佐田 雄一

株式会社ミガロ.

システム事業部 システム1課

# Webコンポーネントのカスタマイズ入門

カスタムコンポーネントを開発できれば、さらなる Web 開発の効率化が見込める。 VCL for the Web ならではのカスタムコンポーネント作成手法を述べる。

●はじめに ●カスタムコンポーネントを利用するメリット ●Ajax・JavaScript との連携 ●数値専用 WebEdit の作成例 ●まとめ



略歴 1985 年 12 月 6 日生 2009 年甲南大学経営学部卒 2009 年 04 月株式会社ミガロ.入社 2009 年 04 月システム事業部配属

**現在の仕事内容** Delphi/400 を利用したシステム開 発や保守作業を担当。Delphi およ び Delphi/400 のスペシャリストを 目指して精進している。

# 1.はじめに

「VCL for the Web (IntraWeb)」は Delphi/400の開発機能の1つである。 この機能を用いると、C/S (クライアン ト/サーバー型)アプリケーションの開 発と同様の手法で、Web アプリケーショ ンを作成することができる。

この VCL for the Web を活用するこ とで、Web 開発に馴染みの薄い開発者 であっても、使い慣れた Delphi/400 を 用いて、容易に Web アプリケーション を作成することが可能になるだろう。

本稿では、Delphi/400を使用した Webアプリケーション開発のバリエー ションをさらに広げていただけるよう、 特にコンポーネントのカスタマイズ手法 について取り上げ、くわしく紹介する。

# 2.カスタムコンポーネント を利用するメリット

Delphi/400 では、開発機能の1つで ある "継承" を活用して、既存のコンポー ネントから新しいコンポーネントを作成 することができる。【図 1】

このようにして作成されたコンポーネ ントは「カスタムコンポーネント」と呼 ばれ、要件にあわせてさまざまな追加機 能を組み込むことができる。

本稿では、VCL for the Web ならで はのカスタムコンポーネント作成手法に ついて述べる。

なお、C/Sアプリケーションの開発 におけるカスタムコンポーネントの作成 手法については、『ミガロ.テクニカル レポート 2012』に、「カスタマイズコン ポーネント入門~ Delphi/400 開発効率 向上」を掲載している。わかりやすく解 説されているので、ぜひ参考にしていた だきたい。

# 3.Ajax·JavaScript との連携

VCL for the Web 専用コンポーネン トの特徴の1つとして、Ajax と連携し た Async イベントが利用できる点が挙 げられる。

・Ajax

Ajax とは「Asynchronous JavaScript + XML」の略で、JavaScript によって 実現する機能の総称である。Ajax の仕 組みを作成することは難しいが、Delphi ではイベントで提供されているため、簡 単に利用することが可能である。

例えば、C/S アプリケーションの開 発においては、TEdit でフォーカスが抜 けた際に OnExit イベントが実行され る。一方、VCL for the Web アプリケー ションの TIWEdit に おいて は、 OnAsyncExit イベントが実行される。 このイベントでは、HTML 上でフォー カスが抜けたことを JavaScript で検知 し、非同期で処理を行うことが可能であ る。【図 2】

#### ・Async イベント

C/S アプリケーションにおけるイベン トの多くは、VCL for the Web アプリ ケーションでは Async イベントで代替

図1						
 図1 コンポーネント継承の流れ						
 TIWEdit TIWXXXEdit						
 ※標準のEditコンボーネント ※カスタムコンボーネント						
 図2 OnAsyncexit時1~名称を取得9 る例 ユーザーID 0086						
 コートのEditがらフォーカスが離れると、 AsyncExitで名称を取得する例 この際、画面の再読込は行われず、						
ユーザーID 0086 佐田 雄一 動的にHTMLソースが変更される						
図3						
 図3 コンポーネントを新規作成する						
 新規作成     「     新規作成     「     ローローローローローローローローローローローローローローローローローロ						
 □ IntraWeb FireMonkey FireMonkey 7ォーム MSBuild ターゲットフ VCL Metropolis UI フ □ Web サービス Metropolis UI 7 アイル オーム						
 C WebSnap     Profiling     VOL フォーム     VOL フレーム     エンボーネント     スレッドオブジェクト						
- ご その他のファイル     ・ ユニットテスト     ・ エニットテスト						
データモジュール ユニット						
 OK キャンセル ヘルプ						
 図4						
 図4 フレームワークを選択する(Version XE3~)						
ーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーー						
VCL for Delphi Win32(W)						
○ FireMonkey for Delphi(E) VerXE3では VCL for Delphi Win32」を選択						
 <<戻る(B)						
	50					

