプリケーション構築方法

VCL for the Web では「Ajax」を使用することが可能である。Ajax とは「Asynchronous JavaScript + XML」の略で、Web ページのリロードを伴わずに、Web サーバーと XML 形式のデータのやり取りを行って処理を進めていく対話型 Web アプリケーションのことである。

従来の Web アプリケーションは、図 1a のように入力後ボタン等を押下することで、サブミットを行い、新しいページを要求するという構成であった。【図 1a】

Ajax を使用すると、図 1b のようにページを切り替えることなく、入力中にどんどん最新情報に更新していくことが可能になるのである。【図 1b】

この仕組みを VCL for the Web では使用することができる。しかも、JavaScript を使用せず、Delphi 言語をそのまま使用することが可能であるため、とても魅力的である。

では、VCL for the Web での Ajax 使用方法を見てみよう。

図 2 は、入力用コンポーネントである IWEdit のイベント一覧である。【図 2】

この中にある「OnAsync 〜」というイベントが、Ajax に関連するものである。通常のコンポーネントと同様のイベントが用意されていることがわかるであろう。つまり、通常のイベントと同じ感覚で処理を記述することができるのである。

具体例として、図 3 のような仕様を検

図1a


サブミットすることで
リクエスト送信
Webサーバー
新しいページをレスポンスとして返す

図1b


キー入力時に都度リクエスト
Webサーバー
ページ自体は変わらず、画面変更部分だけが更新

図2


オブジェクト インスペクタ
IWEdit1 TIWEdit
プロパティ
イベント
CustomHint
OnAsyncChange
OnAsyncClick
OnAsyncDoubleClick
OnAsyncEnter
OnAsyncExit
OnAsyncKeyDown
OnAsyncKeyPress
OnAsyncKeyUp
OnAsyncMouseDown
OnAsyncMouseMove
OnAsyncMouseOut
OnAsyncMouseOver
OnAsyncMouseUp
OnHTMLTag
OnSubmit
IntraWeb (VCL for the Web) 10

討する。【図3】

顧客コード欄（edtCustCd）にフォーカスインしたタイミングとフォーカスアウトしたタイミングとでイベント処理を行い、画面上の値が変更された際に顧客マスターより値を取得すればよいのである。【ソース1】

このプログラムを実行すると、図4のようになる。【図4】

このようにAjaxを使用すると、都度画面をリロードしなくても対話できるアプリケーションが開発できるので、とても便利なのである。

3.動的に生成したファイルのダウンロード

Webアプリケーションを作成していると、ファイルをダウンロードさせたい場面があるだろう。

この際、あらかじめ特定のフォルダに用意されたファイルをダウンロードするだけならば、WebApplication変数にあるSendFileメソッドを実行すればよいだけである。

しかしながら、データベースから抽出した結果等をダウンロードさせるとなると、動的にファイルを生成しなければならない。Webアプリケーションは Webサーバー上で実行されるため、クライアント側の要求ごとにWebサーバー側に動的にファイルを生成するというのはファイル名の重複等の問題もあり、難しいのではないだろうか。

では、どうすれば抽出結果から、ファイルをクライアント側にダウンロードさせることができるのだろうか。このような場合、サーバーではファイルではなくストリームとして作成し、クライアントへデータ転送を行えばよいのである。

具体例を検討してみよう。図5を見ていただきたい。【図5】

これは、データベースから取得した結果を、IWGridに表示している例である。この画面にあるCSV出力リンクをクリックすると、画面表示されているグリッド内容をクライアントにCSVファイルとして転送させるのである。

ソースコードを紹介しよう。【ソース2】

このプログラムでは、CSVデータを作成するために文字列リスト（TStringList）を使用している。この文字列リストには、ストリームオブジェクトに書き込むSaveToStreamメソッドがあるので、メモリーストリームに保存できる。そうすることで、CSVデータを直接ストリーム化することができるのである。

さらに、作成したストリームはWebApplication変数のSendStreamメソッドを使用すれば、クライアントにダウンロードさせることができる。

この方式を使用すれば、Webサーバー側にファイルを生成することが一時的にも生じることなく、クライアント側にダウンロードさせることが可能になる。

Webアプリケーションでは、CSV形式のダウンロードが要求される場面が多いと思うので、本ロジックを開発時の参考にしていただきたい。

なお、このプログラムをそのまま実行してしまうと、VCL for the Webの画面ロックの制約により、ダウンロード後、画面上の動作が一切受け付けられなくなってしまう。これを解消するには、ダウンロード対象となるIWLinkコンポーネントのOnScriptEventsイベントに、設定が必要となるので注意してほしい。【図6】

