【セッションNo.2】

Delphi/400による Webアプリケーション開発入門

株式会社ミガロ.
プロダクト事業部　技術支援課
尾崎　浩司

【アジェンダ】

- はじめに
- Webアプリケーションについて
- WebBrokerによるWebアプリ開発
  - WebBrokerとは？
  - WebBroker作成手順
  - WebBrokerによるアプリ実装例
- IntraWebによるWebアプリ開発
  - IntraWebとは？
  - IntraWeb作成手順
  - IntraWebアプリ開発のポイント
  - IntraWeb関連情報
- まとめ

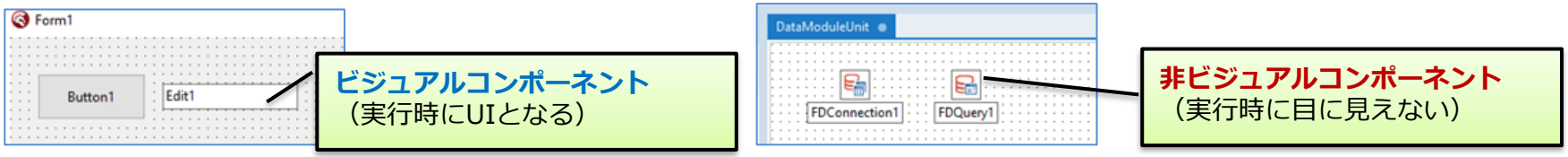
はじめに

■ Delphi/400によるアプリ開発

- Delphi/400：統合開発環境
  - Embarcadero Delphiに
    IBM iへの接続ミドルウェアを追加
  - 開発手法は、Delphiと同様の
    ビジュアル開発を実現

- Delphiに用意される2つの主なフレームワーク
  - **VCL**：Win APIをラッピングしたコンポーネントによる Win開発フレームワーク
  - **FireMonkey** ：マルチデバイス(iOS/Android/Win/Mac)開発フレームワーク

⇒ **ビジュアルコンポーネントと非ビジュアルコンポーネントの組み合わせで開発**



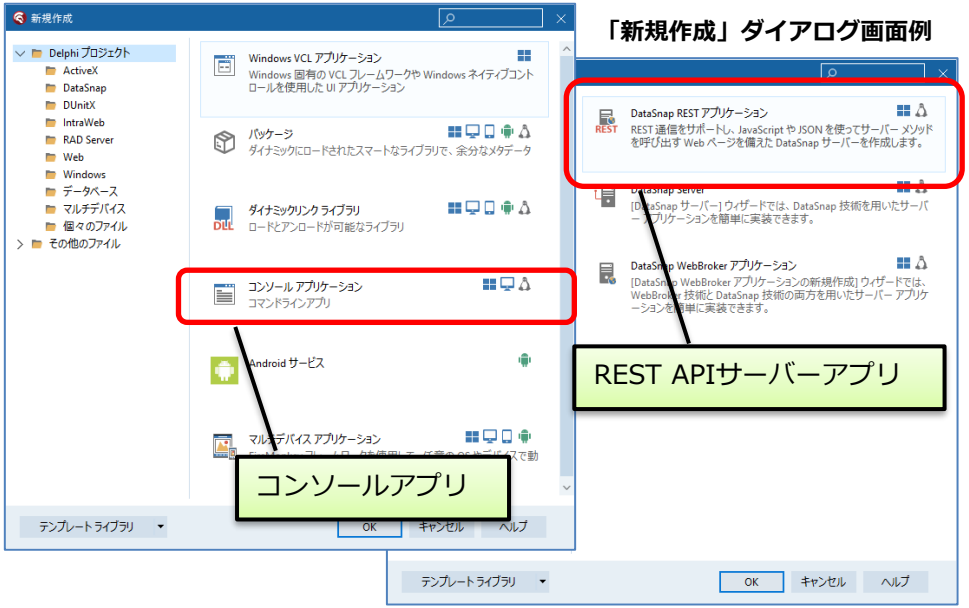
VCL、FireMonkeyのいずれも、主に**デスクトップアプリ（モバイル含む）**の
開発で使用される事が多い！ （所謂C/S形式アプリは、この方式で作成する事が多い）

Delphi/400によるアプリ開発

- デスクトップアプリ以外にも色々な形態のプログラムが可能
  - ✓ コンソールアプリ
  - ✓ REST API サーバーアプリ
  - ✓ Windows サービスアプリ
  - ✓ Webアプリ
  - …

- **[ファイル]→[新規作成]→[その他]**
  から作成したいフレームワークを選択

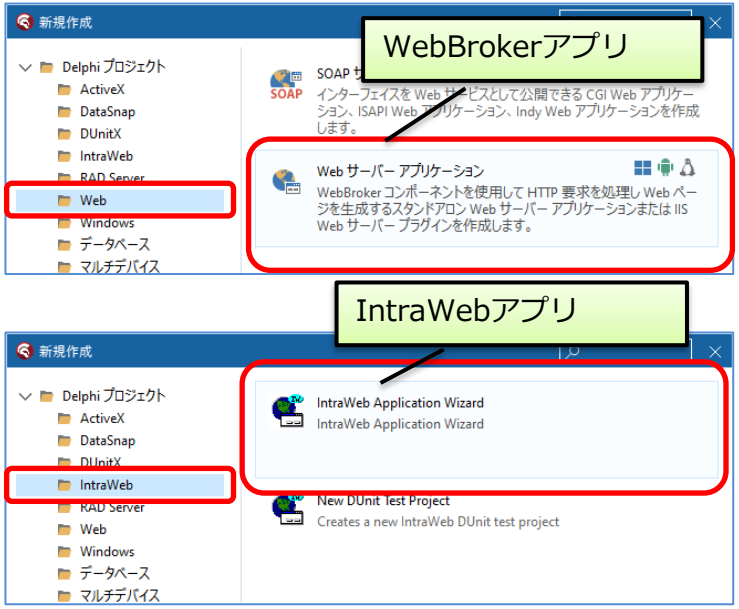
- これらは、
  主に非ビジュアルコンポーネントを使用して開発



今回は多彩なプログラム開発形態から、**Webアプリ**開発についてご紹介！

Delphi/400によるWebアプリ開発手法

- Delphi標準機能による開発
  - ✓ **WebBroker**
    - 主に**非ビジュアルコンポーネント**を使用して開発
- アドインソフトを追加した開発
  - ✓ **IntraWeb**　(Atozed Software)
    - **ビジュアルコンポーネント**による Webアプリ開発を実現
- 別ツールとの連携による開発
  - ✓ **Sencha**　(Embarcadero/IDERA)
    - フロントエンド処理は、JavaScriptベースのフレームワークSencha ExtJSで開発
    - バックエンド処理は、DelphiのRAD Server (REST-JSON)を使用して開発

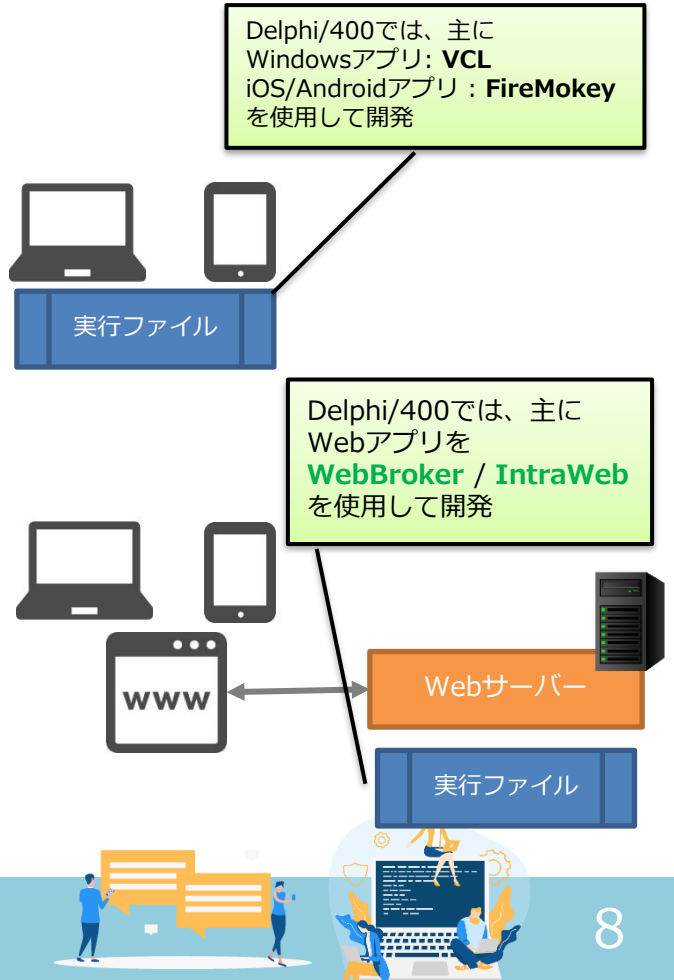


Delphi標準機能の**WebBroker**とアドイン機能の**IntraWeb**を紹介！

Webアプリケーションについて

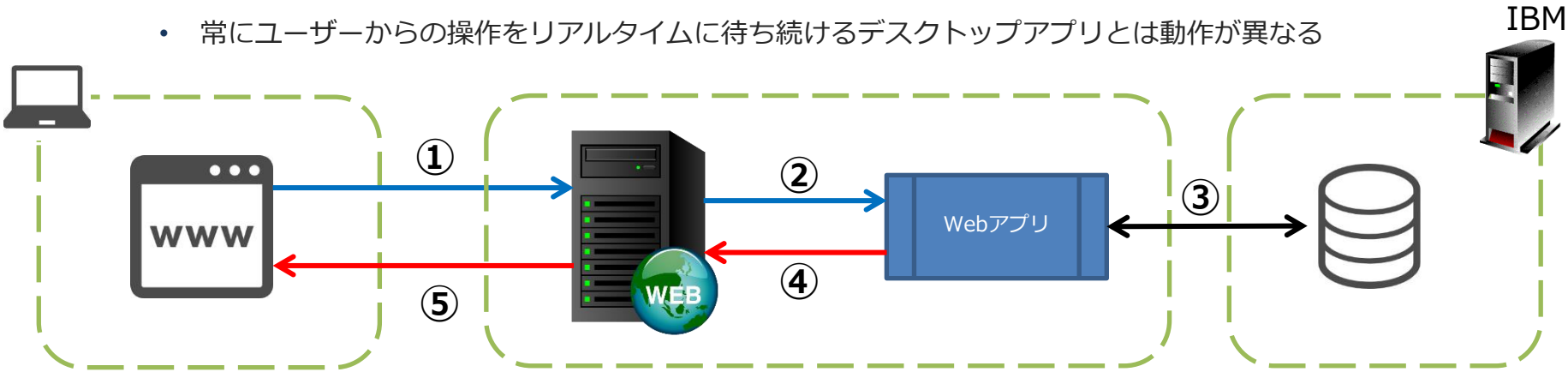
■ デスクトップアプリとWebアプリ

- 2つの主な違い
  - **デスクトップ（モバイル）アプリケーション**
    - 処理は主に、操作元の**PC/モバイル上**で行われている為、**レスポンスよく操作性が高い**
    - 画面は**OSの機能を利用して表示**する為、より表現力が高い画面を作成する事ができる
    - あらかじめPC/モバイルに**実行ファイルをインストール**しておく必要がある
  - **Webアプリケーション**
    - 処理は主に、**Webサーバ上**で行われる為、**処理の要求毎にレスポンスを待つ必要**がある
    - 画面はブラウザ上に**HTMLで表示**される為、表現力はデスクトップアプリよりは劣る事が多い
    - アプリケーションを事前にPC/モバイル上に**インストールしておく必要がない**



■ Webアプリケーション　処理の流れ

- リクエストとレスポンス
  - 常にユーザーからの操作をリアルタイムに待ち続けるデスクトップアプリとは動作が異なる



■ 処理手順

① ブラウザからURLを指定することにより、処理をWebサーバーに要求（**リクエスト**）
② WEBサーバーは、指定されたURLより対象のWEBアプリを呼出し
③ WEBアプリが処理を実行し、IBM i 等のデータベースからデータを抽出
④ WEBアプリは処理結果をHTMLの形に整形してWEBサーバーに渡す
⑤ WEBサーバーは、リクエストしたブラウザに対し、処理結果HTMLを返す（**レスポンス**）

