Delphi/400

IntraWebを使用したWeb開発のTips紹介

株式会社ミガロ.
システム事業部
システム1課
福井　和彦

略　歴
生年月日:1972年3月20日
最終学歴:1994年　大阪電気通信大学 工学部卒業
ミガロ入社年月:2001年04月　株式会社ミガロ.入社
社内経歴:2001年04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の受託開発を担当しており、
要件確認から納品・フォローに至るまで、システム開発全般に
携わっている。

株式会社ミガロ.
システム事業部
システム1課
石山　智也

略　歴
生年月日:1988年4月5日
最終学歴:2012年　近畿大学 経済学部卒業
ミガロ入社年月:2019年6月　株式会社ミガロ.入社
社内経歴:2019年6月　システム事業部配属

現在の仕事内容:
主にDelphi/400を利用したシステムの受託開発をメインに担当している。
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、テレワークの増加や業務効率化の観点から業務システムのWeb化は注目の存在となっている。
Webシステムはクライアントサーバーシステムとは違い、クライアントPCにアプリケーションをインストールや再配布を行う手間が要らず、ブラウザとインターネット環境があればどこからでもシステムを使用できることが利点である。
Delphi/400で Web サーバー アプリケーションを構築するツールとしてIntraWebがある。IntraWebを使用すると、従来のGUIアプリケーションと同様にWebサーバーアプリケーションを構築することができる。本稿ではIntraWeb 15を使用したWebアプリケーション開発のTipsを紹介する。また実行結果は、MicrosoftのEdgeを使用して確認する。

2.IntraWebでの新規プロジェクトの作成

2-1.IntraWeb 15のインストール

本題に入る前に、IntraWeb 15を使用できる開発環境を整える必要がある。Delphi/400 10.2Tokyoを標準インストールすると、IntraWebは古いバージョンのバンドル版がインストールされている。IntraWeb 15を使用するには、古いバンドル版をアンインストールした後、IntraWeb 15をインストールする必要がある。その手順を簡単にまとめておく。
＜手順＞
① IntraWeb 15を次のサイトより取得する。
https://www.atozed.com/intraweb/download/v15/
② バンドル版をアンインストールするツールを次のサイトより取得する。
https://www.atozed.com/intraweb/bundled/removal-tool/
③②で取得したツール「IWBundledRemovalTool.exe」を実行してバンドル版をアンインストールする。この時「Delete IntraWeb package･･･」のチェックボックスはチェックしておく。
④ ①で取得したIntraWeb 15をインストールする。

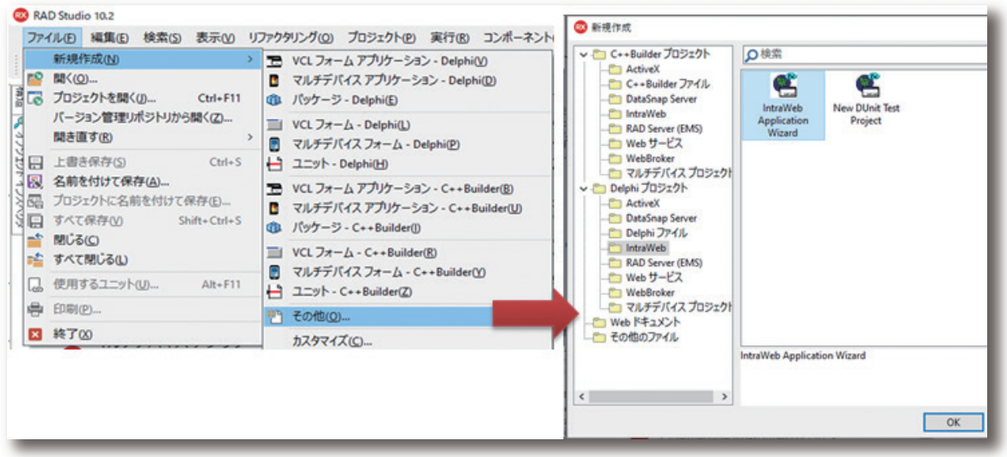
IntraWeb 15のライセンスキーを持っていない場合でも、評価版として使用することができる。ライセンスキーが必要な方は、ミガロ.営業担当まで問合せをしてほしい。

2-2.新規プロジェクトの作成

過去のレポートでもIntraWebでの新規プロジェクトの作成方法については掲載しているが、本稿では執筆時点での最新バージョンであるIntraWeb 15を例に解説するため改めて記載する。

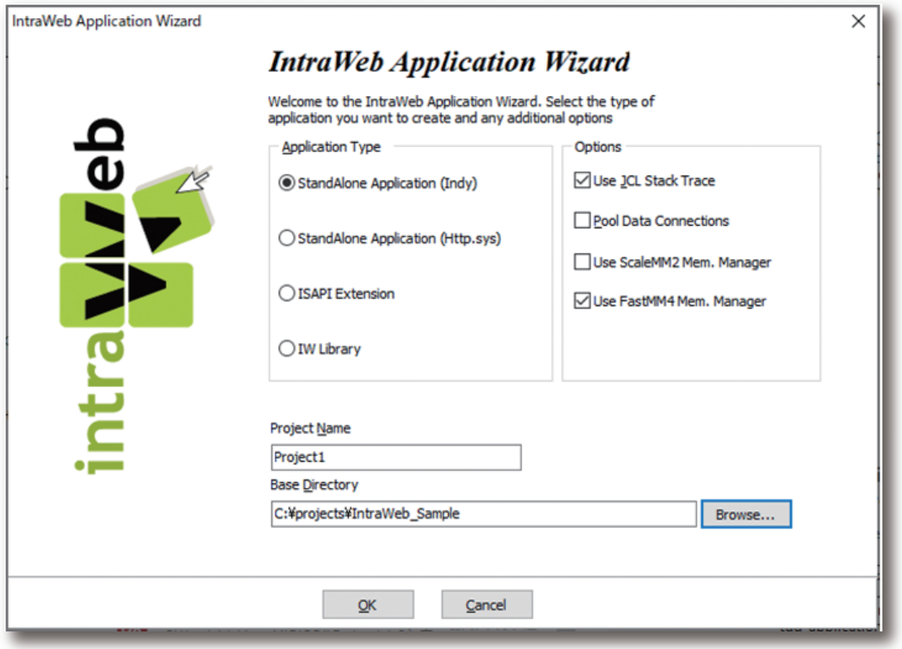
新規プロジェクトの作成は「ファイル」→「新規作成」→「その他」の順に選択し、表示された新規作成画面で「Delphi プロジェクト」→「IntraWeb」→「IntraWeb Application Wizard」をクリックする。**【図1】**

図 1　IntraWeb新規プロジェクトの作成



表示された「IntraWeb Application Wizard」でWebアプリケーションの種類、プロジェクト名、ベースディレクトリを設定する。【図2】

図 2　IntraWeb Application Wizard



【図2】にあるWebアプリケーションの種類とオプションについて簡単に記載する。

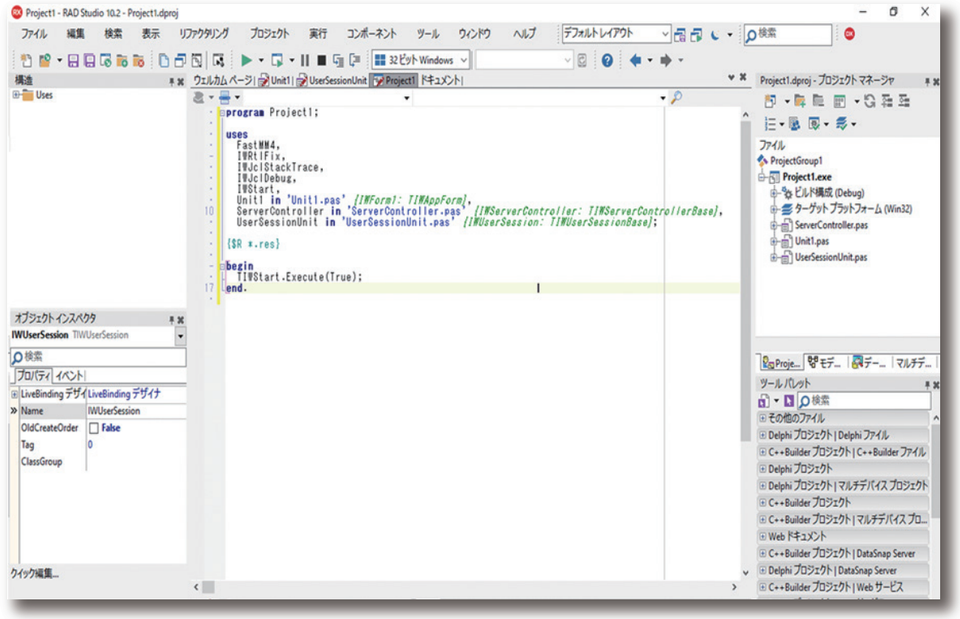
<Webアプリケーションの種類>
① StandAlone Application（Indy）、
StandAlone Application（Http.sys）（スタンドアローン）
WebサーバーもIntraWebが提供するモード。
② ISAPI Extension（アプリケーション）
WebサーバーはIISを用いて提供するモード。
③ IW Library（ASPX配置）
②と同様にWebサーバーはIISを用いるが、ASP.NETアプリケーションとして配置可能なモード。