4.コネクションプーリングを使用したDelphi/400アプリ開発

「コネクションプーリング」とは、データベースへアクセスの都度、接続（コネクション）を確立するのではなく、あらかじめ事前に一定数のコネクションを確立しておき、それを使い回す手法のことである。データベースアクセスの負荷を減らすために用いられる。

コネクションプーリングはWebアプリケーション開発でよく使用する手法だが、これをVCL for the Webでも使用することが可能である。

VCL for the Webで、コネクションプーリングを使用する手順を紹介しよう。

新規作成にて「VCL for the Web Application Wizard」を選択すると、ウィザードが起動する。この画面で、［Options］→［Pool Data Connections］のチェックをONにすればよい。【図7】

このオプションを設定してプロジェクトを新規作成すると、図8のようになる。【図8】

ServerController.pasに、Poolと命名されたIWDataModulePoolコンポーネントが設定されているのがわかる。このPoolコンポーネントには、PoolCountというプロパティが用意されており、ここに指定した数の接続が維持されるのである。（Delphi/400の場合、PoolCountに指定した数だけジョブが生成されると考えればよい。）

では、このコネクションプーリングが使用できる環境下で、実際にデータベースにアクセスするにはどうすればよいのだろうか？

データベース接続（コネクション）を複数のWebセッションで共有して使用するため、使用開始する際に明示的にコネクションをロックし、使用が終了したらロックを解除する必要がある。

実際の使用例をソース3に示すので参考にしてほしい。【ソース3】

コネクションプーリング制御下でデータベースにアクセスする場合、アクセス対象とするデータモジュールをロックするために、変数を定義する。（ソース3では、ADataModuleという変数を宣言。）

そして、データベースにアクセスする直前でLockDataModule関数を実行すると、その時点で空きのあるコネクションを探し、ロックを行う。

使用が終了したら、UnlockDataModule手続きでアンロックする、という流れである。（これらの関数/手続きは、ServerControler.pasに処理が記述されているため、使用する際には、データモジュールユニットと共にユニット参照を行うこと。）

このようなロジックを追加することで、Webアプリケーションに対し、コネクションプーリング制御ができるのでぜひ検討してみてほしい。

なお、CO400Connectionドライバ（dbExpress接続）でコネクションプーリングを行う際、データモジュール生成時（OnCreate）に、あらかじめTSQLConnectionの接続を行うと正常に動作しない。初回のデータベースアク

図3


ホームページ
ProjSample01
Unit1
受注日:
顧客コード:
品番:
XXXXXXXXXX
顧客名
lblCustName : TIWLabel
オブジェクトインスペクタ
edtCustCD
プロパティ
イベント
OnAsyncChange
OnAsyncClick
OnAsyncDoubleClick
OnAsyncEnter
OnAsyncExit
OnAsyncKeyDown
OnAsyncKeyPress
OnAsyncKeyUp
OnAsyncMouseDown
OnAsyncMouseMove
OnAsyncMouseOut
OnAsyncMouseOver
OnAsyncMouseUp
OnHTMLTag
OnSubmit
edtCustCDAsyncEnter
edtCustCDAsyncExit
【動作仕様】
顧客コード欄(edtCustCD)に値を入力後、フォーカスが抜ける際に、顧客マスターを検索する。
マスターに値が存在すれば、名称欄に顧客名を表示する。
マスターに値が存在しない場合、エラーメッセージを表示。

図4


顧客マスターに存在するコードを入力した場合、顧客名が表示
顧客マスターに存在しないコードを入力した場合、エラーメッセージが表示
受注日:
顧客コード:
品番:
12345678
顧客コードが正しくありません。

図5


CSV出力リンク
lnkLink : TIWLink
閉じる
取引先一覧
取引マスタよりデータを取得し、Variantに表示

セス時（初めてTSQLQuery等がオープンする時）に、接続されるようにするのがポイントとなるので注意いただきたい。

5.スマートフォン対応ページの作成方法

近年、スマートフォンがブームとなっている。今後業務アプリケーションにもスマートフォン向けサイトの対応を迫られることが予想される。

スマートフォンは標準でブラウザ機能を持っているので、これまでのWebアプリケーションをそのまま動作させることもできるが、PCと違い、スマートフォンでは解像度の制約がある。そのため、通常のサイトをそのまま一覧表示すると、図9のようにとても細かな表示となってしまう。【図9】