WebBrokerを使用したWebアプリ開発

WebBrokerとは？

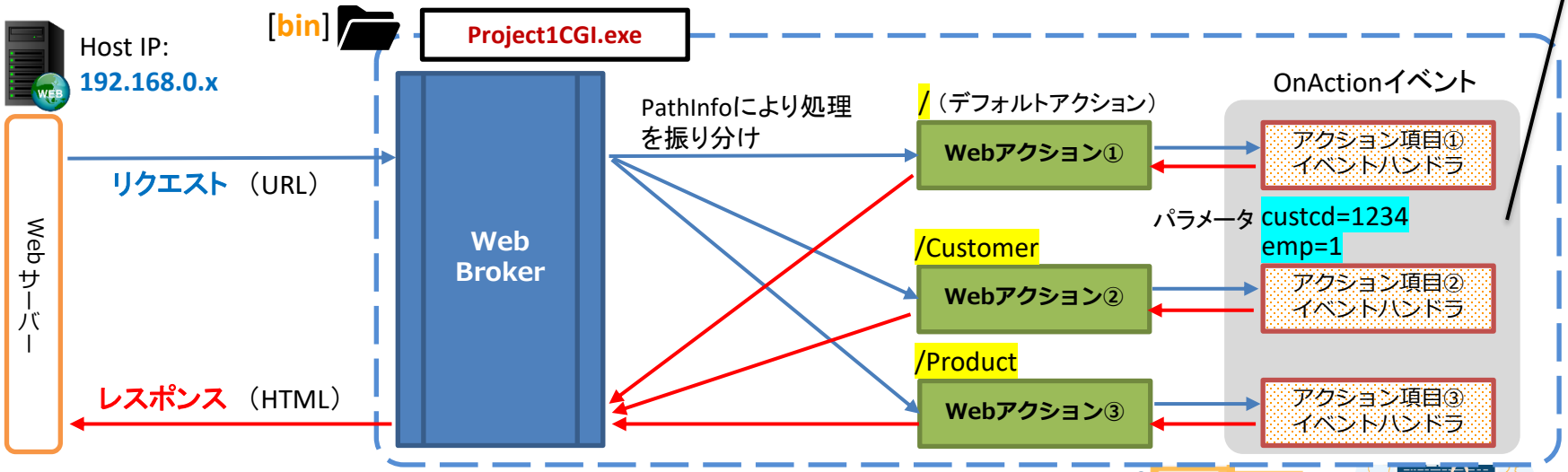
- Webアプリを構築する為のDelphi標準フレームワーク
  - リクエストをPathInfoで振り分けし、アクションを呼び出す
  - イベント内で作成されたレスポンスを呼出し元ブラウザへ返す

開発者は独自のWebアクションを追加し、アクション実行時のイベント(**OnAction**)を作成すればよい。

(1)実行パラメータ(Query)を受け取り
(2)データベース検索等の処理を実行
(3)実行結果よりレスポンスHTMLを生成

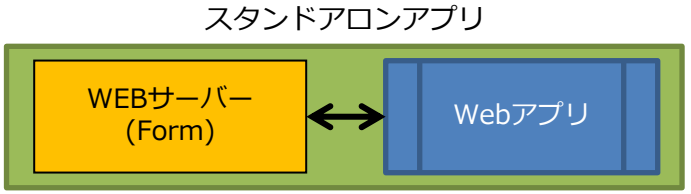
（実行方法） ホスト名 ディレクトリ プログラム名 PathInfo部 Query部

http://192.168.0.x/bin/Project1CGI.exe/Customer?custcd=1234&emp=1

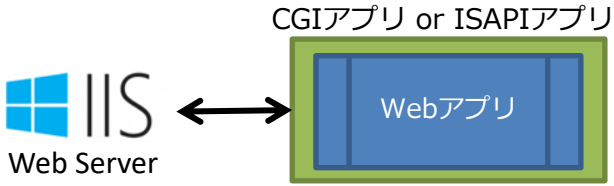


WebBrokerとは？

- 主なWebBrokerアプリの種類
  1. **スタンドアロンアプリケーション** (*.exe)
     - 専用のWebサーバーとWebアプリを両方生成
     - **Webサーバー（IIS）不要**で実行可能
     - IDE上の[実行]ボタンから、実行・**デバッグが可能**



  2. **CGIアプリケーション** (*.exe)
  3. **ISAPIアプリケーション** (*.dll)
     - IIS(Webサーバー）上で動作するWebアプリを作成
     - 実行するPC(あるいはWindows Server)に**IISを構築**し、配置
     - CGIの場合、ブラウザからの要求毎に都度プロセスを生成
       - スタンドアロン形式同様のシンプルな構成
       - 実行毎に都度DB接続が必要で、接続の為の時間が余分にかかる
     - ISAPIは、Webサーバー上の１つのプロセスで実行（要求毎にスレッドを分割して実行）
       - 実行インスタンス（セッション）毎の管理が必要
       - DB接続を共通化できる為、実行速度は有利



⇒ スタンドアロンアプリを後から、CGI/ISAPIアプリに移行する事も可能

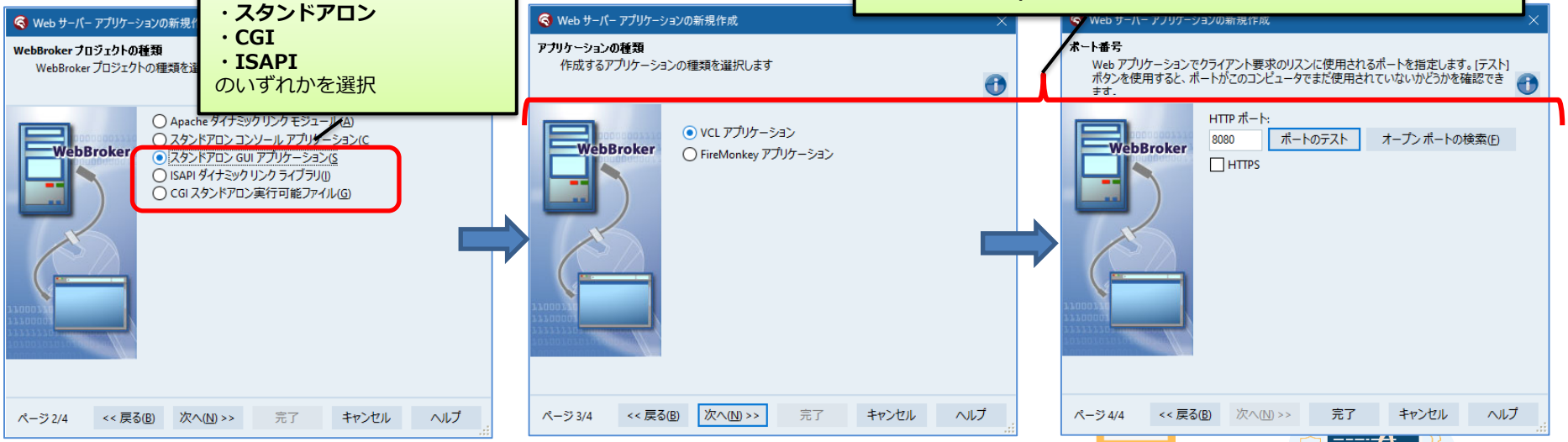
WebBroker作成手順

- WebBrokerプロジェクトの新規作成
  - **[ファイル]→[新規作成]→[その他]→[Web(WebBroker)] →[Webサーバーアプリケーション]**
  - ウィザードに従って初期設定

[プロジェクトの種類]

・スタンドアロン
・CGI
・ISAPI
のいずれかを選択

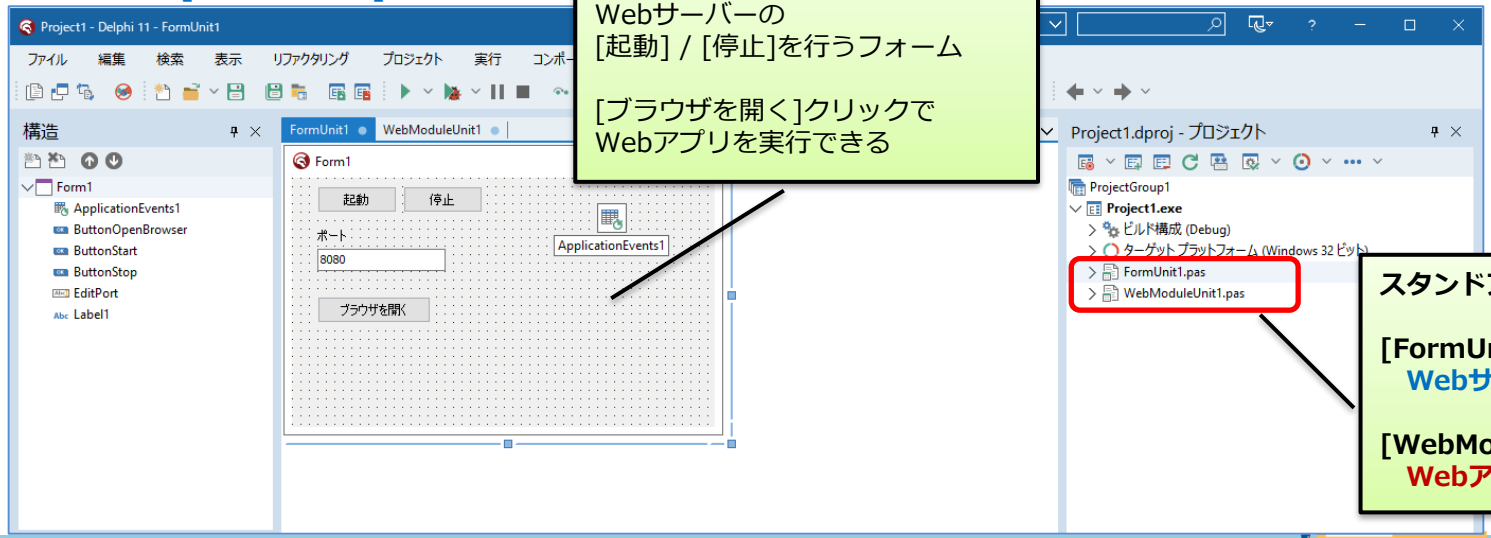
スタンドアロン形式の場合のみ、下記初期設定も行う。
・Webサーバーを VCL or FireMonkey のいずれで作成するか？
・リクエストに使用するポート番号を幾つにするか？
→　**通常、VCL / 8080ポートでOK**



WebBroker作成手順

- WebBrokerプロジェクトの新規作成
  - スタンドアロンの場合、**Webサーバーの[Form]** と**Webアプリの[Webモジュール]**の2つが生成
  - CGI/ISAPIの場合は、**Webアプリの[Webモジュール]**のみが生成

FormUnit1：**[Webサーバー]**



Webサーバーの[Form]

Webサーバーの
[起動] / [停止]を行うフォーム

[ブラウザを開く]クリックで
Webアプリを実行できる

スタンドアロンの場合、下記が生成

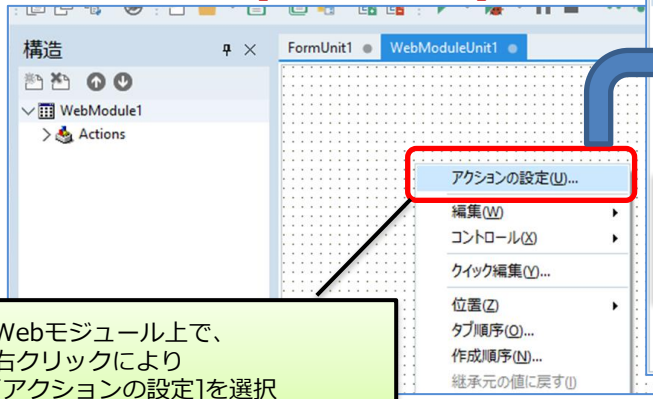
[FormUnit1.pas]
Webサーバーの[Form]

[WebModuleUnit1.pas]
Webアプリの[Webモジュール]

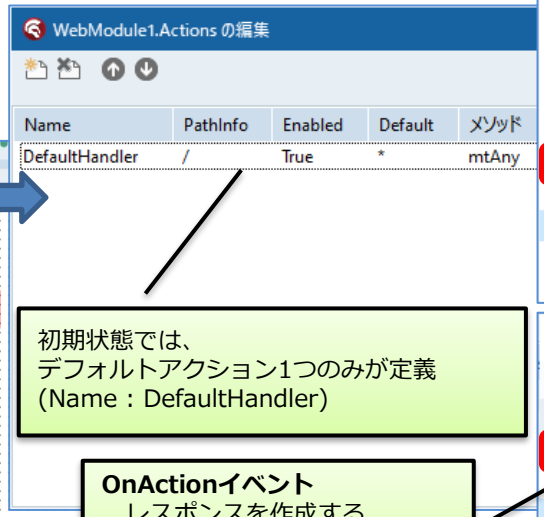
WebBroker作成手順

- WebBrokerプロジェクトの新規作成
  - **[Webモジュール]**は、[データモジュール]のような非ビジュアルコンポーネントのコンテナ
  - [アクションの設定]画面で、**Webアクション**を登録
  - **OnActionイベント**にアクション呼び出し時のレスポンスを作成するロジックを記述