<オプションの種類>
① Use JCL Stack Trace
JCL（JEDI Code Library）を使用してスタックトレースが可能になる。
② Pool Data Connections
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、それを使い回すことが出来るようになる。
③ Use ScaleMM2 Mem.Manager、
Use FastMM4 Mem.Manager
メモリリークを検出するためにScaleMM2、FastMM4を使用することが可能になる。

【図2】の状態でOKボタンを押下すると新規プロジェクトが作成される。【図3】
開発画面には、ServerController、Unit、UserSessionUnitの3つが自動で生成される。
ServerControllerは、サーバー側の処理を扱うファイルで、例えばブラウザの戻るボタン押下時やアプリケーション終了時の制御を行うといったことが出来る。

Unitは実際に開発を行う画面で、このファイルに対してコンポーネントを貼り付けて開発を行う。
UserSessionUnitはクライアントサーバー開発を行う時のDataModuleと同じ扱いになる。
次の章では明細形式のWeb画面を作成するテクニックを紹介する。

図 3　IntraWeb 新規プロジェクト



3.明細形式のWeb画面を作成するテクニック

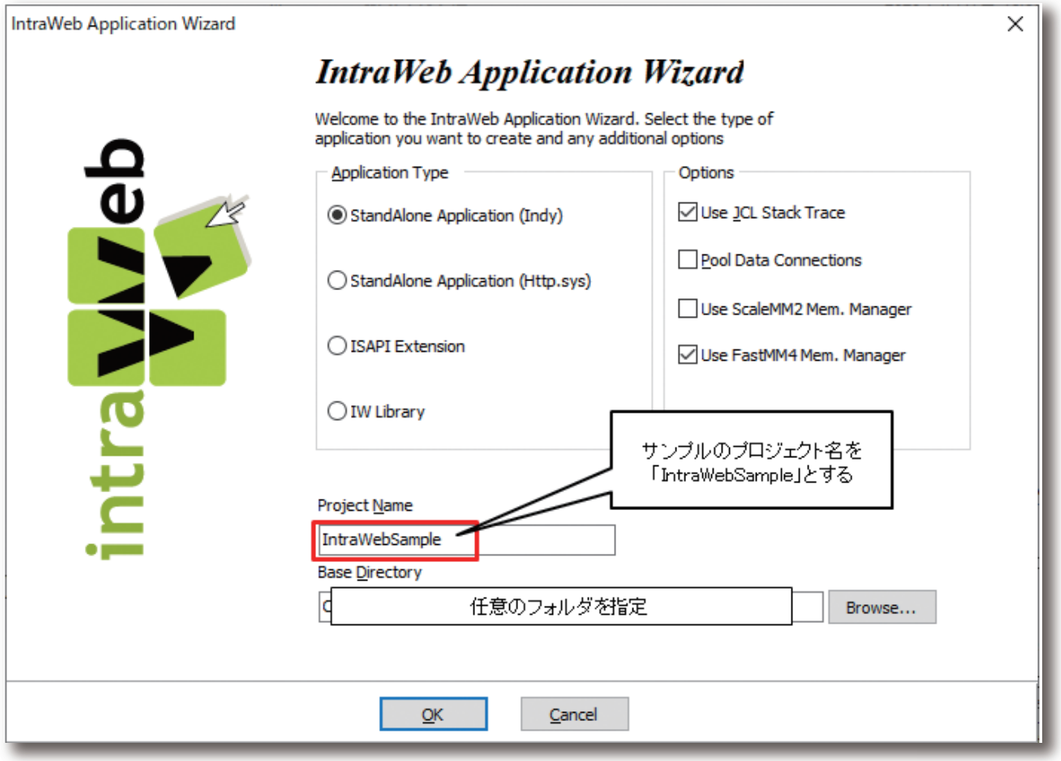
この章では明細形式のWeb画面でよく見られる「件数が一定数を超えた際に頁を分ける機能」や、「明細のタイトルをクリックしたら並び替えが行える機能」の実装方法について、次の順番に従って説明を行う。

① 「TIWGrid」を使用した明細表示
② 明細表示のカスタマイズテクニック
- 件数が一定数を超えた際に頁を分ける機能を実装
- 明細のタイトルをクリックしたら並び替えが行える機能を実装

3-1.「TIWGrid」を使用した明細表示

まず始めに本章以降で使用するプロジェクトを、プロジェクト名「IntraWebSample」として新規作成する。**【図4】**

図 4　本稿で使用する新規プロジェクトの作成



作成されたプロジェクトのUnit1を開き、表示されたフォームのNameプロパティに「frmList」と設定する。そして、**【図5】**に従ってコンポーネントを配置し各プロパティを設定する。本稿では明細表示のコンポーネントとしてTIWGridを使用する。配置と設定が終わったら、Unit1をファイル名「ListFrm.pas」として名前を付けて保存を行う。

それでは、実行してブラウザにどのように表示されるか確認をしてみよう。実行方法は、メニューバーより「実行」→「実行」と選択するか、キーボードの「F9」を押下する。初回実行時、Windowsの警告が出る場合がある。
【図6】　警告が出たら「アクセスを許可する」をクリックして先に進める。

図 5　「明細形式の表示」の画面設計

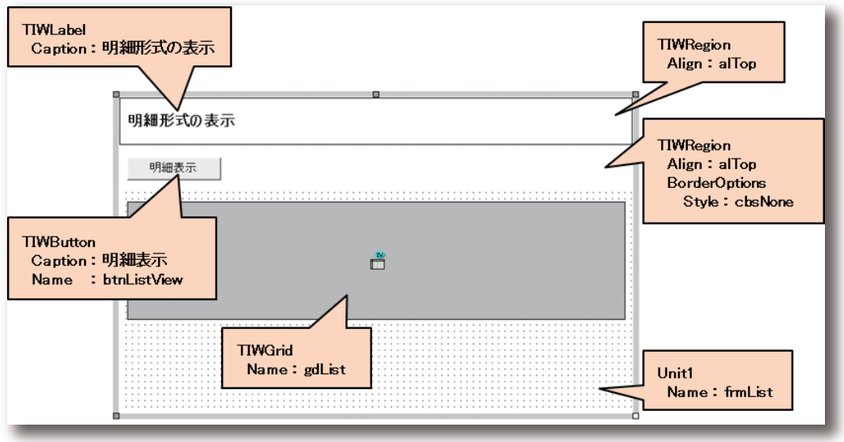
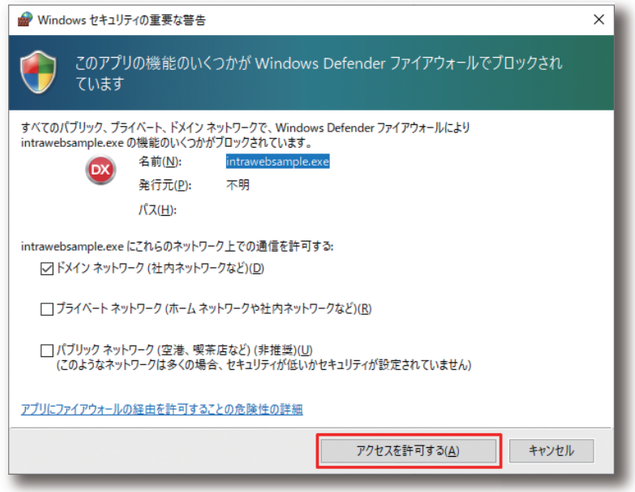
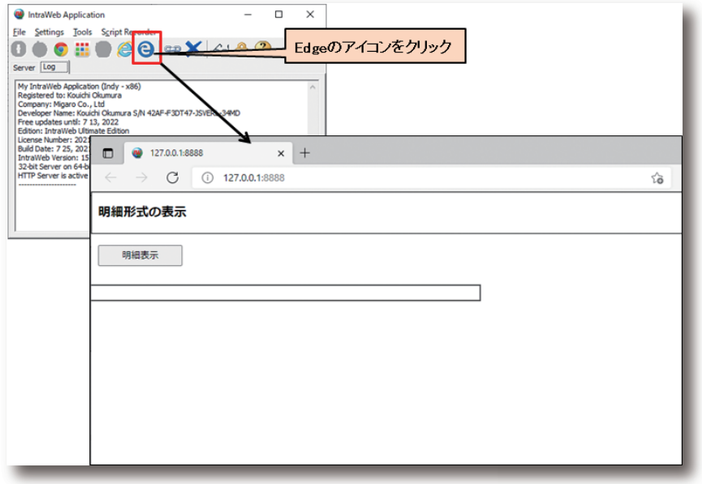