解決策の1つとして、スマートフォンに最適な画面設計を行えばよいわけだが、スマートフォンは機種により解像度が異なるなど、そのままではなかなか調整が難しいものがある。

この場合役に立つのが、HTMLのメタタグの1つである「ViewPort」である。このメタタグは、スマートフォンでの可視領域やズームなどの設定するものである。記述例を図10に示す。【図10】

では、このメタタグをVCL for the Webのフォームにどのように組み込めばよいのだろうか。それは、IWFormにあるExtraHeaderプロパティに記述すればよいのである。【図11】

そして、メタタグを設定したフォームを実行すると、図12のようにスマートフォンに最適化され、不要な拡大縮小を伴わない画面が構築できるのである。【図12】

実際の画面設計では項目数を制限するなど、スマートフォンならではの制約も出てくるであろうが、メタタグの可視領域を設定することで、よりスマートフォンに最適化された画面が構築できる。

6.まとめ

今回は、VCL for the Webにまつわる4つのTIPSを紹介した。Webアプリケーション開発の一助としてほしい。

また、これまでWebアプリケーションを作成したことがない方も、一度VCL for the Webによる開発を実践いただき、GUI開発と同様に生産性の高い開発が可能であることを確認いただけると幸いである。

M

図6

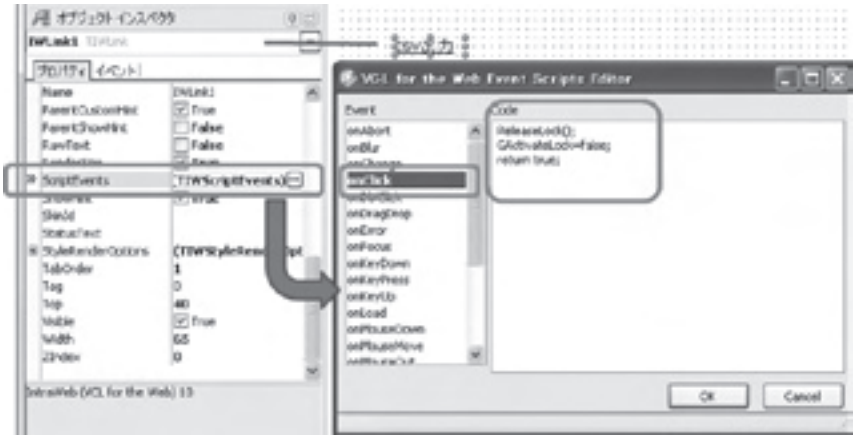
VCL for the Web Event Scripts Editor
Event
Code
OK
Cancel

図7


VCL for the Web Application Wizard
The VCL for the Web
Version 10.0.23
Welcome to the VCL for the Web Application Wizard. Select the type of application you want to create and any additional options
Application Type
StandAlone Application
Service Application
ISAPI Extension
Options
Pool Data Connections
Create User Session
(Includes DataModule)
Use ServerManager Lite
Main Form
HTML 4.0
PDA HTML 3.2
WAP
Project Name
Project1
Project Directory
C:¥TEMP¥DEMO
Ok
Cancel