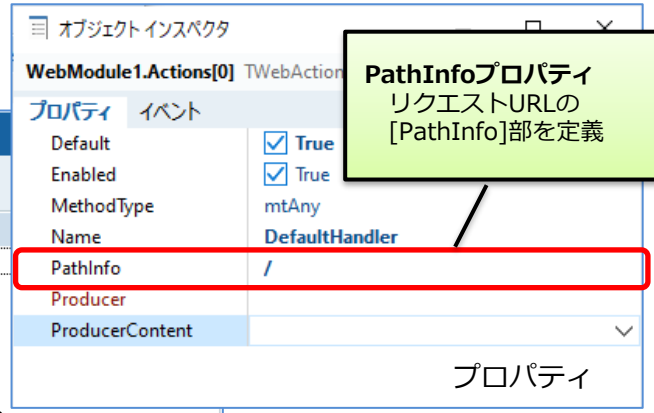
WebModuleUnit1: **[Webモジュール]**



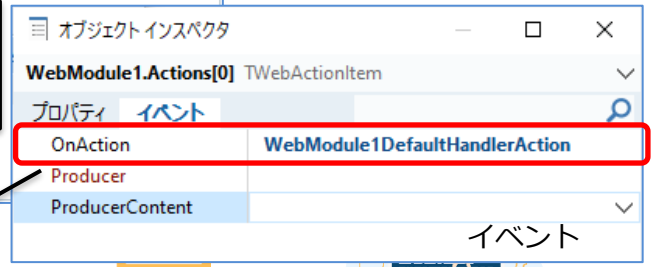
Webモジュール上で、
右クリックにより
[アクションの設定]を選択



初期状態では、
デフォルトアクション1つのみが定義
(Name：DefaultHandler)



PathInfoプロパティ
リクエストURLの
[PathInfo]部を定義



OnActionイベント
レスポンスを作成する
イベント

WebBroker作成手順

- WebBrokerプロジェクトの新規作成
  - **OnActionイベント**
    - **Requestパラメータ** ：WebBrokerにより自動的にリクエストの情報が格納される
    - **Responseパラメータ** ：開発者がレスポンス情報を独自にロジックで追加

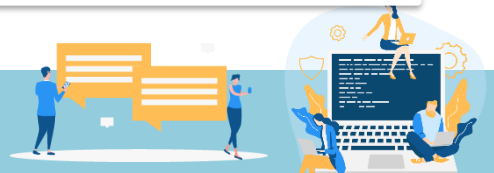
⇒ 初期状態で、Responseをセットするサンプルのソースコードが記述されている。

初期状態のDefaultHanderアクションのOnActionイベント定義

```
procedure TWebModule1.WebModule1DefaultHandlerAction(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
begin
  Response.Content :=
    '<html>' +
    '<head><title>Web サーバー アプリケーション</title></head>' +
    '<body>Web サーバー アプリケーション</body>' +
    '</html>';
end;
```

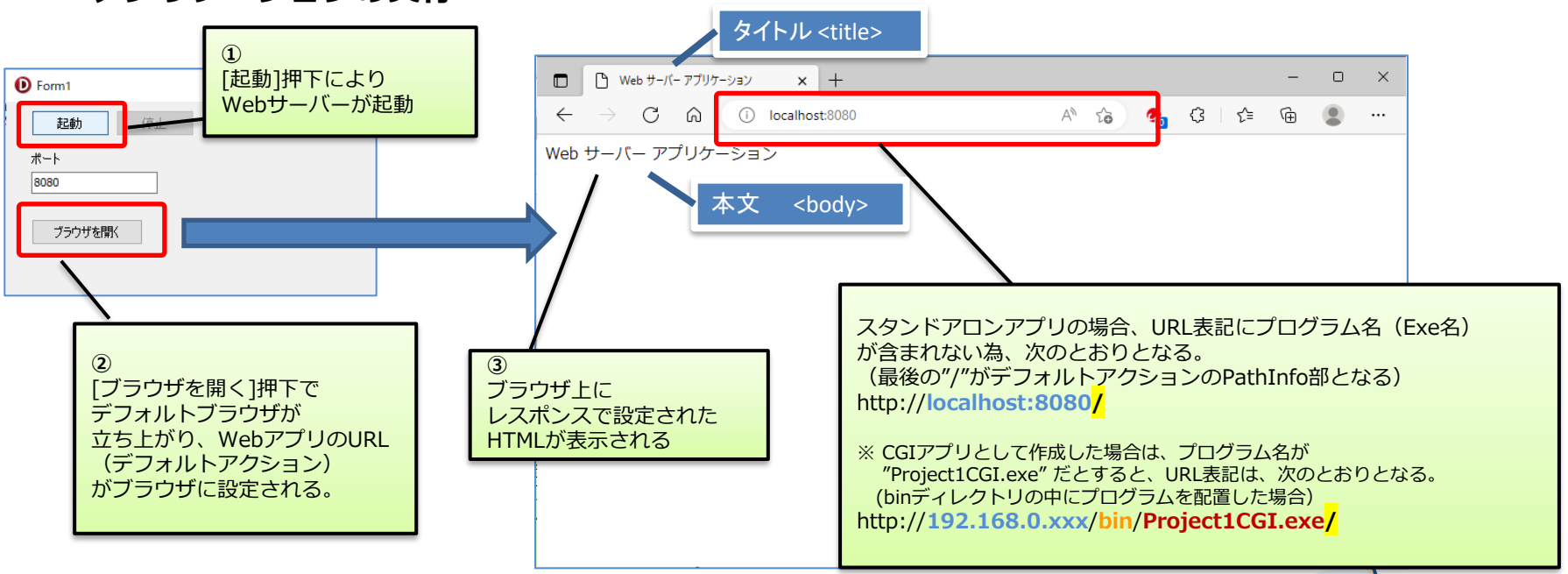
Responseパラメータの**Contentプロパティ**に
ブラウザに返したいHTMLを記述

サンプルは、下記要素を持つHTMLとなっている
タイトル <title> ： Webサーバーアプリケーション
本文 <body> ： Webサーバーアプリケーション



WebBroker作成手順

- WebBrokerプロジェクトの新規作成
  - **アプリケーションの実行**



WebBroker アプリ(1)

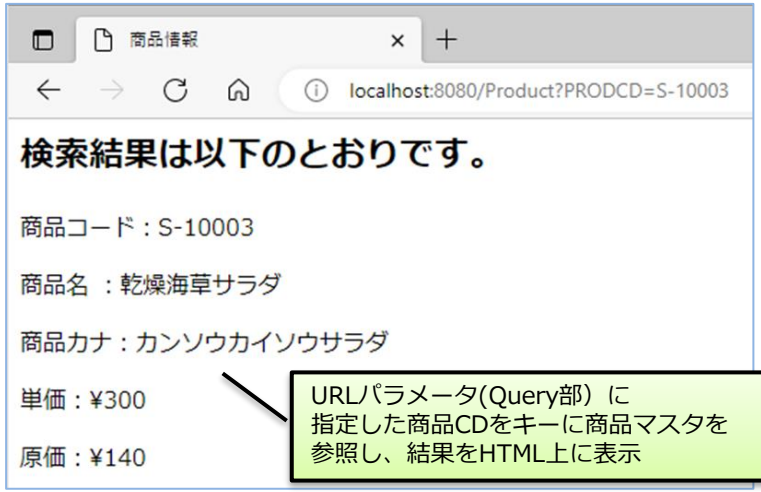
- 商品マスタ検索アプリ
  - 商品コードをパラメータに、商品マスタを検索し、結果をブラウザに表示

（実行方法）　　PathInfo部　　Query部

http://localhost:8080/Product?PRODCD=S-10003

IBM i

実行結果



ファイルレイアウト: MPRODP（商品マスタ）

DSPFMT　　レ コ ー ド 設 計 書　　日付 22/12/06　時刻 11:35:17

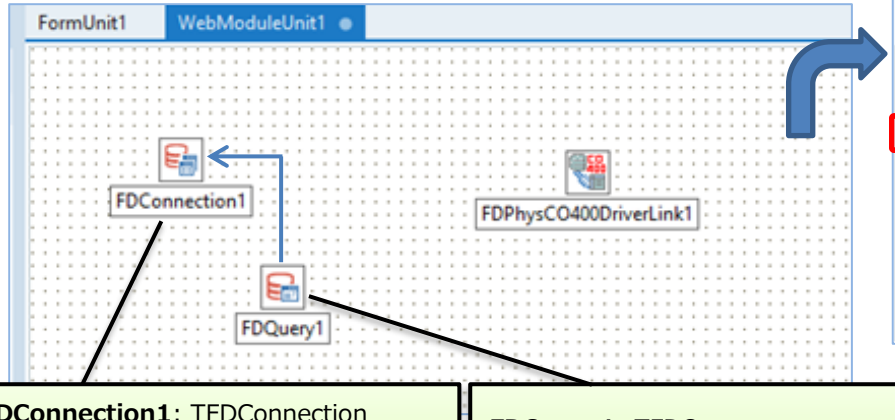
物理ファイル	TECSEM22/MPRODP	様式名	MPRODR	レコード長	110
様式記述	商品マスタ				

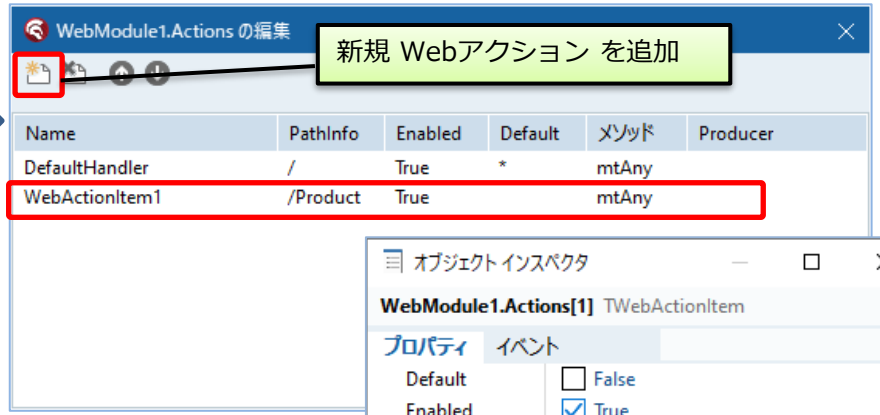
5= 詳細

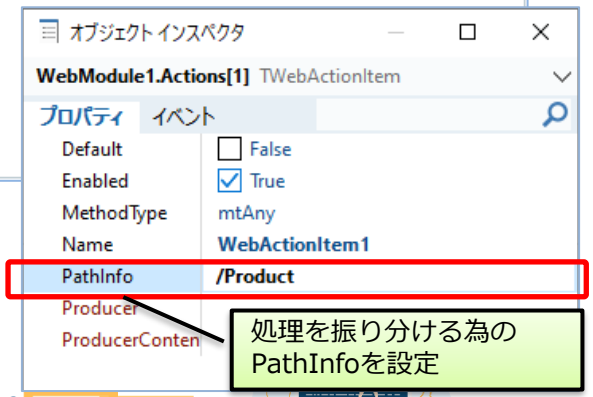
選択	項目名	桁数	属性	キー順	開始	終了	テキスト記述／欄見出し
_	ODPDCD	10	A	1 ANN	1	10	商品ＣＤ
_	ODPDNM	32	0		11	42	商品名
_	ODPDKN	42	0		43	84	商品カナ
_	ODTANK	9 0	S		85	93	単価
_	ODGEUN	9 0	S		94	102	原単価
_	ODUNIT	6	0		103	108	単位
_	ODCTCD	2	A		109	110	カテゴリＣＤ

WebBroker アプリ(1)

- Webモジュール
  - [データモジュール]同様、データベースコンポーネント等の**非ビジュアルコンポーネント**を配置
  - **Webアクション**を追加し、アクションを識別するPathInfoプロパティに”/Product”を指定







FDConnection1: TFDConnection

IBM iへの接続パラメータを定義
LoginPrompt := False;

FDQuery1: TFDQuery

Connection := FDConnection1;

WebBroker アプリ(1)

- OnActionイベント

```
procedure TWebModule1.WebModule1WebActionItem1Action(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
var
  sODPDCD: String;    //商品コード
  sODPDNM: String;    //商品名
  sODPDKN: String;    //商品カナ
  cODTANK: Currency; //単価
  cODGEUN: Currency; //原価
begin
  //商品コード取得
  sODPDCD := Request.QueryFields.Values['PRODCD'];

  //商品データ取得
  with FDQuery1 do
  begin
    SQL.Text := 'SELECT * FROM MPRODP WHERE ODPDCD = :ODPDCD';
    ParamByName('ODPDCD').AsString := sODPDCD;
    Open;
    try
      sODPDNM := FieldByName('ODPDNM').AsString;
      sODPDKN := FieldByName('ODPDKN').AsString;
      cODTANK := FieldByName('ODTANK').AsCurrency;
      cODGEUN := FieldByName('ODGEUN').AsCurrency;
    finally
      Close;
    end;
  end;

  //レスポンスを返す
  Response.Content :=
    '<html>' +
    '<head><title>商品情報</title></head>' +
    '<body>' +
    '<h2>検索結果は以下のとおりです。</h2>' +
    '<p>商品コード:' + sODPDCD + '</p>' +
    '<p>商品名 :'    + sODPDNM + '</p>' +
    '<p>商品カナ:'   + sODPDKN + '</p>' +
    '<p>単価:'       + cODTANK.ToString + '</p>' +
    '<p>原価:'       + cODGEUN.ToString + '</p>' +
    '</body>' +
    '</html>';
end;
```