図 6　Windowsの警告画面



実行すると始めに「IntraWeb Application」の画面が開く。画面には各ブラウザのアイコンが並んでいる。その中から Edgeのアイコンをクリックする。そうするとEdgeが開きWeb画面が表示される。**【図7】**

図 7　実行結果イメージ

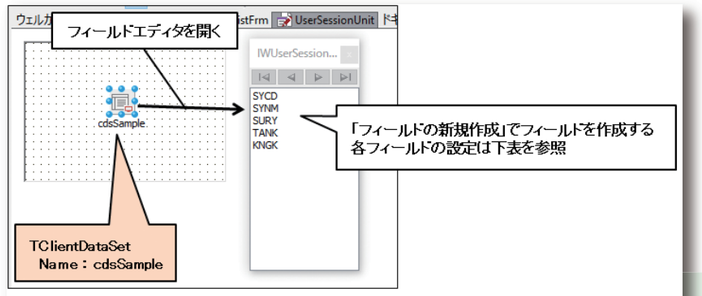


この時点では明細にはまだ何も表示はされない。実行結果が確認できたら、Edgeの「×」ボタンをクリックしてブラウザを閉じ、IntraWeb Application画面も同様に「×」ボタンをクリックして終了し開発画面へ戻る。

続いて、明細に表示するデータの準備を行う。データはプログラムでTClientDatasetへ作成して保持する。

UserSessionUnitを開き、画面にTClientDatasetを配置する。Nameプロパティは「cdsSample」と設定する。続いて**【図8】**に従ってTClientDatasetの各設定を行う。

図 8　TClientDatasetの設定



FieldName	Name	DisplayLabel	データ型	Size	FieldKind
SYCD	cdsSampleSYCD	商品コード	String	6	fkData
SYNM	cdsSampleSYNM	商品名	String	42	fkData
SURY	cdsSampleSURY	数量	Integer		fkData
TANK	cdsSampleTANK	単価	Integer		fkData
KNGK	cdsSampleKNGK	金額	Currency		fkData

次に、UserSessionUnitのOnCreateイベントでcdsSampleのCreateDataSetメソッドを呼び出し、メモリ上にデータセットを作成する。**【ソース1】**

ソース 1

UserSessionUnitのOnCreateイベント

```
procedure TIWUserSession.IWUserSessionBaseCreate(Sender: TObject);
begin
  // データセットを作成
  cdsSample.CreateDataSet;
end;
```

そして、データ作成の処理としてUserSessionUnitのPublicのプロシージャ「SampleDataCreate」を作成する。実装部は**【ソース2】**に従って記述する。この処理で100件のデータを作成する。UserSessionUnitの開発は以上となる。

ソース 2

UserSessionUnitのSampleDataCreateプロシージャ

```
<宣言部>
 public
    [ Public declarations ]
    procedure SampleDataCreate;  // サンプルデータ作成
  end;
<実装部>
procedure TIWUserSession.SampleDataCreate;
var
  i: Integer;
begin
  // cdsSampleをクリア
  cdsSample.EmptyDataSet;

  for i := 1 to 100 do
  begin
    cdsSample.Append;
    cdsSampleSYCD.AsString   := FormatFloat('000000', i * 10);
    cdsSampleSYNM.AsString   := 'サンプル商品 (' + FormatFloat('000000', i * 10) + ') ';
    cdsSampleSURY.AsInteger  := i;
    cdsSampleTANK.AsInteger  := i * 1000;
    cdsSampleKNGK.AsCurrency := i * (i * 1000);
    cdsSample.Post;
  end;
end;
```

ここからは、frmListの明細へcdsSampleのデータを表示する処理を実装していく。frmListを開き、メニューバーより「ファイル」→「使用するユニット」を選び、ServerControllerを選択する。次にfrmListのPrivateプロシージャとして「GridClear」「GridLayout」「ListDataSet」の3つを宣言する。**【ソース3】**各プロシージャの実装部は**【ソース4～7】**に従って記述する。

ソース 3

frmListのPrivateプロシージャ宣言

```
private
    [ Private 宣言 ]
    procedure GridClear;        // 明細初期化
    procedure GridLayout;       // 明細の項目幅調整
    procedure ListDataSet;      // 明細データセット
```

ソース 4

frmListのGridClearプロシージャ

```
procedure TfrmList.GridClear;
begin
  with gdList do
  begin
    Clear;

    // 行数初期化
    RowCount := 1;

    // タイトル設定
    Cell[0, 0].Text := '商品コード';
    Cell[0, 1].Text := '商品名';
    Cell[0, 2].Text := '数量';
    Cell[0, 3].Text := '単価';
    Cell[0, 4].Text := '金額';

    // 列表題Alignment
    Cell[0, 0].Alignment := taCenter;
    Cell[0, 1].Alignment := taCenter;
    Cell[0, 2].Alignment := taCenter;
    Cell[0, 3].Alignment := taCenter;
    Cell[0, 4].Alignment := taCenter;
  end;
end;
```

ソース 5

frmListのGridLayoutプロシージャ

```
procedure TfrmList.GridLayout;
begin
  with gdList do
  begin
    // 列幅の初期設定
    Cell[0, 0].Width := '100';  // [00] 商品コード
    Cell[0, 1].Width := '300';  // [01] 商品名
    Cell[0, 2].Width := '100';  // [02] 数量
    Cell[0, 3].Width := '170';  // [03] 単価
    Cell[0, 4].Width := '180';  // [04] 金額

    // 高さの初期設定
    Cell[0, 0].Height := '35';
  end;
end;
```

ソース 6

frmListのListDataSetプロシージャ

```
procedure TfrmList.ListDataSet;
var
  i: Integer;
begin
  // 明細
  with gdList do
  begin
    // サンプルデータを参照
    UserSession.cdsSample.First;

    while not UserSession.cdsSample.Eof do
    begin
      RowCount := RowCount + 1;
      i := RowCount - 1;

      // Alignmentの設定
      Cell[i, 0].Alignment := taCenter;
      Cell[i, 1].Alignment := taLeftJustify;
      Cell[i, 2].Alignment := taRightJustify;
      Cell[i, 3].Alignment := taRightJustify;
      Cell[i, 4].Alignment := taRightJustify;

      【ソース7】へ続く
```

ServerControllerで宣言されている
UserSessionメソッドを使用して
UserSessionUnitのcdsSampleを参照する

ソース 7

frmListのListDataSetプロシージャ（続き）

```
      // サンプルデータ
      Cell[i, 0].Text := UserSession.cdsSampleSYCD.AsString;                    // 商品コード
      Cell[i, 1].Text := UserSession.cdsSampleSYNM.AsString;                    // 商品名
      Cell[i, 2].Text := FormatFloat('#,0', UserSession.cdsSampleSURY.AsInteger);  // 数量
      Cell[i, 3].Text := FormatFloat('#,0', UserSession.cdsSampleTANK.AsInteger);  // 単価
      Cell[i, 4].Text := FormatFloat('#,0', UserSession.cdsSampleKNGK.AsCurrency); // 金額

      UserSession.cdsSample.Next;
    end;
  end;

  // 明細の項目幅調整
  GridLayout;
end;
```