図8

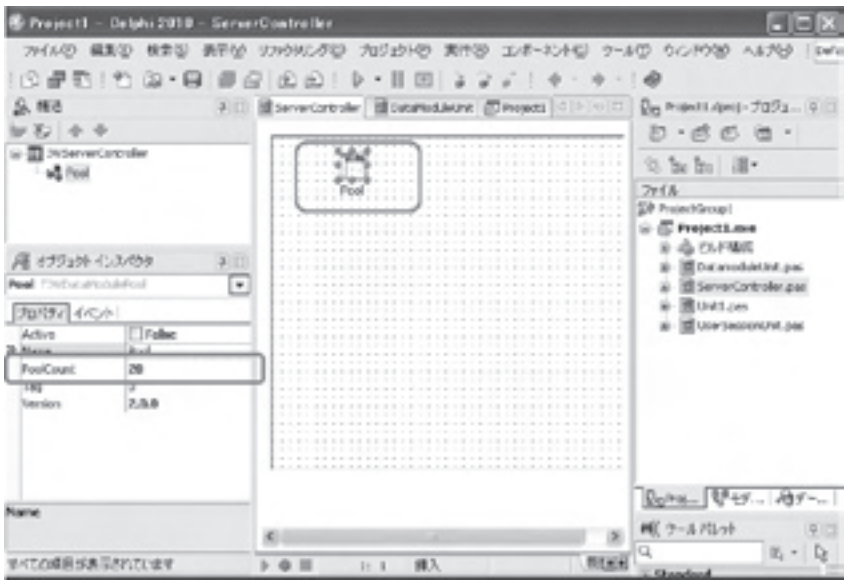

図9

図12

図10

図11

ソース1

```
{ TIWForm1 }

procedure TIWForm1.edtCustCDAsyncEnter(Sender: TObject;
  EventParams: TStringList);
begin
  // フォーカス取得時、現時点の値（初期値）を変数に保持
  FCustCode := edtCustCD.Text;
end;

procedure TIWForm1.edtCustCDAsyncExit(Sender: TObject;
  EventParams: TStringList);
begin
  // フォーカス終了時、初期値と値が変わった場合、処理を行う
  if FCustCode <> edtCustCD.Text then
  begin
    // 顧客名欄を初期化（クリア）する
    edtCustCD.BGColor := clWindow;
    lblCustName.Caption := '';

    // コード値が入力されている場合、顧客マスタ検索を行う。
    if edtCustCD.Text <> '' then
    begin
      try
        // GetCustName関数で顧客コードより顧客名を取得
        lblCustName.Caption := GetCustName(edtCustCD.Text);
      except
        // エラー（例外生成）時、エラーメッセージを表示する
        on E: Exception do
        begin
          edtCustCD.BGColor := clWebAQUA;
          lblCustName.Caption := E.Message;
        end;
      end;
    end;
  end;
end;
```

ソース2

```
procedure TIWForm2.IWLink1Click(Sender: TObject);
var
  i, j: Integer;
  sStr: String;
  sCSVList: TStringList;       // 各列毎の文字列を保持するリスト
  sCSVRowList: TStringList;    // 各行毎の情報を管理するリスト
  memCSVStream: TMemoryStream; // CSV出力用　メモリーストリーム
begin
  //CSVデータを管理する文字列リストを生成
  sCSVList := TStringList.Create;
  try
    //CSV用文字列リストの初期化
    sCSVList.Clear;
    //各列毎にある値を管理する文字列リストを生成
    sCSVRowList := TStringList.Create;
    try
      //画面グリッドよりデータ取得する
      for i := 0 to IWGrid1.RowCount - 1 do
      begin
        //列毎の値を保持するリストの初期化
        sCSVRowList.Clear;
        //各列の値をリストに追加
        for j := 0 to IWGrid1.ColumnCount - 1 do
        begin
          sStr := IWGrid1.Cell[i, j].Text;
          sCSVRowList.Add(sStr);
        end;
        //CSV用文字列リストに行情報を代入
        sCSVList.Add(sCSVRowList.CommaText);
      end;
    finally
      sCSVRowList.Free;
    end;

    //メモリーストリームの作成
    memCSVStream := TMemoryStream.Create;
    memCSVStream.Clear;
    //CSV用文字列リストをメモリーストリームに保存
    sCSVList.SaveToStream(memCSVStream);

    //メモリーストリームからのダウンロード処理
    WebApplication.SendStream(memCSVStream, True, '', 'CustomerList.csv');
  finally
    sCSVList.Free;
  end;
end;
```

ソース3

```
function TIMForm1.GetCustName(ACustCode: String): String;
var
  ADataModule: TDataModule1; //ロックを行うデータモジュール
begin
  //初期化
  Result := '';
  // データモジュールをロックする
  ADataModule := LockDataModule;
  try
    //データを抽出する
    with ADataModule.qrySQL do
    begin
      SQLConnection := ADataModule.connAS400; // ロックデータモジュールを使用
      //データ取得
      Active := False;
      ParamByName('CONOCD').AsAnsiString := ACustCode;
      //データセットを開く
      Active := True;
      try
        //対象データが存在するしない場合エラーとする
        if Eof and Bof then
          raise Exception.Create('顧客コードが正しくありません。');
        //取得した顧客名をセットする
        Result := FieldByName('CONAME').AsString;

      finally
        //データセットを閉じる
        Active := False;
      end;
    end;
  finally
    // データモジュールの使用が完了したらアンロックする
    UnlockDataModule(ADataModule);
  end;
end;
```