Request.QueryFields プロパティ

URLのQuery部にセットされた値を取得

SQLで商品マスタを検索し、情報を取得

レスポンスとしてHTMLを返す

WebBroker アプリ(2)

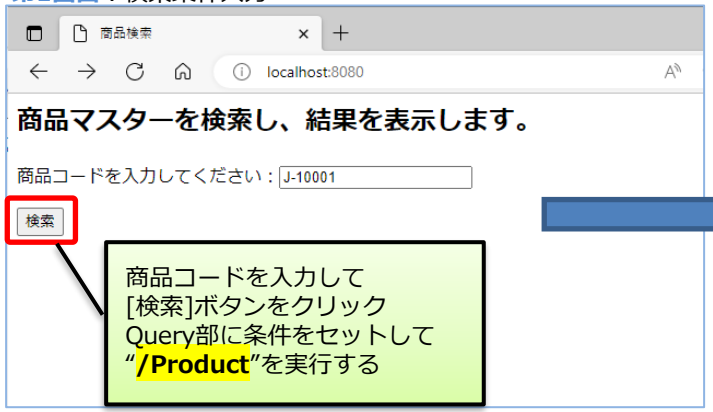
- 商品マスタ検索アプリに、商品コード入力フォームを追加
  - デフォルトページ（PathInfo:/）を開くと、検索フォームを表示
  - 検索フォーム上の[検索]ボタンクリックにより、商品検索結果の画面を表示

（実行方法）

PathInfo部

http://localhost:8080/

第1画面：検索条件入力



第2画面：検索結果表示

URL欄にフォームで指定した
条件がセットされて結果を表示

WebBroker アプリ(2)

- デフォルトアクションのOnActionイベントを変更
  - 入力フォーム（**<form>タグ**）を含むHTMLを作成し、レスポンスとして返す
  - 入力フォームで指定した値をパラメータにセットして、第2画面のプログラム(“**/product**”)を呼び出す

DefaultHanderアクションのOnActionイベント定義

```
procedure TWebModule1.WebModule1DefaultHandlerAction(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
begin
  //レスポンスを返す
  Response.Content:=
    '<html>' +
    '<head>' +
    '<title>商品検索</title>' +
    '</head>' +
    '<body>' +
    '<h2>商品マスターを検索し、結果を表示します。</h2>' +
    '<form method="get" action="/Product">' +
    '<p>商品コードを入力してください:<input type="text" name="PRODCD"></p>' +
    '<p><input type="submit" value="検索"></p>' +
    '</form>' +
    '</body>' +
    '</html>';
end;
```

HTML：<form>タグ

入力・送信フォームを作成する際に使用する要素

<form>タグに設定する属性

属性	概要
method	Webサーバーにデータを送信する形式（**get / post**） URLのQuery部に値をセットする場合は、”**get**”を使用する。
action	実行するプログラムを指定　（**PathInfo部を指定**） ※CGIアプリの場合は、下記のようにセットする。 （プログラム名：Project1CGI.exeの場合） action=“/bin/Project1CGI.exe/Product”
name	**Query部**にセットするパラメータ名

WebBroker アプリ(2)

- GetメソッドとPostメソッド
  - Webサーバーへデータを送信する方法

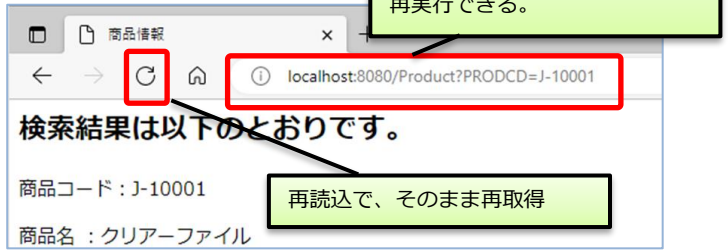
2つのメソッドの主な違い

	Getメソッド	Postメソッド
送信方法	URLにパラメータを付与して送信する方法	パラメータをURLには付与せず送信する方式
パラメータ	URLの**?以降の部分**	HTTPにおける**body部**
実行結果	実行結果画面のURL欄にパラメータが残る。（URLコピーで再実行可能）	実行結果画面にパラメータが残らない。（再読込で、再実行不可）
パラメータ	URLの文字数に依存	長いパラメータや画像情報などもセット可能
呼び出し方法	<form method="**get**">	<form method="**post**">
主な使途	情報を**検索**したり**取得**する場合に使用	情報を**登録**や**更新**する場合に使用

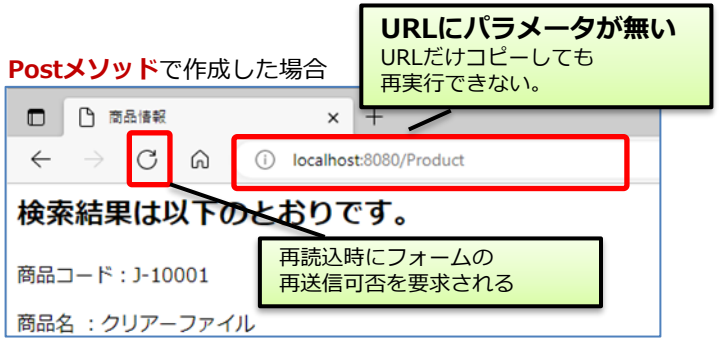
Formメソッドから送信された値をDelphiで受け取り方法

送信方法	Delphiで使用するプロパティ
getメソッド	QueryFieldsプロパティを使用する `Request.QueryFields.Values['パラメータ名']`
postメソッド	ContentFieldsプロパティを使用する `Request.ContentFields.Values['パラメータ名']`

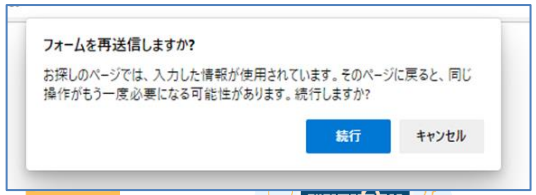
Getメソッドで作成した場合



Postメソッドで作成した場合

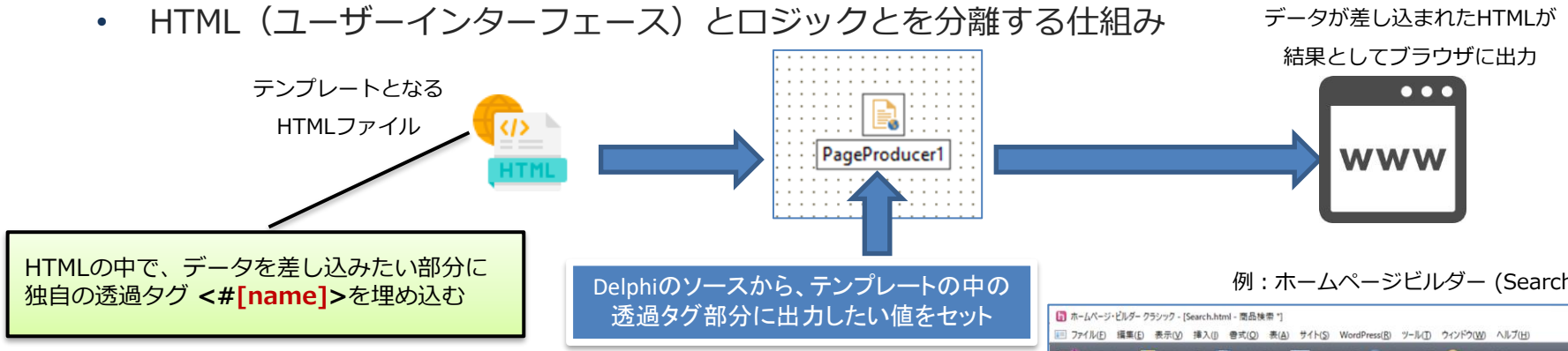


フォーム再送信可否ダイアログ

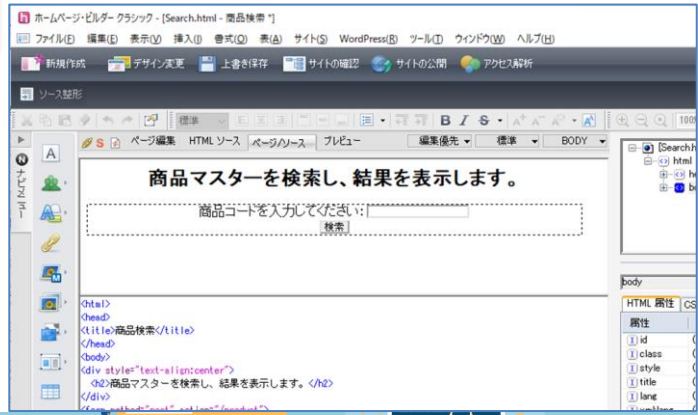


WebBroker アプリ(3)

- ページプロデューサー（TPageProducer）
  - HTML（ユーザーインターフェース）とロジックとを分離する仕組み



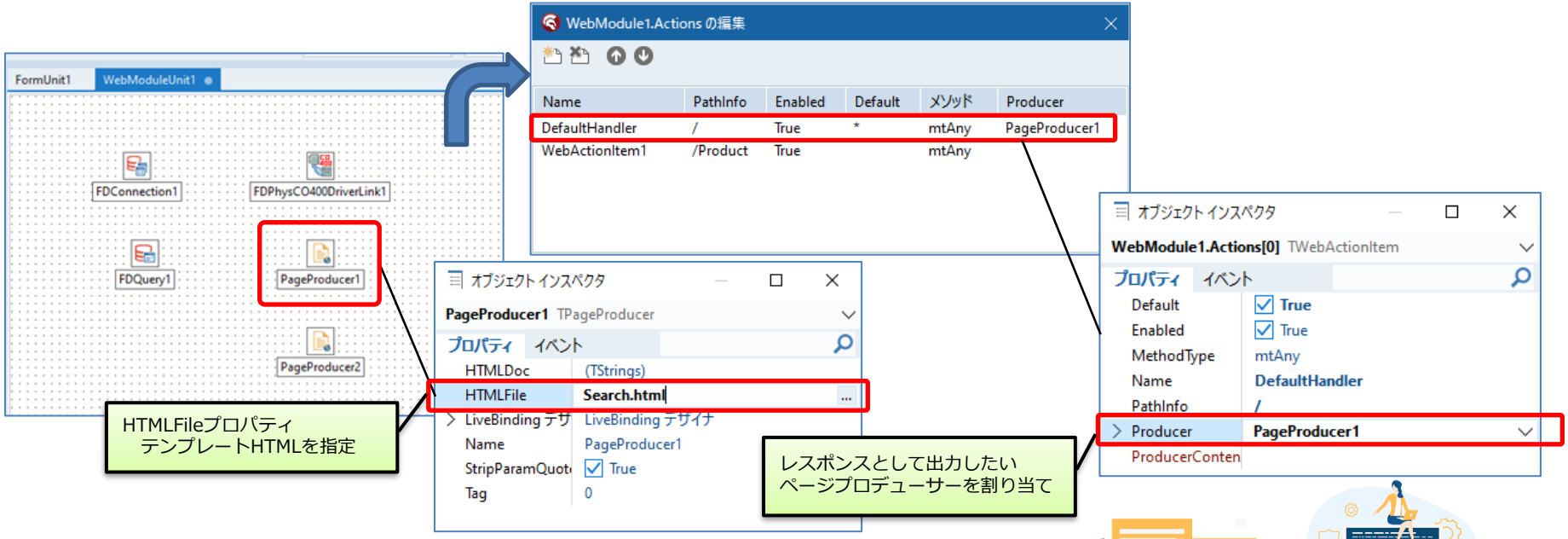
例：ホームページビルダー (Search.html)



- **ソースコード**と**HTML**を分離できる為、
  - Delphiソース部分に、HTMLが含まれなくなるので、**Webモジュールがシンプル**になる
  - HTML部分は、一般的な**HTML作成ツール**（ホームページビルダーなど）で作成ができる
  - HTMLレイアウト変更時に、プログラムの**再コンパイルが不要**