続いてfrmListのOnCreateイベントに、**【ソース8】**に従って明細の初期表示の処理を実装する。そして「明細表示」ボタン(btnListView)のOnClickイベントには、**【ソース9】**のようにここまで設定してきた各プロシージャの実行を記述する。

ソース 8

frmListのOnCreateイベント

```
procedure TfrmList.IWAppFormCreate(Sender: TObject);
begin
  // 明細初期設定
  gdList.RowCount    := 1;
  gdList.ColumnCount := 5;

  // 明細初期化
  GridClear;

  // 明細の項目幅調整
  GridLayout;
end;
```

【ソース4】参照

【ソース5】参照

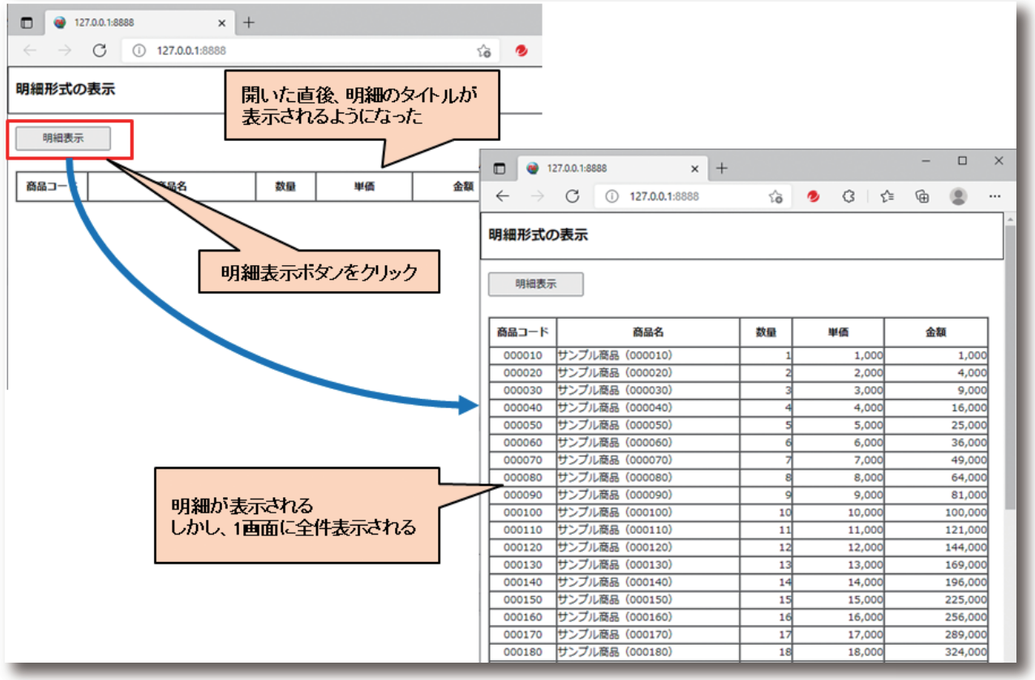
ソース 9

frmList.btnListViewのOnClickイベント

```
procedure TfrmList.btnListViewClick(Sender: TObject);
begin
  // 明細初期化
  GridClear;
  // サンプルデータ作成
  UserSession.SampleDataCreate;
  // 明細データセット
  ListDataSet;
end;
```

それでは実行して確認してみよう。今回は【図7】の時とは違い明細のタイトルが表示されている。続いて画面の「明細表示」ボタンをクリックすると、明細にデータが表示される。しかし現時点では、1画面に全件を明細表示しただけとなる。**【図9】**

図 9　実行結果イメージ(明細表示)



3-2.明細表示のカスタマイズテクニック

この節では前節で作成した明細表示の画面に対して、機能追加の実装を進めていく。実装には外部ライブラリを使用する。IntraWebにおいても、画面をカスタマイズするために外部ライブラリを使用することが可能である。前節で画面に配置したTIWGridは、HTMLにはtableタグとして出力される。そのtableタグの表示内容をカスタマイズできる、「DataTables」という外部のjQueryプラグインライブラリを使用して機能追加の実装を行う。

（公式サイト：https://www.datatables.net/）

まず始めにfrmListのOnCreateイベントへ**【ソース10】**に従って追記を行う。ここではCDN（Content Delivery Network）にて公開されている、jquery DataTablesのスタイルシートとJavascriptの参照設定を追記する。参照先はURLで記述しており、その中にバージョン情報も含まれている。最新のバージョンについては、https://cdn.datatables.netを参照いただきたい。

続いてfrmListのOnRenderイベントへスクリプトの追加を記述する。**【ソース11】**

記述した中に有る”#TBLGDLIST”の”GDLIST”は、画面のTIWGridのNameプロパティの設定値を大文字にしたものとなる。もし**【図5】**の設定時にTIWGridのNameプロパティを異なる設定にしている場合は、その設定に合わせていただきたい。

ソース 10

frmListのOnCreateイベント（追記）

```
procedure TfrmList.IWAppFormCreate(Sender: TObject);
begin
  // jQuery DataTables の参照設定
  ContentFiles.Add('https://cdn.datatables.net/1.11.0/css/jquery.dataTables.min.css');
  ContentFiles.Add('https://cdn.datatables.net/1.11.0/js/jquery.dataTables.min.js');

  // 明細初期設定
  gdList.RowCount    := 1;
  gdList.ColumnCount := 5;

  // 明細初期化
  GridClear;

  // 明細の項目幅調整
  GridLayout;
end;
```

追記する

最新のバージョンは以下のサイトで確認
https://cdn.datatables.net

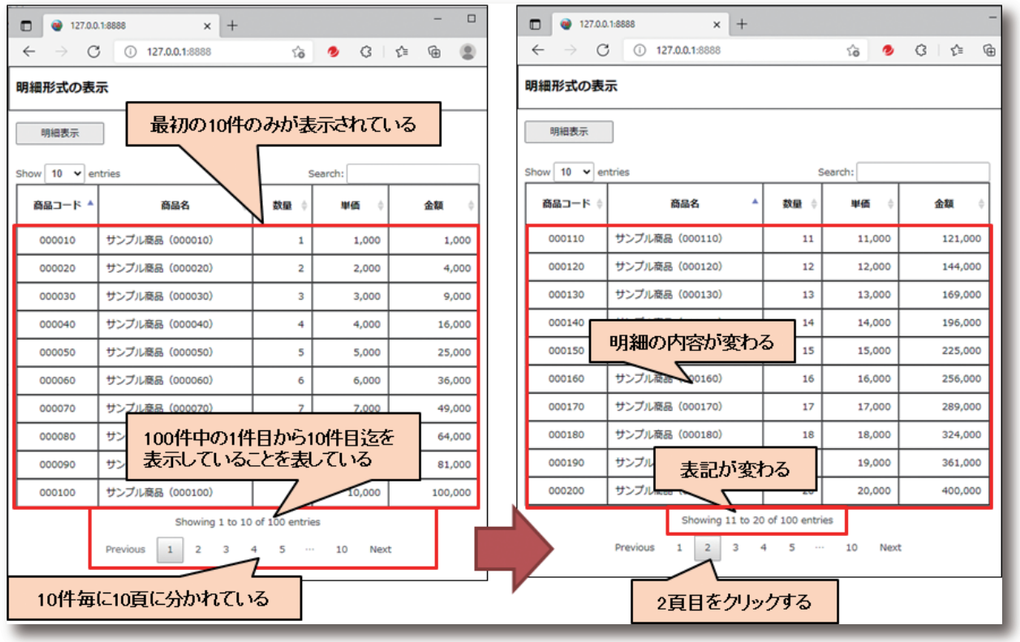
ソース 11

frmListのOnRenderイベント

```
procedure TfrmList.IWAppFormRender(Sender: TObject);
begin
  // HTMLスクリプトを追加
  AddToInitProc('$("#TBLGDLIST").DataTable();');
end;
```