されている。加えて、これらは Ajax の 処理に依存しているため、中には実行タ イミングや処理の種類が異なるものも存 在する。

例えば、C/Sアプリケーションにお ける TEdit の OnChange イベントは、 1 文字でも値が変更されたタイミングで 実行される。それに対し、TIWEdit の OnAsyncChange イベントでは、フォー カスが抜けたタイミングで、値が変更さ れていれば実行される。

#### ・JavaScript 連携

VCL for the Web アプリケーション のコンポーネントは、JavaScript と密 接な関係がある。コンポーネントに JavaScript を埋め込み、キー押下時や フォーカスセット時などの制御を JavaScript で行うことが可能である。

コンポーネントのカスタマイズにおい て、実際に埋め込み、制御を行う手法に ついては、次章でコンポーネントを作成 しながら解説する。

※ Asynchronous = 非同期

# 4.数値専用WebEdit の作成例

カスタムコンポーネントを作成するこ とで、処理の共通化を実現し、開発の効 率化を図ることが可能になる。VCL for the Web のカスタムコンポーネントも 同様であり、その特徴は前述の通りであ る。

ここからは、実際に VCL for the Web アプリケーション用のカスタムコ ンポーネントを作成していこう。

今回は例として、TIWEditを継承し て「TIWNumEdit」という数値入力専 用のWebEditを作成する。前述の AsyncイベントおよびJavaScriptを活 用することで、C/Sアプリケーション のコンポーネントと同等の制御をWeb アプリケーションでも実現できるだろ う。

## (1)コンポーネントの新規作成

C/S アプリケーションで作成する際 と同様に、コンポーネントソースを新規 作成する。手順に沿って説明していこう。 なお、今回の開発環境は Delphi/400 Version XE3 を使用している。

#### ①コンポーネントの新規作成

メニューの [ファイル | 新規作成 | そ の他] を選択し、新規作成ウィンドウよ り「コンポーネント」を選択する。【図 3】

②フレームワークの指定(Version XE3~)
 フレームワーク指定のダイアログでは、VCL for Delphi Win32を選択する。
 【図 4】

③継承元コンポーネントの指定

今回は TIWEdit を継承するため、 TIWEdit を指定する。【図 5】

④コンポーネント名とパレットページ、保存先の指定

クラス名に、今回作成するコンポーネ ント名である TIWNumEdit を指定す る。

パレットページ名に、新規追加するコ ンポーネントパレットのパレットページ 名を指定する。なお、今回はデフォルト 値の Samples を使用する。

ユニット名に、今回追加するカスタム コンポーネントの pas ファイル名、お よび保存先を指定する。なお、今回は保 存先を C:¥Projects¥TechRep¥Sample NumEdit.pas と設定する。【図 6】

⑤コンポーネントの新規作成の完了
 完了すると、図7の画面が表示される。
 【図7】

## (2) 宣言部の作成

今回の数値入力用 WebEdit では、以 下の機能を実装したいと思う。

- ・Value:Edit 値を数値として設定/保 持するための新規プロパティ。
- Text:既存のTextから変更し、読み 取り専用プロパティにする。
- JavaScript で不必要なキー押下を無 効にする。
- ・フォーカスが抜けた時に、数値でない 値がセットされていれば色を変更する。

これらを実現するため、ソース1のよ うに宣言部を記述する。それぞれのプロ パティやイベントの詳細については後述 する。【ソース 1】

#### 1 uses

HookEvents イベントの実装に必要な IWRenderContext、IWScriptEvents、 色の指定を行うために必要な Graphics を宣言部の uses に追加する。