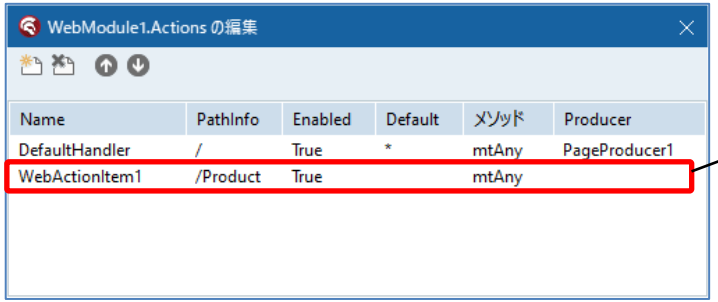
WebBroker アプリ(3)

- シンプルなHTMLの分離
  - **第1画面**：商品コード入力フォームを**Search.html** として用意
  - 単純にテンプレートを出力するだけであれば、OnActionイベントの定義も不要



WebBroker アプリ(3)

- テンプレートHTMLにデータを差し込む実装
  - OnActionイベントでデータを取得後、ページプロデューサーを関連付けしてレスポンスを作成



Webアクション："/Product" OnActionイベント

```
procedure TWebModule1.WebModule1WebActionItem1Action(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
begin
  //商品コード取得
  FODPDCD := Request.ContentFields.Values['PRODCD'];

  //商品データ取得
  with FDQuery1 do
  begin
    SQL.Text := 'SELECT * FROM MPRODP WHERE ODPDCD = :ODPDCD';
    ParamByName('ODPDCD').AsString := FODPDCD;
    Open;
    try
      FODPDNM := FieldByName('ODPDNM').AsString;
      FODPDKN := FieldByName('ODPDKN').AsString;
      FODTANK := FieldByName('ODTANK').AsCurrency;
      FODGEUN := FieldByName('ODGEUN').AsCurrency;
    finally
      Close;
    end;
  end;

  PageProducer2.HTMLFile := 'Product.html';
  Response.Content := PageProducer2.Content;
end;
```

SQLにより取得した商品情報をグローバル変数(private)にセット

第2画面である
テンプレートのHTML(**Product.html**)を設定し、
ページコンテンツをレスポンスとしてセット

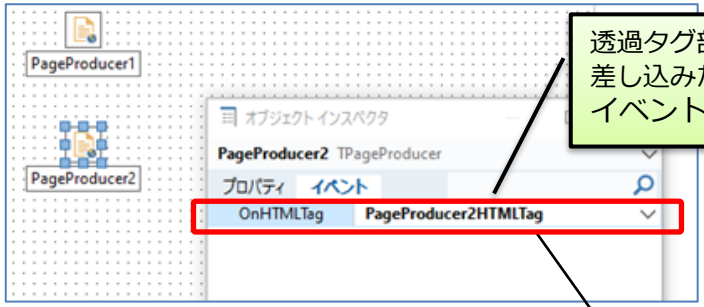
WebBroker アプリ(3)

- テンプレートHTMLにデータを差し込む実装
  - PageProducerの**OnHTMLTagイベント**を使用して、HTMLテンプレートに値をセット

テンプレートHTML（**Product.html**）

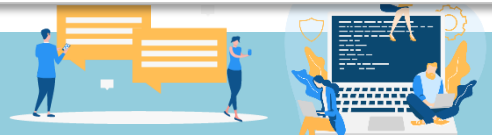
```
<html>
  <head>
    <title>商品情報</title>
  </head>
  <body>
    <h2>検索結果は以下のとおりです。</h2>
    <p>商品コード：<#ODPDCD></p>
    <p>商品名　：<#ODPDNM></p>
    <p>商品カナ：<#ODPDKN></p>
    <p>単価：<#ODTANK></p>
    <p>原価：<#ODGEUN></p>
  </body>
</html>
```

データを差し込みしたい部分に
<#[name]>の透過タグを設定



透過タグ部分にデータを
差し込みたい場合、OnHTMLTag
イベントを定義

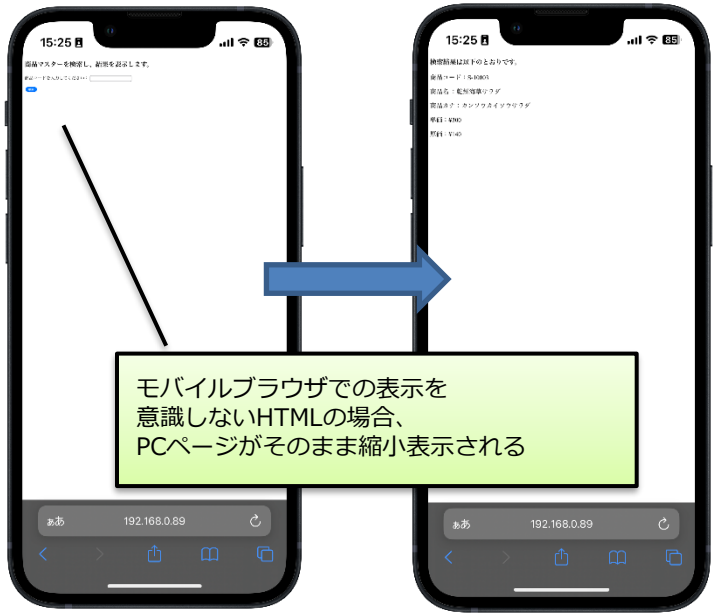
```
procedure TWebModule1.PageProducer2HTMLTag(Sender: TObject; Tag: TTag;
  const TagString: string; TagParams: TStrings; var ReplaceText:
string);
begin
  //独自のタグ部分を実行時に置き換える
  if TagString = 'ODPDCD' then ReplaceText := FODPDCD;
  if TagString = 'ODPDNM' then ReplaceText := FODPDNM;
  if TagString = 'ODPDKN' then ReplaceText := FODPDKN;
  if TagString = 'ODTANK' then ReplaceText := FODTANK.ToString;
  if TagString = 'ODGEUN' then ReplaceText := FODGEUN.ToString;
end;
```



WebBroker アプリ(3)

- 実行例
  - テンプレートHTMLの調整（レスポンシブ対応）で、スマートフォンに最適化する事が可能

調整前のテンプレートHTMLで実行



レスポンシブ対応したテンプレートHTMLに入れ替え

WebBroker アプリ(4)

- 取引先一覧出力アプリ
  - 取引先マスタから検索条件に合致する
    データを抽出し、表形式でブラウザに表示

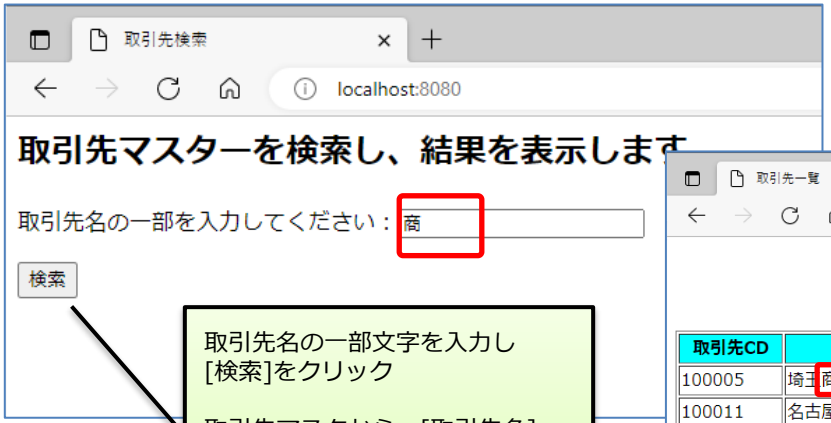
ファイルレイアウト: MACNTP（取引先マスタ）

DSPFMT　　レ コ ー ド 設 計 書　　日付 22/12/07　時刻 17:48:41

物理ファイル	TECSEM22/MACNTP	様式名	MACNTR	レコード長	222
様式記述	取引先マスタ				

5= 詳細

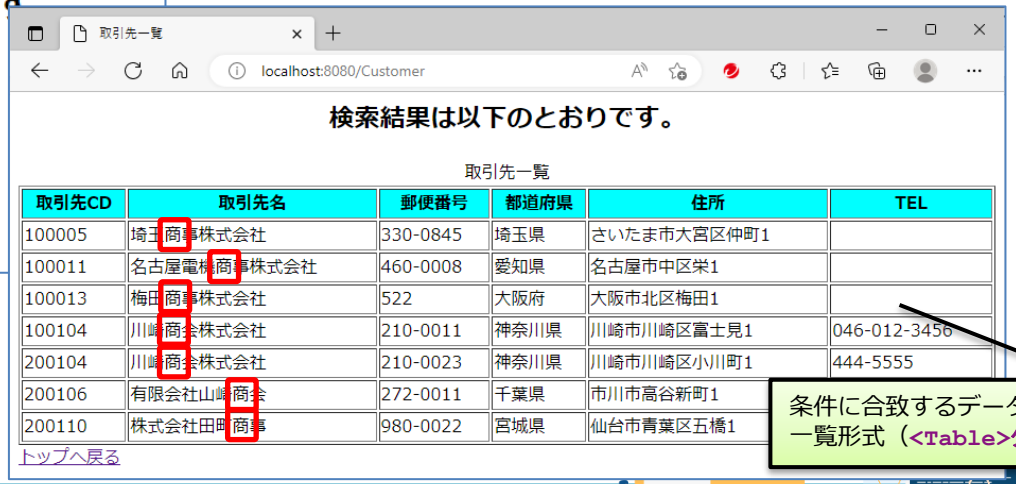
選択	項目名	桁数	属性	キー順	開始	終了	テキスト記述／欄見出し
_	NTACCD	6	A	1 ANN	1	6	取引先ＣＤ
_	NTACNM	42	0		7	48	取引先名
_	NTACKN	42	0		49	90	取引先カナ
_	NTACYB	8	A		91	98	郵便番号
_	NTTDFK	10	0		99	108	都道府県
_	NTACAD	62	0		109	170	住所
_	NTACTL	14	A		171	184	ＴＥＬ
_	NTACFX	14	A		185	198	ＦＡＸ



取引先名の一部文字を入力し
[検索]をクリック

取引先マスタから、[取引先名]
を部分一致条件で検索

Postメソッドで
“/Customer”アクションを実行



取引先CD	取引先名	郵便番号	都道府県	住所	TEL
100005	埼玉商事株式会社	330-0845	埼玉県	さいたま市大宮区仲町1	
100011	名古屋電機商事株式会社	460-0008	愛知県	名古屋市中区栄1	
100013	梅田商事株式会社	522	大阪府	大阪市北区梅田1	
100104	川崎商会株式会社	210-0011	神奈川県	川崎市川崎区富士見1	046-012-3456
200104	川崎商会株式会社	210-0023	神奈川県	川崎市川崎区小川町1	444-5555
200106	有限会社山崎商会	272-0011	千葉県	市川市高谷新町1	
200110	株式会社田町商事	980-0022	宮城県	仙台市青葉区五橋1	

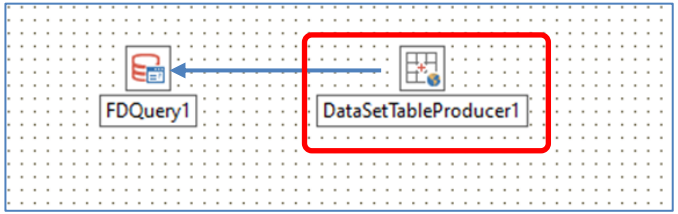
トップへ戻る

条件に合致するデータを
一覧形式（**<Table>タグ**）で出力

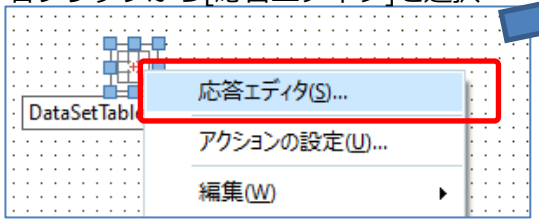
WebBroker アプリ(4)