それでは実行して確認してみよう。Web画面が開くと**【図9】**の時点から更に、いくつか項目や表記が追加されており、明細のタイトル部にも▲▼が表示されている。ただ全て英語表記となっている。明細を表示して追加された各機能を見ていこう。明細を表示すると最初の10件のみが明細に表示される。明細の下には「Showing 1 to 10 of 100 entries」と表記されており、100件中の1件目から10件目迄を表示していることを表している。更にその下には、「Previous」「1・・・10」「Next」が表示されている。これは、10件毎に10頁に分かれている各頁間を移動する機能になっている。それぞれクリックすると明細の内容が変わっているのと、「Showing（X）to（Y）of 100 entries」の（X）と（Y）の値が変わっているのが分かる。これで件数が一定数を超えた際に頁を分ける機能が実装されたことになる。**【図10】**

図 10　件数が一定数を超えた際に頁を分ける機能



次に明細のタイトルをクリックしてみると、クリックした項目の昇順で明細が並び変わる。しかも頁内ではなく、100件全体での並び替えとなっている。更に同じ項目をクリックすると今度は降順に切り替わる。これで明細のタイトルをクリックしたら並び替えが行える機能も実装されたことになる。**【図11】**

明細の上に追加された項目についても見てみる。左上のコンボBOX「Show」は1頁当りの件数を示している。初期値は「10」でコンボBOXリストには既定の値がセットされている。コンボBOXの値を変える毎に、選択した値の件数が1頁当りの件数となって、明細の表示内容が変わる。

また右上の「Search」は検索項目になる。Search項目に”12”と入力してみると、”1”を入力した時点で検索が行われ、”2”を入力すると更に検索が行われるのが分かる。しかも、この検索は、全件（100件）かつ全項目を対象に曖昧検索（入力した文字列を含む検索）を行っている。**【図12】**

図 11　明細の並び替え機能

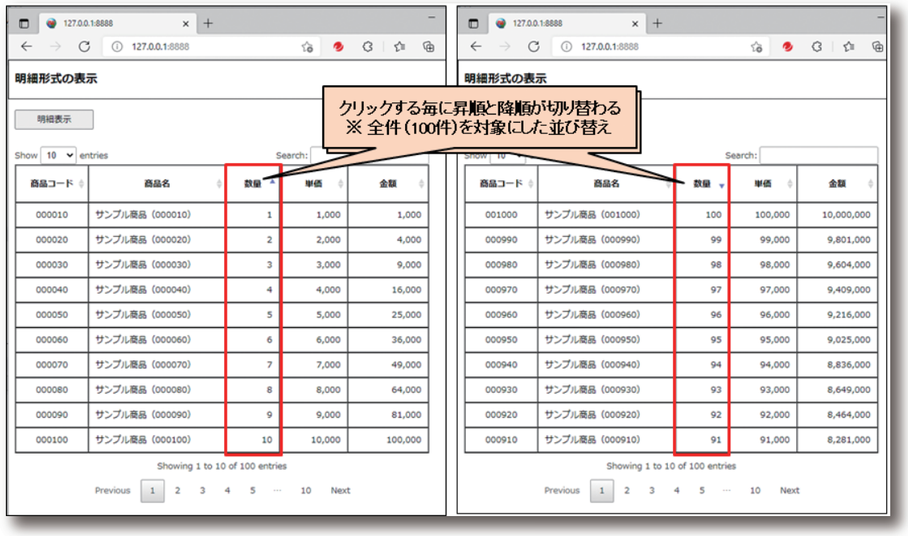
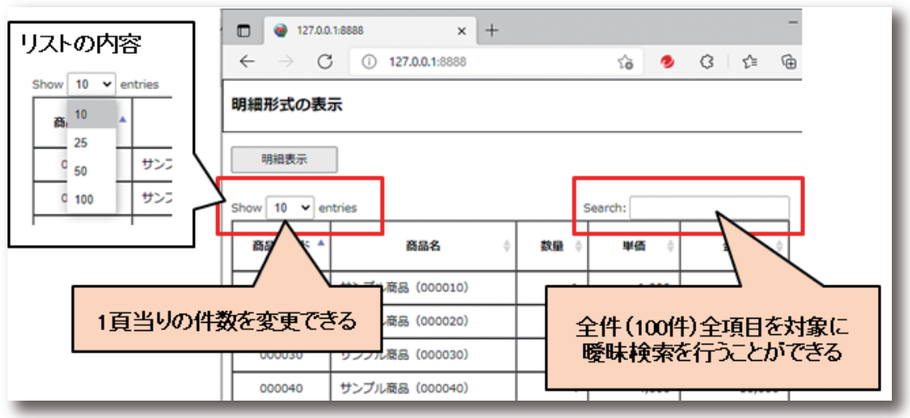


図 12　追加機能「Show」「Search」



ここまでの機能を僅か3行のソースコードを追加するだけで実現することができる。また、ShowコンボBOXやSearch項目の非表示や、1頁当りの件数の初期値を変更することも可能である。frmListのOnRenderイベントのソースコードを変更することで実装できる。詳しくは**【ソース12】**を参照し、実装して結果を確認してもらいたい。

この節の最後に、追加された表記が英語になっているものを日本語に変更する方法を説明する。変更箇所はfrmListのOnRenderイベントで、ソースコードを**【ソース13】**に従って変更することで日本語表記に変わる。ここでもCDNにて公開されているJSON（JavaScript Object Notation）を参照する。

ソース 12

frmListのOnRenderイベント（変更）

```
procedure TfrmList.IWAppFormRender(Sender: TObject);
var
  sScriptStr: string;
begin
  // HTMLスクリプトを追加
//  AddToInitProc('$("#TBLGDLIST").DataTable();');

  sScriptStr := '$("#TBLGDLIST").DataTable({';
  sScriptStr := sScriptStr + 'displayLength: 20';
  sScriptStr := sScriptStr + ', lengthChange: false ';
  sScriptStr := sScriptStr + ', searching: false ';
  sScriptStr := sScriptStr + '});';

  //HTMLスクリプトを追加
  AddToInitProc(sScriptStr);
end;
```

「displayLength: 20」
1頁当りの件数の初期値を設定

「lengthChange: false」
ShowコンボBOXを非表示にする

「searching: false」
Search項目を非表示にする

ソース 13

frmListのOnRenderイベント（日本語対応）

```
procedure TfrmList.IWAppFormRender(Sender: TObject);
var
  sScriptStr: string;
begin
  // HTMLスクリプトを追加
//  AddToInitProc('$("#TBLGDLIST").DataTable();');

  sScriptStr := '$.extend( $.fn.dataTable.defaults, {';
  sScriptStr := sScriptStr + 'language: {';
  sScriptStr := sScriptStr
                + 'url: "http://cdn.datatables.net/plug-ins/9dcbecd42ad/i18n/Japanese.json"';
  sScriptStr := sScriptStr + '}';
  sScriptStr := sScriptStr + '});';

  sScriptStr := sScriptStr + '$("#TBLGDLIST").DataTable();';

  //HTMLスクリプトを追加
  AddToInitProc(sScriptStr);
end;
```

実行して確認してみよう。【図10】では英語表記になっていた箇所が日本語表記に変わっている。【図13】

以上で「明細形式のWeb画面を作成するテクニック」の説明は終了となる。次の章ではCSVファイルを取扱うテクニックを紹介する。

図 13　日本語表記対応

4. CSVファイルのアップロード／取込を行うテクニック

この章では「CSVファイルをアップロード」し「アップロードしたCSVファイルのデータを取り込み画面明細へ表示する」方法について説明を行う。

4-1.CSVファイルのアップロード

最初はCSVファイルをアップロードする機能を実装していく。新しくフォームを追加して実装するため、メニューバーより「ファイル」→「新規作成」→「その他」の順に選択し、表示された新規作成画面で「Delphiプロジェクト」→「IntraWeb」→「New Form」をクリックする。【図14】

新しく追加された「Unit1」のNameプロパティを「frmCSVUpload」と設定する。【図15】に従ってコンポーネントを配置し各プロパティを設定する。この時「TIWFileUploader」のTextStringsプロパティの設定も合わせて行う。【図16】

この画面は「TIWFileUploader」を使用してアップロードしたCSVファイルを読み込み、そのデータを「TIWGrid」に表示をする。そして「更新」ボタンをクリックしたら、前章で設定したUserSessionUnitのcdsSampleへ更新を行い、当画面を閉じてfrmListの明細へ反映させる動きとなる。