- TDataSetTableProducer
  - TFDQuery等のDataSetから、HTMLのTableタグを自動生成する

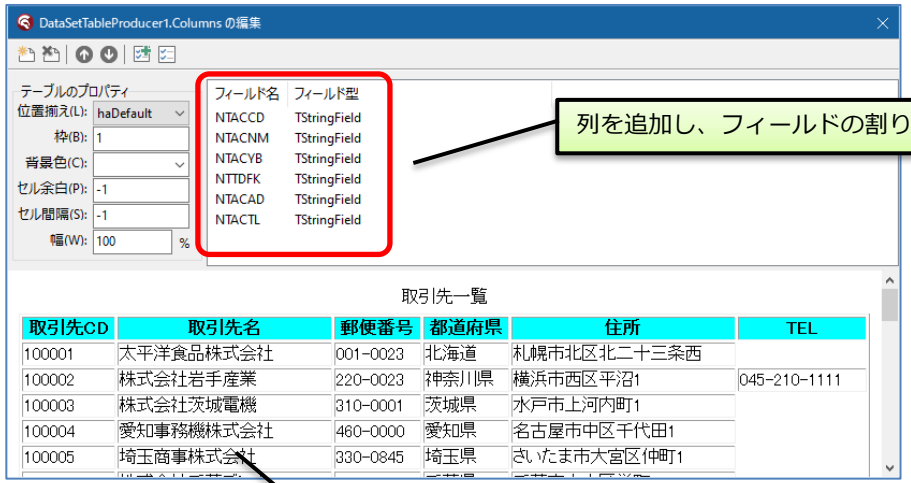
TDataSetProducerのDataSetプロパティを設定



右クリックから[応答エディタ]を選択



出力したいフィールドを追加しレイアウトを作成



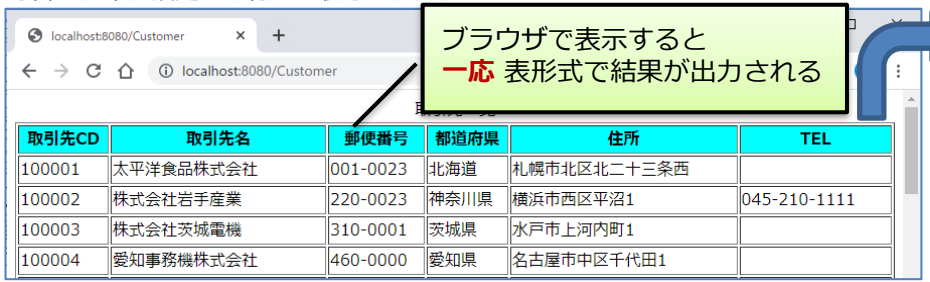
列を追加し、フィールドの割り当てを行う

DataSetをオープンした状態の場合、
出力イメージのプレビュー確認が可能

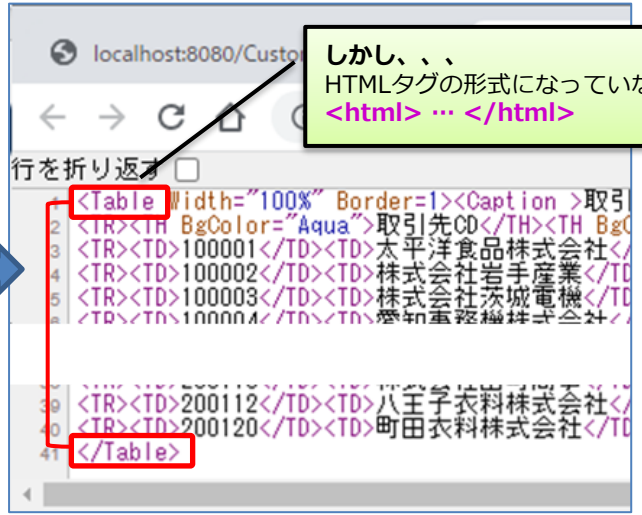
WebBroker アプリ(4)

- TDataSetTableProducer
  - HTMLの<table>タグ部分のみ生成される

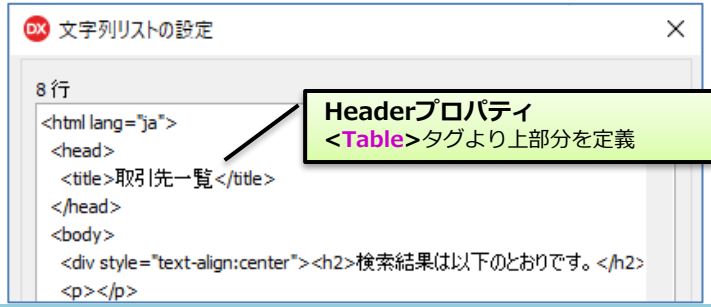
応答エディタ設定のみ行った状態のアプリ



ページのソースを表示



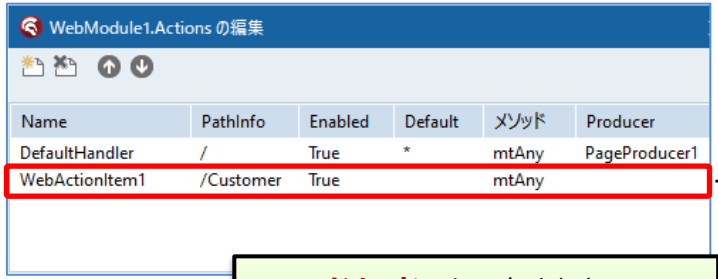
- HTMLタグ形式で出力できるよう、Header/Footer各プロパティに情報を追加



生成イメージ

WebBroker アプリ(4)

- TDataSetTableProducer
  - 条件に合致したデータをSQLで抽出し、結果のデータセットを使用してレスポンスを返す。



Postメソッドでセットされた
パラメータ[InpVal]の値を受け取る

DataSetTableProducer1で自動生成されたHTMLを
レスポンスをして返す

Webアクション："**/Customer**" OnActionイベント

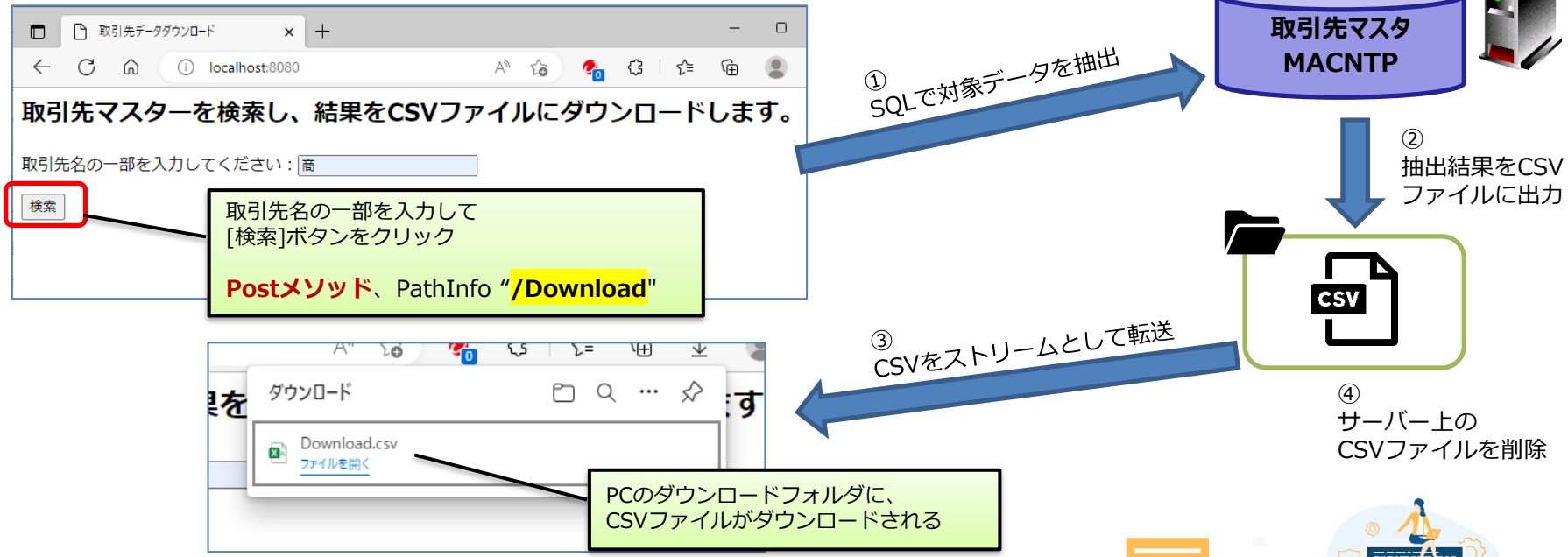
```
procedure TWebModule1.WebModule1WebActionItem1Action(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
var
  sInpVal: String;
begin
  //検索条件入力値取得(Postメソッド)
  sInpVal := '%' + Request.ContentFields.Values['InpVal'] + '%';

  //取引先条件セット
  with FDQuery1 do
  begin
    SQL.Text := 'SELECT * FROM MACNTP ' +
                'WHERE NTACNM LIKE :NTACNM ' +
                'ORDER BY NTACCD';
    ParamByName('NTACNM').AsString := sInpVal;
  end;

  //レスポンスを返す
  Response.Content := DataSetTableProducer1.Content;
end;
```

WebBroker アプリ(5)

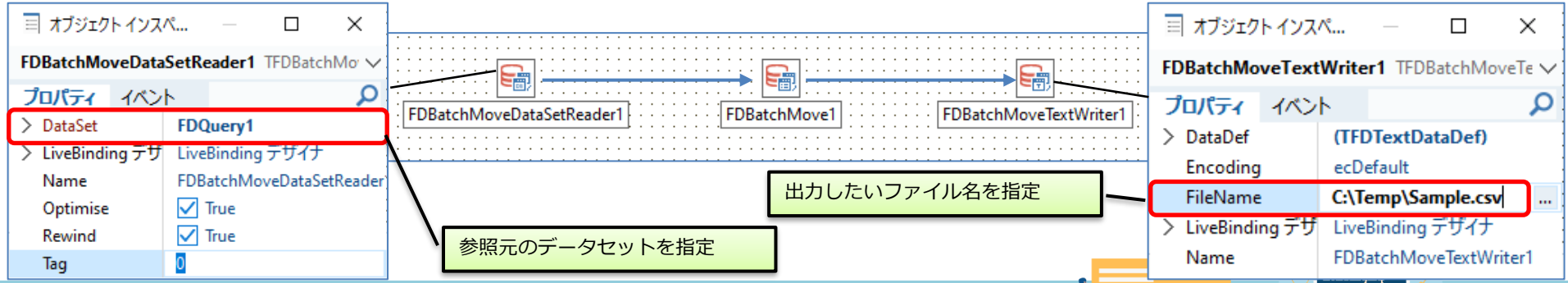
- ファイルダウンロードアプリ
  - 検索条件に合致する取引先データをCSVファイル形式でダウンロード



WebBroker アプリ(5)

- CSVファイル生成：　TFDBatchMove
  - 種類の異なるデータ移動元から移動先へデータ移動を行うためのエンジンを実装
    【活用例】 ➢ [IBM i]のデータを[SQL Server]のデータへコピー
    ➢ [IBM i]のデータをテキストファイル（CSV等）へコピー
  - [参照元：Reader] と [転送先：Writer] を関連付けし、Executeメソッドで転送

	DataSet	テキスト
Reader（参照元）	TFDBatchMoveDataSetReader	TFDBachMoveTextReader
Writer（転送先）	TFDBatchMoveDataSetWriter	TFDBatchMoveTextWriter



WebBroker アプリ(5)

Webアクション："**/Download**" OnActionイベント

```
procedure TWebModule1.WebModule1WebActionItem1Action(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
var
  sInpVal: String;
  sFileName: String;
  fStream: TFileStream;
begin
  //検索条件入力値取得(Postメソッド)
  sInpVal := '%' + Request. ContentFields.Values['InpVal'] + '%';

  //取引先条件セット
  with FDQuery1 do
  begin
    SQL.Text := 'SELECT * FROM MACNTP ' +
                'WHERE NTACNM LIKE :NTACNM ' +
                'ORDER BY NTACCD';
    ParamByName('NTACNM').AsString := sInpVal;
  end;

  //Webサーバー上に出力する一時ファイル名を取得
  sFileName := TPath.GetTempFileName;

  //CSVファイルを生成(一時ファイル名で保存)
  FDBatchMoveDataSetReader1.DataSet := FDQuery1;
  FDBatchMoveTextWriter1.FileName := sFileName;
  FDBatchMove1.Execute;

  //作成したCSVファイルをクライアントに送信
  fStream := TFileStream.Create(sFileName, fmOpenRead);
  try
    Response.ContentType := 'text/csv';
    Response.ContentStream := fStream;
    Response.SendResponse;
  finally
    fStream.Free;
  end;

  //生成した一時ファイルを削除
  DeleteFile(sFileName);
end;
```

TPath.GetTempFileName

ユニークな一時ファイルを生成
（使用する場合、uses System.IOUtils を追加）

ReaderとWriterの情報を設定し、
Executeメソッドで転送

**生成したCSVファイルを
ストリームとして送信**

送信後、サーバー上の
一時ファイルは削除

WebBroker アプリ(5)

- Responseパラメータ
  - プロパティ

HTMLを返す

ファイルを返す

プロパティ	概要
Content	送信するContenctを指定 ［記述例］ Response.Content := ‘<html> … </html>’;
ContentType	送信したいContentのメディアタイプを指定 ［記述例］ Response.ContentType := 'text/csv’; 参考：メディアタイプ一覧： https://developer.mozilla.org/ja/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types
ContentStream	送信したいストリームオブジェクトを指定 ［記述例］ Response.ContentStream := fStream;

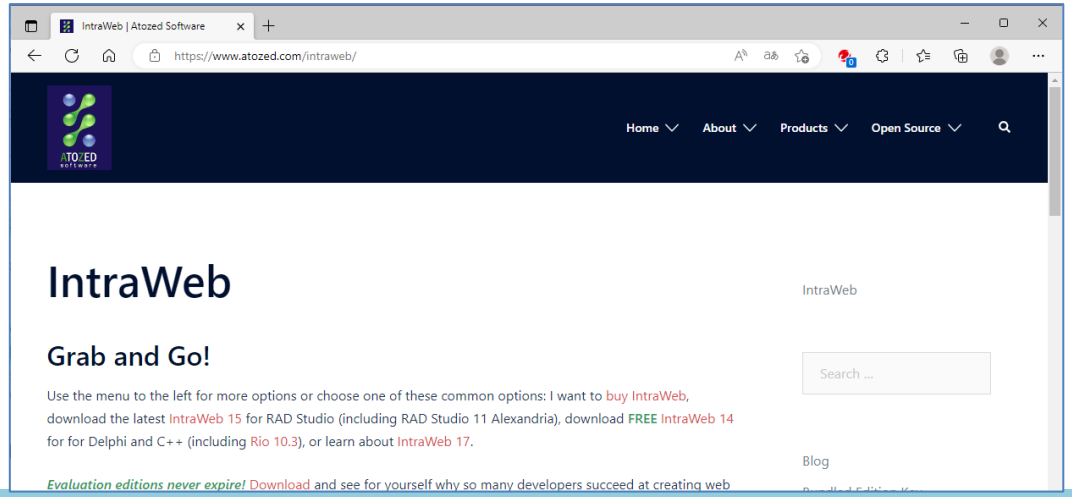
  - メソッド

メソッド	概要
SendResponse	予め設定したContentを送信する。 （WebアクションのOnActionイベント実行後に、自動的に送信される為、通常は記述不要） ［記述例］ Response.SendResponse;
SendRedirect	パラメータで指定したURIへリダイレクトする ［記述例］ Response.SendRedirect(‘https://www.migaro.co.jp/’)
SendStream	パラメータに指定したストリームをそのまま転送する ［記述例］ Response.SendStream(fStream);

IntraWebを使用した開発

IntraWebとは？

- Atozed Software社が提供する
  Delphi及びDelphi/400でWebアプリ構築を簡略化するツール
  - VCLフォームアプリケーション同様に、**フォーム上にコンポーネントを配置するビジュアル設計**が可能。
  - Delphi 10.2 Tokyoまでは、Delphi製品にIntraWebバンドル版が付属。（11 Alexandriaには搭載されない。）
  - 2022年12月現在、製品最新版は、「**IntraWeb15**」（Delphi 2009以降で使用可能）
    - 次期バージョン「IntraWeb17」がHP上に明記されているが、2022年12月現在、β版となっている。



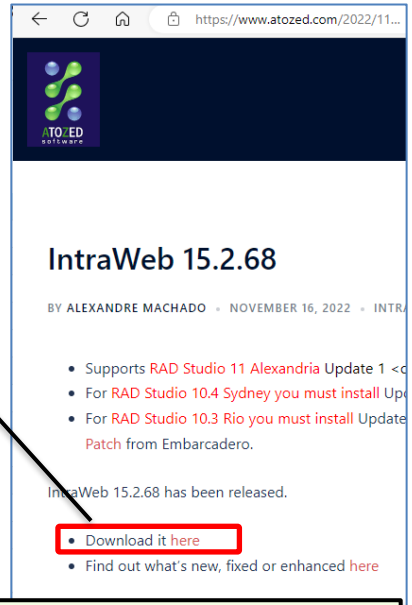
Atozed Software
IntraWeb製品ページ

https://www.atozed.com/intraweb/

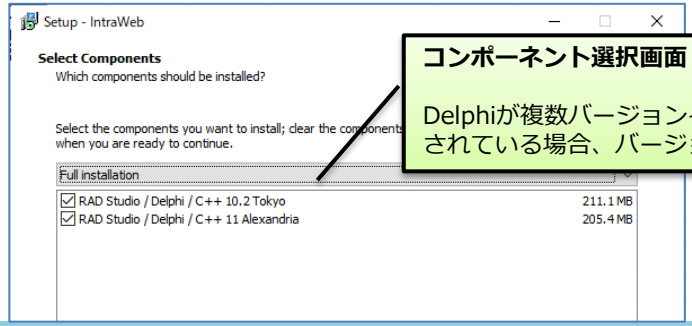
IntraWeb導入方法

- IntraWebのダウンロード
  - https://www.atozed.com/intraweb/download/v15/
    - 2022年12月現在　最新版：ver. 15.2.68
    - IntraWebは、Delphi 2009以降に対応
  - Delphi/400 10.2 Tokyo以前のバージョンの場合、インストール前に、Delphiに標準搭載されるバンドル版の削除が必要
    https://doc.atozed.com/en/iw14/gettingstarted/intraweb-bundled-removal-tool/
- IntraWeb15 インストール手順
  - ダウンロードしたインストーラーを実行（Delphiは終了しておく事）
  - 基本的には、インストールウィザードを初期設定で進めばよい。

ダウンロードページ

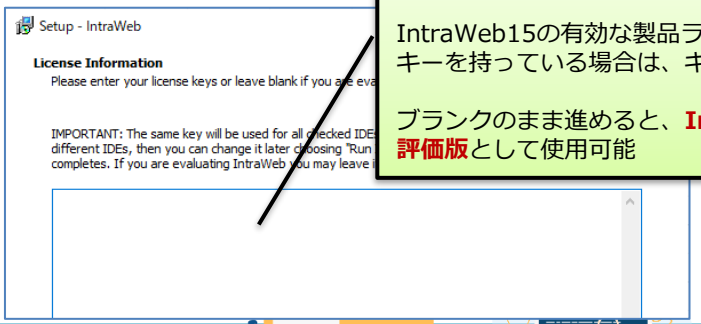


Download it here.
の部分をクリックすると
インストーラーが取得できる



コンポーネント選択画面

Delphiが複数バージョンインストールされている場合、バージョンを選択



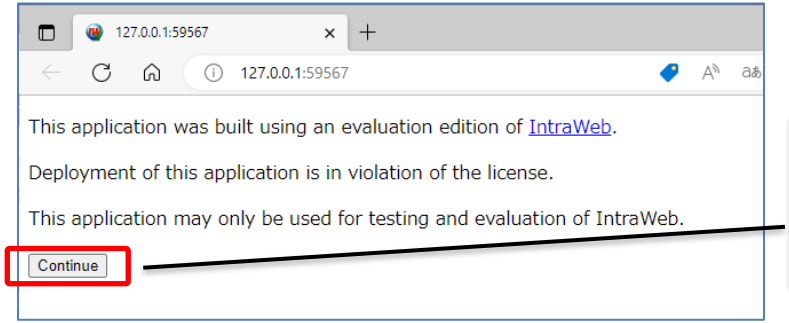
ライセンスキー入力画面

IntraWeb15の有効な製品ライセンスキーを持っている場合は、キーを入力

ブランクのまま進めると、**IntraWeb評価版**として使用可能

IntraWeb導入方法

- IntraWeb評価版について
  - IntraWebの機能評価の為に使用するものだが、**評価期限に制限無く、継続使用する事が可能**
  - ただし、製品ライセンス版に対し**以下の制限**がある
    - Webサーバーが入ったPCのブラウザからのみアクセス可能
      http://localhost:[ポート番号] (あるいは、http://127.0.0.1:[ポート番号])からのみ接続可能
    - 起動時に、評価版である旨を表す画面が時折表示される
    - 運用目的での利用やWebサーバー上への配置は許可されない



Webアプリ実行時に
評価版でビルドされたプログラム
である旨が表示される事がある。

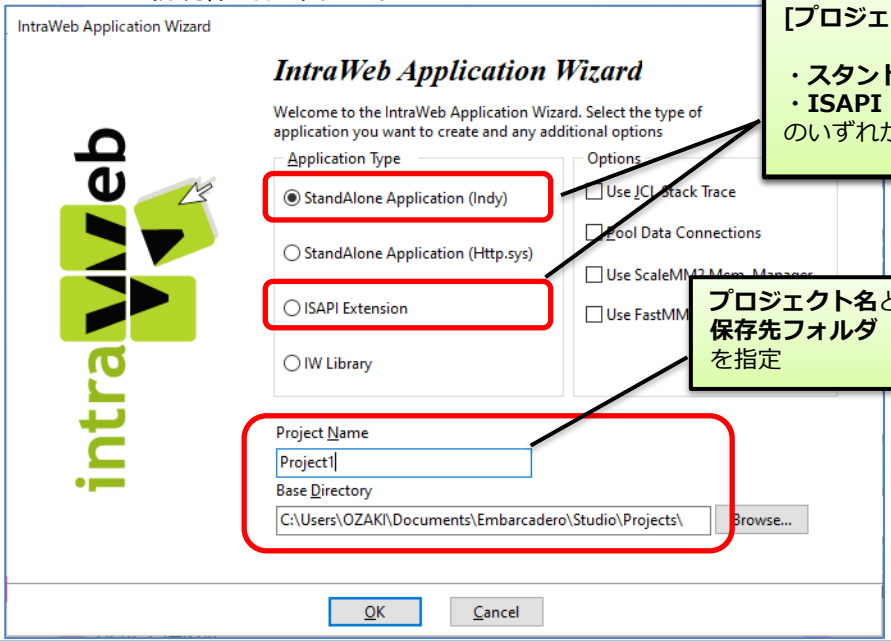
[Continue]クリックでアプリ画面に進む

  - IntraWeb製品ライセンス版は、Delphi/400ユーザー様はミガロ．から購入が可能
    - IntraWebが1年間リビジョンアップ可能な正式ライセンスキーを提供

IntraWeb作成手順

- IntraWebプロジェクトの新規作成
  - **[ファイル]→[新規作成]→[その他]→[IntraWeb] →[IntraWeb Application Wizard]**

IntraWeb新規作成ウィザード

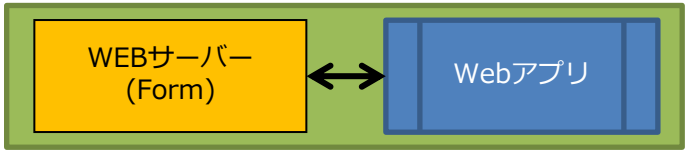


[プロジェクトの種類]

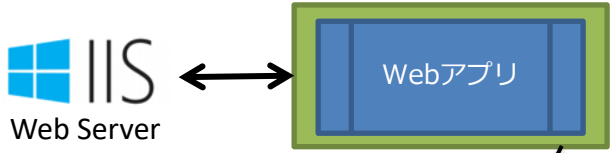
・スタンドアロン
・ISAPI
のいずれかを選択

プロジェクト名と
保存先フォルダ
を指定

スタンドアロンアプリ



ISAPIアプリ

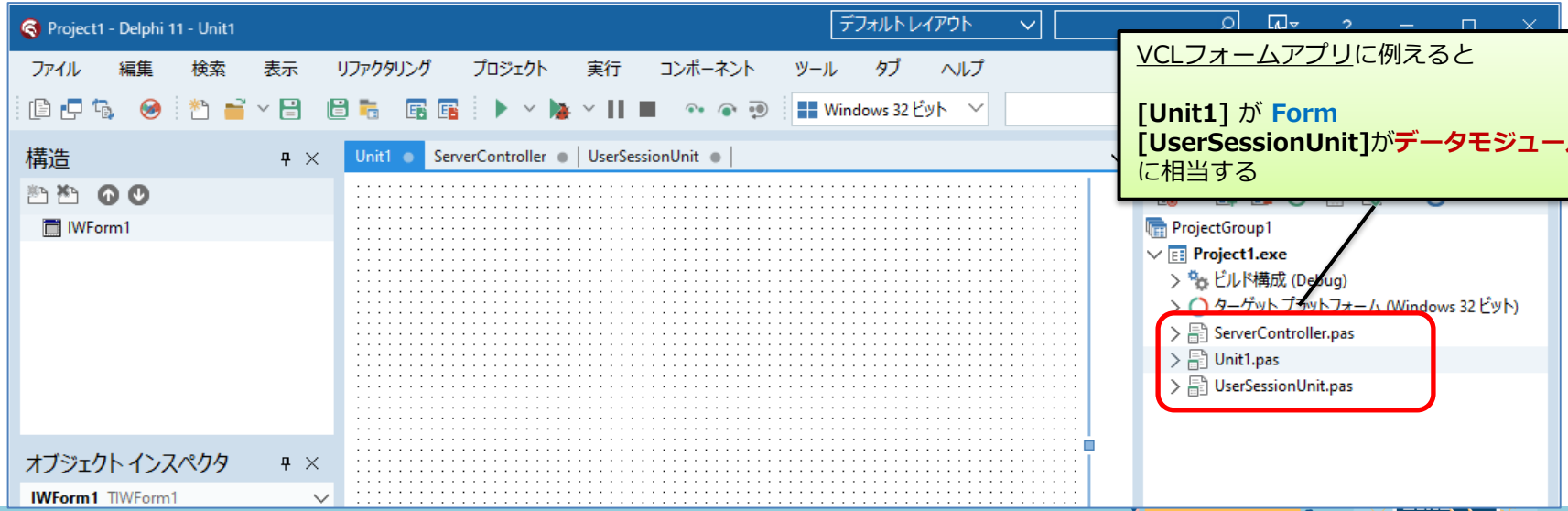


Webサーバー上で動作するアプリについて、

WebBroker では、**CGIアプリ**も作成できるが、
IntraWeb では、**ISAPIアプリ**のみが作成可能
（セッション管理の仕組みが用意されている）