配置と設定が終わったら、Unit1をファイル名「CSVUploadFrm.pas」として「名前を付けて保存」を行う。

図 14　「CSVアップロード用画面」の追加

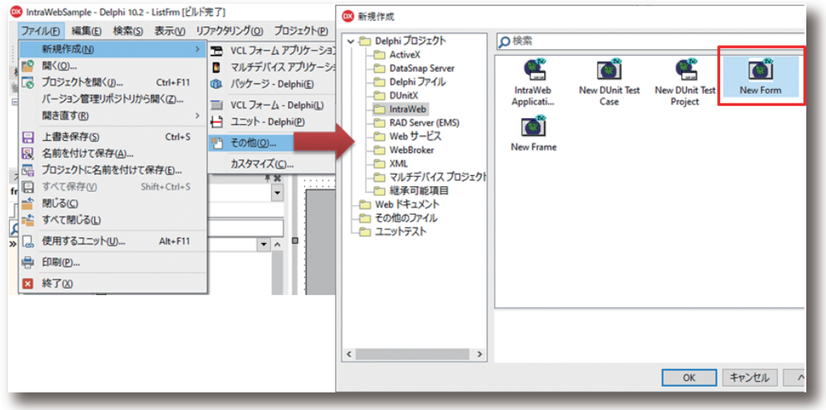


図 15　「CSVアップロード用画面」の画面設計

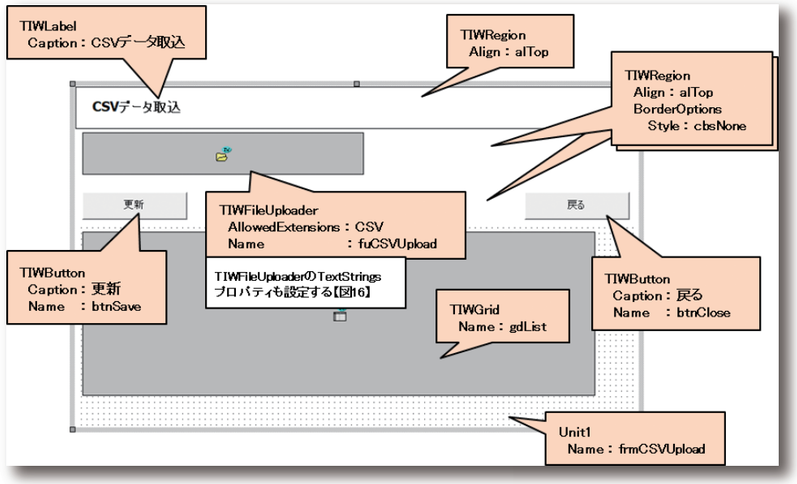
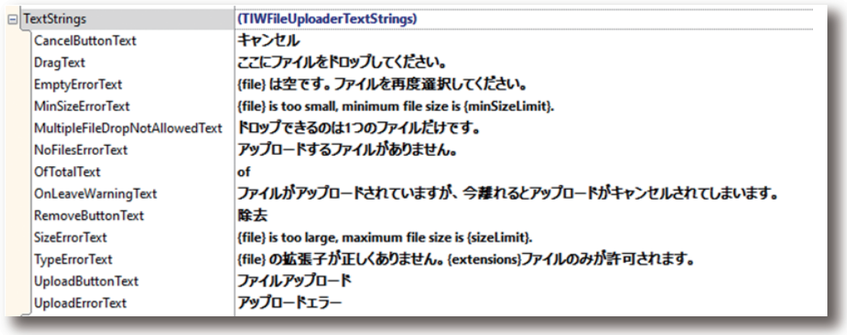


図 16　TIWFileUploaderのTextStringsプロパティの設定



TextStrings	(TIWFileUploaderTextStrings)
CancelButtonText	キャンセル
DragText	ここにファイルをドロップしてください。
EmptyErrorText	{file} は空です。ファイルを再度選択してください。
MinSizeErrorText	{file} is too small, minimum file size is {minSizeLimit}.
MultipleFileDropNotAllowedText	ドロップできるのは1つのファイルだけです。
NoFilesErrorText	アップロードするファイルがありません。
OfTotalText	of
OnLeaveWarningText	ファイルがアップロードされていますが、今離れるとアップロードがキャンセルされてしまいます。
RemoveButtonText	除去
SizeErrorText	{file} is too large, maximum file size is {sizeLimit}.
TypeErrorText	{file} の拡張子が正しくありません。{extensions}ファイルのみが許可されます。
UploadButtonText	ファイルアップロード
UploadErrorText	アップロードエラー

始めに、「TIWFileUploader」を使用してアップロードしたCSVファイルを読み込み、そのデータを「TIWGrid」に表示するところまでを実装していく。前章でfrmListに実装したのと同様に、frmCSVUploadのPrivateプロシージャとして「GridClear」「GridLayout」の2つを宣言する。合わせてプライベート変数「FUploadFile」もString型で宣言しておく。

【ソース14】 各プロシージャの実装部については、GridClearは**【ソース4】**、GridLayoutは**【ソース5】**と同様の処理を実装する。

ソース 14

frmCSVUploadのPrivateプロシージャと変数の宣言

```
private
    { Private 宣言 }
    FUploadFile: String;        // CSVファイル保管場所

    procedure GridClear;        // 明細初期化
    procedure GridLayout;       // 明細の項目幅調整
```

そしてfrmCSVUploadのOnCreateイベントにも、前章でfrmListに実装したのと同様に明細の初期表示の処理を実装する。**【ソース8】**

続いて、TIWFileUploaderの各イベントに処理を実装していく。まずはOnAsyncUploadCompletedイベントに**【ソース15】**に従って処理を実装する。ここでは、CSVファイルのアップロードが終わった時点で、アップロードしたCSVファイルのフルパスをFUploadFileに退避する。

次にOnAsyncUploadSuccessイベントに**【ソース16～17】**に従って処理を実装する。ここでは、OnAsyncUploadCompletedイベントで取得したCSVファイルのフルパス：FUploadFileを使用して、アップロードしたCSVファイルを開き、画面明細へ展開する処理となっている。

ソース 15

frmCSVUpload.fuCSVUploadのOnAsyncUploadCompletedイベント

```
procedure TfrmCSVUpload.fuCSVUploadAsyncUploadCompleted(Sender: TObject;
  var DestPath, FileName: string; var SaveFile, Overwrite: Boolean);
begin
  //アップロードしたファイルのパスとファイル名を取得
  FUploadFile := DestPath + FileName;
end;
```

ソース 16

frmCSVUpload.fuCSVUploadのOnAsyncUploadSuccessイベント

```
procedure TfrmCSVUpload.fuCSVUploadAsyncUploadSuccess(Sender: TObject;
  EventParams: TStringList);
var
  i: Integer;
  sStrList: TStringList;
  sRowList: TStringList;
begin
  // 明細初期化
  GridClear;

  //CSVデータを保持するリストの作成
  sStrList := TStringList.Create;
  try
    //行データを保持するリストの作成
    sRowList := TStringList.Create;
    try
      sStrList.LoadFromFile(FUploadFile);
      //先頭行は、タイトルの為無視して、2行目から取得
      for i := 1 to sStrList.Count - 1 do
      begin
        //行のテキスト（カンマ区切り文字列）をsRowListにセット
        sRowList.Clear;
        sRowList.CommaText := sStrList[i];
        //sRowListの要素より明細へ値をセット
        gdList.RowCount := gdList.RowCount + 1;
```

【ソース17】へ続く

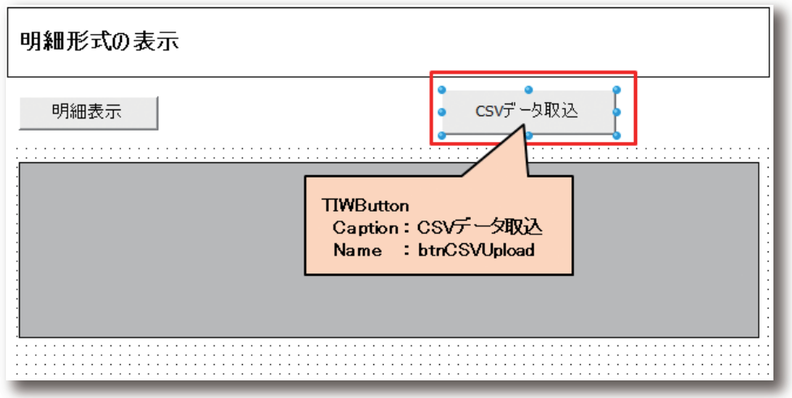
ソース 17

frmCSVUpload.fuCSVUploadのOnAsyncUploadSuccessイベント（続き）

```
        // Alignmentの設定
        gdList.Cell[gdList.RowCount - 1, 0].Alignment := taCenter;
        gdList.Cell[gdList.RowCount - 1, 1].Alignment := taLeftJustify;
        gdList.Cell[gdList.RowCount - 1, 2].Alignment := taRightJustify;
        gdList.Cell[gdList.RowCount - 1, 3].Alignment := taRightJustify;
        gdList.Cell[gdList.RowCount - 1, 4].Alignment := taRightJustify;

        // サンプルデータ
        gdList.Cell[gdList.RowCount - 1, 0].Text := sRowList[0];
        gdList.Cell[gdList.RowCount - 1, 1].Text := sRowList[1];
        gdList.Cell[gdList.RowCount - 1, 2].Text := FormatFloat('#,0', StrToIntDef(sRowList[2], 0));
        gdList.Cell[gdList.RowCount - 1, 3].Text := FormatFloat('#,0', StrToIntDef(sRowList[3], 0));
        gdList.Cell[gdList.RowCount - 1, 4].Text := FormatFloat('#,0', StrToCurrDef(sRowList[4], 0));

      end;
    finally
      sRowList.Free;
    end;
  finally
    sStrList.Free;
  end;

  // 明細の項目幅調整
  GridLayout;
end;
```

ここまで実装できたら、frmListからfrmCSVUploadを呼び出すように、frmListへ処理を追加する。frmListを開き【図17】に従ってTIWButtonを追加する。そして追加した「CSVデータ取込」ボタン（btnCSVUpload）のOnClickイベントには【ソース18】に従って処理を実装する。合わせてfrmListのuses節に「CSVUploadFrm」を追加し、プライベート変数「frmCSVUpload」の宣言を行う。

図 17　frmListに「CSVデータ取込ボタン」を追加



ソース 18

frmListのbtnCSVUploadのOnClickイベント

＜usesに「CSVUploadFrm」を追加＞

```
uses
  Classes, SysUtils, IWAppForm, IWApplication, IWColor, IWTypes,
  IWVCLBaseControl, IWBaseControl, IWBaseHTMLControl, IWControl, IWCompLabel,
  Vcl.Controls, Vcl.Forms, IWVCLBaseContainer, IWContainer, IWHTMLContainer,
  IWHTML40Container, IWRegion, IWCompGrids, IWCompButton, CSVUploadFrm;
```

＜private変数を宣言＞

```
 private
    { Private 宣言 }
    frmCSVUpload: TfrmCSVUpload;

    procedure GridClear;        // 明細初期化
    procedure GridLayout;       // 明細の項目幅調整
    procedure ListDataSet;      // 明細データセット
```

＜OnClickイベント＞

```
procedure TfrmList.btnCSVUploadClick(Sender: TObject);
begin
  frmCSVUpload := TfrmCSVUpload.Create(WebApplication);
  frmCSVUpload.Show;
end;
```

実行する前にアップロード用のCSVファイルを準備しておく。CSVファイルのデータイメージは【図18】を参照いただきたい。それでは実行して確認してみよう。今節で追加したfrmListの「CSVデータ取込」ボタンをクリックすると、frmCSVUploadへ遷移する。画面上部には背景色が赤色の「ファイルアップロード」ボタンが表示されている。CSVファイルをアップロードする方法は以下の2通りがある。【図19】

①「ファイルアップロード」ボタンをクリックして、ファイル選択。ダイアログを開き、ダイアログよりCSVファイルを選択する。

②CSVファイルを直接「ファイルアップロード」ボタンへドラッグ＆ドロップする。

どちらかの方法でCSVファイルのアップロードを行う。そうすると画面の明細にデータが表示され、アップロードしたCSVファイルのデータが確認できる。【図20】

では、CSVファイル以外をアップロードしたらどうなるのか確認してみよう。PDFファイルを用意しアップロードを行うと、【図21】のようにエラーが発生する。これは、【図15】でTIWFileUploaderのAllowedExtensionsプロパティに”CSV”を設定したことで、拡張子を制限しているためである。

TIWFileUploaderは先程の2通りのアップロードの方法を既に実装しており、またWebサーバーで実行した場合には、アップロードしたファイルをWebサーバー内のフォルダに保管する機能も実装している。そのためTIWFileUploaderを使用することで、Webアプリケーションでのファイルアップロード機能を簡単に実装することが可能となる。

図 18　CSVファイルのデータイメージ

```
1  商品コード,商品名,数量,単価,金額
2  "100010","サンプル商品（100010）",90,1000,90000
3  "100020","サンプル商品（100020）",91,1000,91000
4  "100030","サンプル商品（100030）",92,1000,92000
5  "100040","サンプル商品（100040）",93,1000,93000
6  "100050","サンプル商品（100050）",94,1000,94000
7  "100060","サンプル商品（100060）",95,1000,95000
8  "100070","サンプル商品（100070）",96,1000,96000
9  "100080","サンプル商品（100080）",97,1000,97000
10 "100090","サンプル商品（100090）",98,1000,98000
11 "100100","サンプル商品（100100）",99,1000,99000
12 "100110","サンプル商品（100110）",80,1000,80000
13 "100120","サンプル商品（100120）",81,1000,81000
14 "100130","サンプル商品（100130）",82,1000,82000
15 "100140","サンプル商品（100140）",83,1000,83000
16 "100150","サンプル商品（100150）",84000
```

図 19 実行結果イメージ（CSVアップロード）

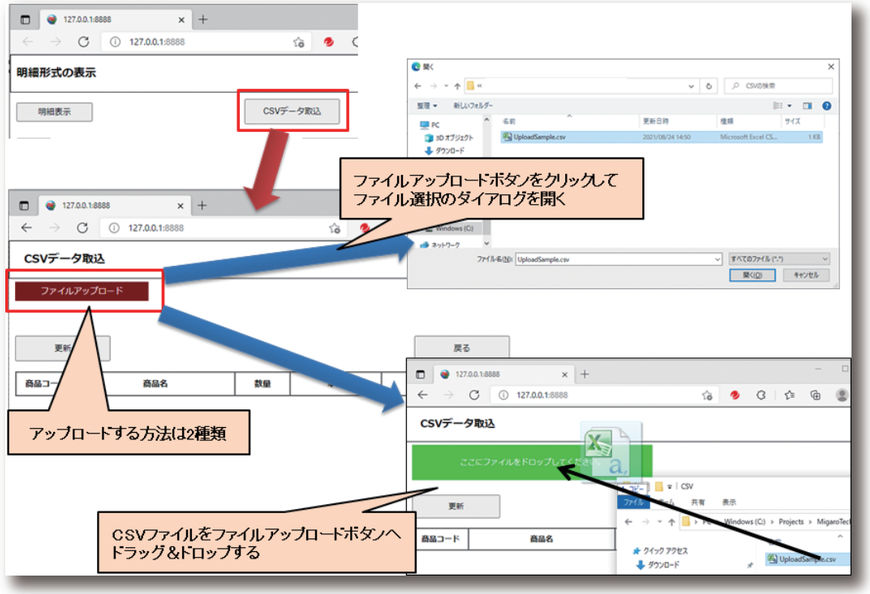


図 20 実行結果イメージ（CSVアップロード）

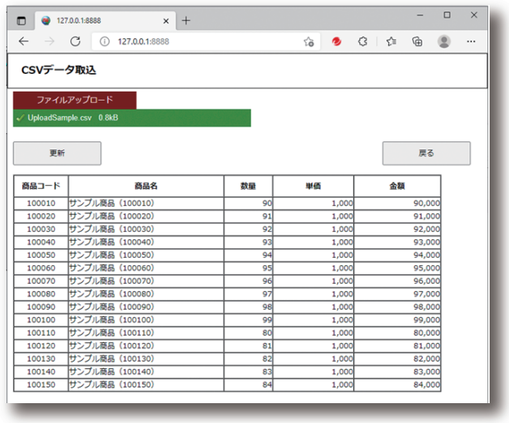


図 21 CSVファイル以外をアップロードした場合

4-2.CSVファイルの取込と明細への反映方法

前節では、CSVファイルをアップロードしてデータを画面明細に取り込むところを実装してきた。この節ではアップロードしたデータを、「更新」ボタンをクリックすることでUserSessionUnitのcdsSampleへ更新し、当画面を閉じてfrmListの明細へ反映させる機能を実装していく。