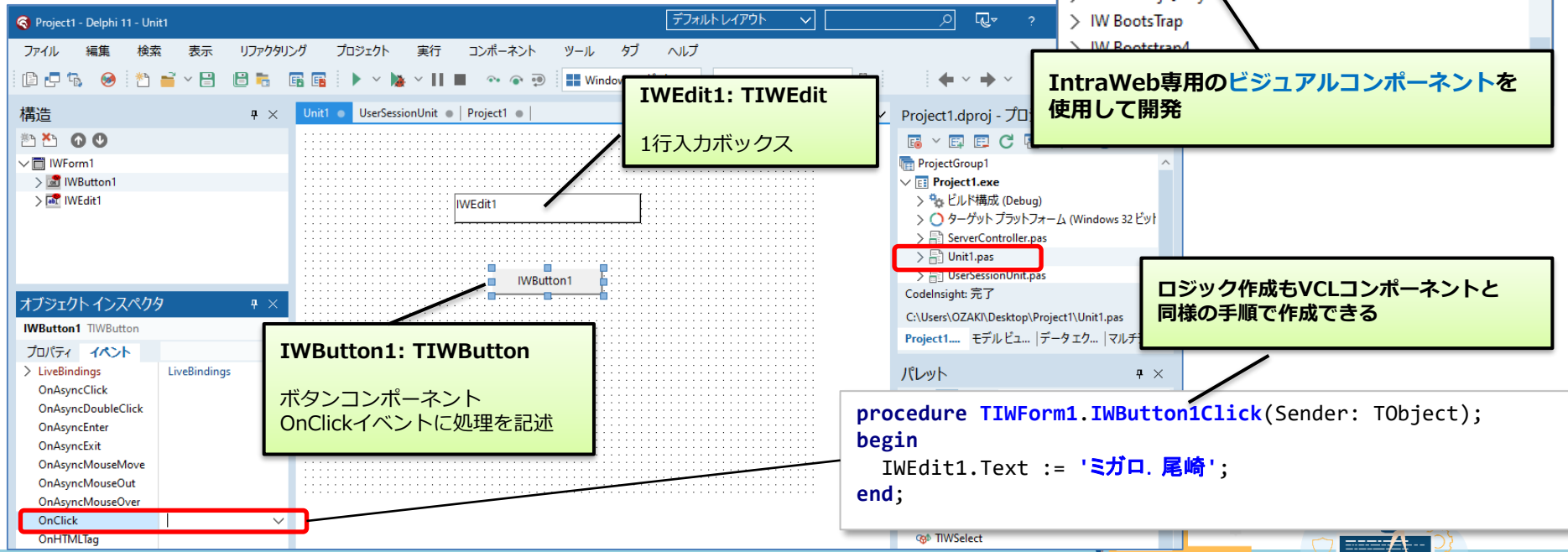
IntraWeb作成手順

- IntraWebプロジェクトの新規作成
  - 3つのユニットが自動生成される
    - **ServerController.pas** ： Webアプリ全体の設定情報をプロパティで定義
    - **Unit1.pas** ： **Webアプリの画面設計を行うユニット**
    - **UserSessionUnit.pas** ： **実行するブラウザ毎のセッション情報を管理するユニット**



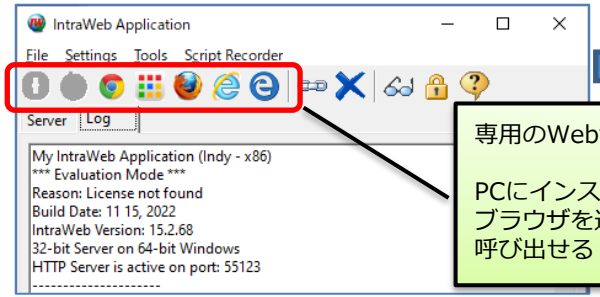
IntraWeb作成手順

- IWForm1(Unit1)
  - 画面レイアウトをVCLフォーム同様の手法で作成可能
  - ロジック作成も、VCLフォーム同様のイベント作成による開発が可能



IntraWeb作成手順

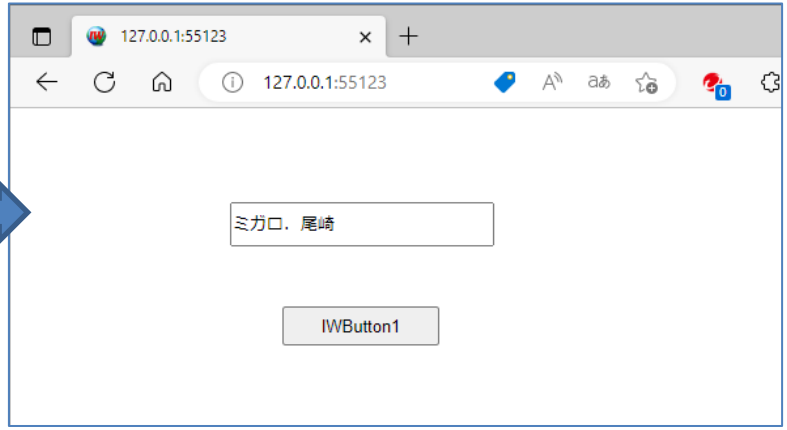
- IntraWebアプリケーションの実行
  - スタンドアロンアプリの場合、[実行]ボタンをクリック



専用のWebサーバーが起動

PCにインストールされている
ブラウザを選択すると、Webアプリを
呼び出せる

実行例



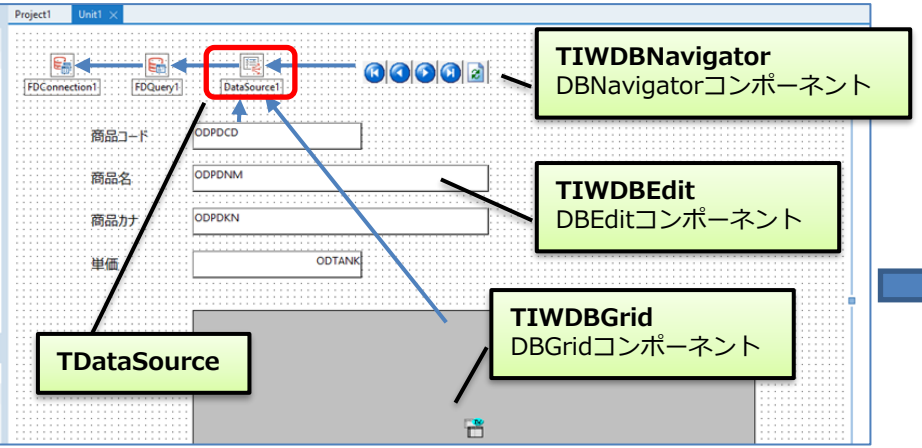
- IntraWeb Standardコンポーネント
  - VCLコンポーネントと類似の名称のものが多く、直感的に使用できる

主な
コンポーネント

VCL	IntraWeb	VCL	IntraWeb
TButton	**TIWButton**	TComboBox	**TIWComboBox**
TEdit	**TIWEdit**	TRadioGroup	**TIWRadioGroup**
TMemo	**TIWMemo**	TListBox	**TIWListBox**
TLabel	**TIWLabel**	TPanal	**TIWRegion**
TCheckBox	**TIWCheckBox**	TStringGrid	**TIWGrid**

IntraWeb作成手順

- IntraWeb Dataコンポーネント
  - データベース連結コンポーネントも使用可能



実行例

主な
DBコンポーネント

VCL	IntraWeb
TDBGrid	TIWDBGrid
TDBEdit	TIWDBEdit
TDBMemo	TIWDBMemo
TDBText	TIWDBLabel
TDBCheckBox	TIWDBCheckBox
TDBComboBox	TIWDBComboBox
TDBRadioGroup	TIWDBRadioGroup
TDBListBox	TIWDBListBox
TDBLookupComboBox	TIWDBLookupComboBox
TDBNavigator	TIWDBNavigator

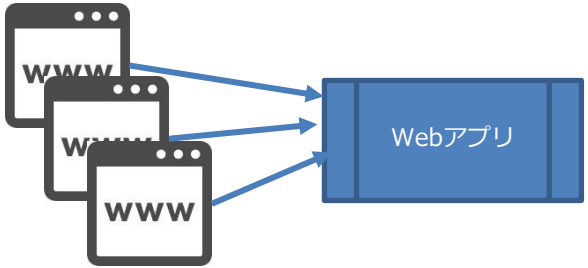
IntraWebアプリ開発のポイント

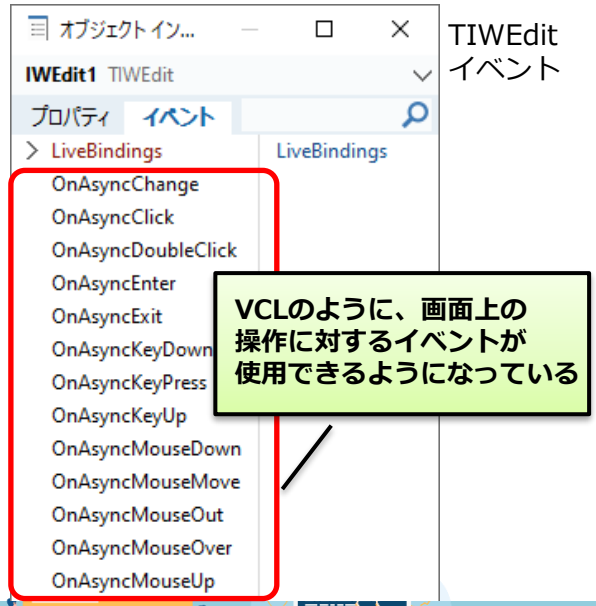
- セッション管理
  - IntraWebアプリは、1つのプロセスを
    複数セッションで共有して使用

  ⇒ グローバル変数やDB処理等は、セッション単位で
    管理が必要な為、 **UserSessionUnit**に定義する。

- OnAsync～イベントの活用 (Ajax)
  - Webアプリは、通常リクエストとレスポンスの繰り返しで
    実行されるが、常にページ全体の書き換えが必要となる

  ⇒ **OnAsync～イベント**を使用すると、
    ブラウザは画面の一部だけをWebサーバーと
    通信して更新する事ができ、よりデスクトップ
    アプリに近い操作感を実現可能





TIWEdit
イベント

IntraWeb関連情報

- Atozed Software社公式サイト
  - ドキュメントページ　：　https://www.atozed.com/intraweb/docs/
  - サンプルプログラム集　：　https://github.com/Atozed/IntraWeb
- ミガロ．テクニカルセミナー
  - これまでに実施したIntraWebに関するセッション

実施回	セッションタイトル	概要
第2回	開発の幅を拡げましょう！　Delphi/400で始めるWEBアプリ入門	Webアプリの基本とIntraWebの開発手法
第3回	知って得する！現役ヘルプデスクが答える - Delphiテクニカルエッセンス 3.5	Cookieについて、開発モードの切替手順
第4回	Delphi/400開発ノウハウお教えします ー　現場で培った開発手法公開	JavaScriptとの連携、グラフ出力
第11回	Delphi/400で本格Webアプリ開発	動的な明細行作成、ポップアップ表示、HTMLとの連携
第18回	開発者が知りたい実践プログラミングテクニック！～ 明日から使えるテクニック集 ～	jQueryを組み合わせたスマートデバイス向け最適化方法
第19回	Delphi/400 テクニカルセッション　-　開発者が知りたい実践プログラミングテクニック！	セッションタイムアウト制御

■ まとめ

- Delphi/400による2つのWebアプリ開発手法
  - **WebBroker**
    - 非ビジュアルコンポーネントを使用したDelphi標準のフレームワーク
    - 画面（ユーザーインターフェース）の設計には、HTMLの作成が必要。
  - **IntraWeb**
    - Webアプリ開発をVCL同様ビジュアルコンポーネントで実現できるアドインフレームワーク
    - セッション管理やAjaxが容易に実装出来き、本格的なWebアプリ開発にも適している。

  - 本セッション内のサンプルプログラム
    - 下記よりダウンロード可能

      https://www.migaro.co.jp/temp/WebSeminarDoc/Tecsem27/202212_2.zip

ご清聴ありがとうございました。