始めにUserSessionUnitにプライベート変数「FDataDspMode」と、Property「DataDspMode」を設定する。**【ソース19】**

ソース 19

UserSessionUnitのプライベート変数とProperty

```
private
  { Private declarations }
  FDataDspMode: Boolean;

public
  { Public declarations }
  procedure SampleDataCreate;   // サンプルデータ作成

  // 「明細形式の表示」のデータ表示モード（True：データ表示）
  property DataDspMode: Boolean read FDataDspMode write FDataDspMode;
end;
```

次にfrmCSVUploadを開き、メニューバーより「ファイル」→「使用するユニット」を選び、ServerControllerを選択する。そして、「更新」ボタン（btnSave）のOnClickイベントに**【ソース20～21】**に従って処理を記述する。ここでは、最初にcdsSampleのデータを全て削除した後で、CSVファイルのデータをcdsSampleへ追加している。その後でUserSessionUnitのDataDspModeプロパティを「True」にしている。DataDspModeプロパティに設定した値は、frmListに戻った際に使用する。最後に「Self.Release」で画面を閉じている。

また、「戻る」ボタン（btnClose）のOnClickイベントにも**【ソース22】**に従って処理を記述する。frmCSVUploadに対する機能の実装は以上となる。

ソース 20

frmCSVUpload.btnSaveのOnClickイベント

```
procedure TfrmCSVUpload.btnSaveClick(Sender: TObject);
var
  i: Integer;
  sStrList: TStringList;
  sRowList: TStringList;
begin
  // UserSessionのcdsSampleをクリア
  UserSession.cdsSample.EmptyDataSet;

  //CSVデータを保持するリストの作成
  sStrList := TStringList.Create;
  try
    //行データを保持するリストの作成
    sRowList := TStringList.Create;
    try
      sStrList.LoadFromFile(FUploadFile);
      //先頭行は、タイトルの為無視して、2行目から取得
      for i := 1 to sStrList.Count - 1 do
      begin
        //行のテキスト（カンマ区切り文字列）をsRowListにセット
        sRowList.Clear;
        sRowList.CommaText := sStrList[i];
```

【ソース21】へ続く

ソース 21

frmCSVUpload.btnSaveのOnClickイベント（続き）

```
        // サンプルデータを追加
        UserSession.cdsSample.Append;
        UserSession.cdsSampleSYCD.AsString    := sRowList[0];                  // 商品コード
        UserSession.cdsSampleSYNM.AsString    := sRowList[1];                  // 商品名
        UserSession.cdsSampleSURY.AsInteger   := StrToIntDef(sRowList[2], 0);  // 数量
        UserSession.cdsSampleTANK.AsInteger   := StrToIntDef(sRowList[3], 0);  // 単価
        UserSession.cdsSampleKNGK.AsCurrency  := StrToCurrDef(sRowList[4], 0); // 金額
        UserSession.cdsSample.Post;
      end;
    finally
      sRowList.Free;
    end;
  finally
    sStrList.Free;
  end;

  // データ表示モードセット
  UserSession.DataDspMode := True;

  // 閉じる
  Self.Release;
end;
```

ソース 22

frmCSVUpload.btnCloseのOnClickイベント

```
procedure TfrmCSVUpload.btnCloseClick(Sender: TObject);
begin
  // 閉じる
  Self.Release;
end;
```

続いてfrmListを開き、OnRenderイベントを【ソース23】に従って変更する。ここではUserSessionUnitのDataDspModeプロパティが「True」の場合に、明細をクリアしてからcdsSampleのデータを明細に表示している。またOnRenderイベントはfrmCSVUploadから戻ってきた際にも発生するため、frmCSVUploadの更新ボタンをクリックすれば、DataDspModeプロパティが「True」になっているため、CSVファイルのデータで洗い替えられたcdsSampleのデータがfrmListの明細に表示されることになる。

ソース 23

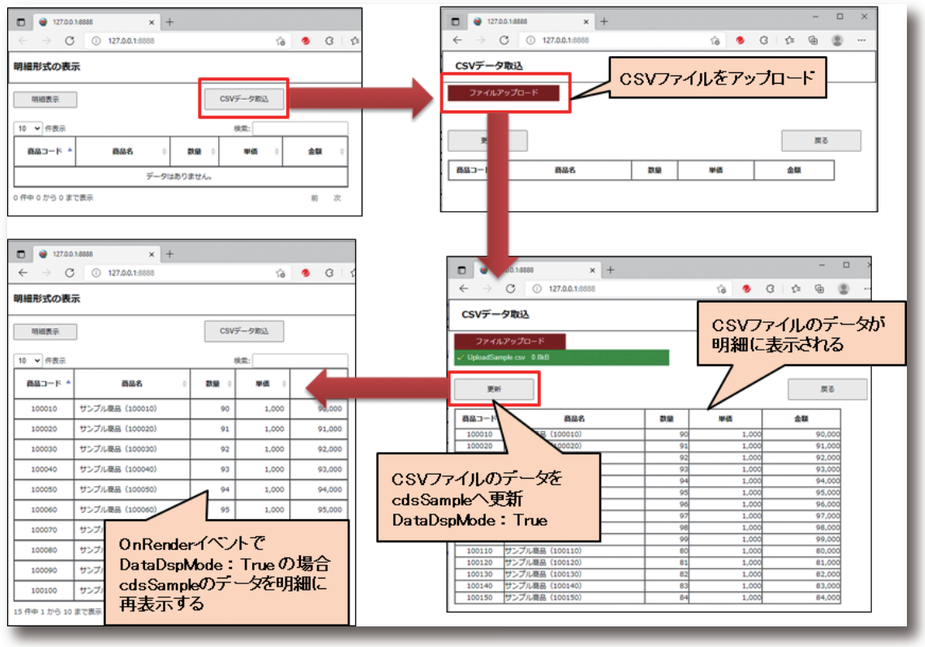
frmListのOnRenderイベント（変更）

```
procedure TfrmList.IWAppFormRender(Sender: TObject);
var
  sScriptStr: string;
begin
  sScriptStr := '$.extend( $.fn.dataTable.defaults, {';
  sScriptStr := sScriptStr + 'language: {';
  sScriptStr := sScriptStr
        + 'url: "http://cdn.datatables.net/plug-ins/9dcbecd42ad/i18n/Japanese.json"';
  sScriptStr := sScriptStr + '}';
  sScriptStr := sScriptStr + '});';

  sScriptStr := sScriptStr + '$("#TBLGDLIST").DataTable();';

  //HTMLスクリプトを追加
  AddToInitProc(sScriptStr);

  // CSVデータ取込画面から戻ってきたときに明細を再表示する
  if (UserSession.DataDspMode) then
  begin
    UserSession.DataDspMode := False;

    // 明細初期化
    GridClear;
    // 明細データセット
    ListDataSet;
  end;
end;
```


それでは実行して確認してみよう。frmCSVUploadでCSVファイルをアップロードした後「更新」ボタンをクリックすると、frmListの明細にアップロードしたCSVファイルのデータが表示されることが確認できる。【図22】

以上で「CSVファイルのアップロード／取込を行うテクニック」の説明は終了となる。

図 22　CSVファイルのアップロード／取込／明細表示



5.さいごに

本稿ではIntraWebを使用したWeb開発のTipsとして、Web画面での明細形式の表示を行う際のテクニックと、CSVファイルをアップロードして取り込み画面に表示する方法について紹介した。これらの機能は、コンポーネントの設定と、CDNで公開されているライブラリの利用、そして少しのソースコードを実装することで実現することができた。また、本稿を執筆するにあたり参考にした、IntraWebの開発元であるAtozed社が公開しているデモプログラムには、IntraWebの活用方法が他にも沢山紹介されている。こちらも是非参考にしていただきたい。
（URL：https://github.com/Atozed/IntraWeb）
本稿がIntraWebを使用してWeb開発を行う際の参考となれば幸いである。