#### ② private 宣言

入力された数値を内部で保管するため の変数FValueをprivate部に宣言する。 またSetValue、GetValue、GetText に ついては後で自動生成されるため、この 時点では記載不要である。

#### ③ protected 宣言

ScriptEvents の設定を行うための HookEvents イベント、およびフォーカ スが抜けた際に実行する EditExit イベ ントを protected 部に宣言する。また、 HookEvents は上位クラスから継承され ているため、宣言の後ろに override; を 記述する。

#### ④ public 宣言

クラス生成時イベント(constructor) を使用するため、public 部に宣言する。

#### ⑤ published 宣言

実際に画面に貼りつけた際にオブジェ クトインスペクタに表示される変数 Value を宣言する。また、Text は上位 クラスから存在するが、write を指定し ない (read のみとする) ことで、読み 取り専用のプロパティになる。

各宣言部が完成したら、Ctrl + Shift + [C] キーを同時に押下することで実 装部、および先述したSetValue、 GetValue、GetTextが自動生成される。

## (3) Createイベントの作成

Create は、コンポーネントの生成時 に実行されるイベントである。

画面にコンポーネントを配置した際 や、ソース内で動的にコンポーネントを 生成した際のプロパティの初期値は、こ こで設定した値となる。

記述例については、ソース2を参考に していただきたい。これらの中で注意が

図5 継承元コンポーキ	シトを指定する			
③ コンポーネントの新規	作成	×		
碓 承 元 コンボーネント     このコンボーネントの継承	《元コンボーネントを選択し	ます。		
			_	
		Q、 検索		
	コンポーネント名	ユニット名 ^		
	TIWDBNavigator TIWDBRadioGroup	IWDBStdCtrls IWDBExtCtrls		
	TIWDBText TIWDynamicImage	IWDBStdCtrls IWCompExtCtrls		
17	TIWEdit TIWFile	IWCompEdit IWCompFile		
	TIWFileUploader TIWFlash	IWCompFileUploader IWCompFlash		
	TIWGradButton TIWGrid	IWCompGradButton IWCompGrids		
	TIWHRule TTWHTMI 40Container	IWHTMLControls IWHTML40Container		
<< 戻る	(B) 次へ(N) >> )	完了 キャンセル ヘルプ	]	
6				
図らったポーネントタ	ل شاريد الم	<sup>皮</sup> ち生た指定す Z		
	とパレットペーン、1	本行元を相定りる		
コンボーネント	1FAX			
新規コンボーネントの名	前とユニットの名前の選択。	l.o		
				_
10	クラス名(C):	TIWNumEdit	コンポーネントクラス名を指定	
	ー パレットページ名(P):	Samples 🔷	インストール時に配置する	
	ユニット名似:	C:¥Projects¥TechRep¥SampleNumEdit	リースファイルの保存失指定	
	検索パス( <u>5</u> ):			
Z				
8-8				
<< 5	<b>3個</b> 次へ110 >>	完了 キャンセル ヘルブ		
7				
1				
図7 コンポーネントソ·	-ス新規作成完了 pleNumEdit	時のソース		
1 unit SampleN	umEdit;			
interface				
System.Sys IWBaseCont	Utils, System.C <sup>.</sup> rol, IWBaseHTML(	lasses, Vcl.Controls, IWVCLBas Control, IWControl, IWCompEdit	eControl,	
type 10 m TIWNumEdit	= class(TIWEd);	+ )]	E	
Private (Privat	e 宣言)	クラス継承		
Protected	ted 宣言 ]			
Public published	<i>宣言 ]</i>			
end;	ned 宣言 /			
20 1	gister;	コンポーネントパレットのパレット	~	
Procedure Re				
20 - procedure Re - implementati	on gister:			
2U → procedure Re → implementati → procedure Re → procedure Re bagin RegisterCo	on gister; nponents('Sample	es', [TIWNumEdit]);		

必要なプロパティやイベントは、次の2 点である。【ソース2】

#### Font.FontName

Create イベント内でフォント名を指 定する場合、「MS ゴシック」のように 日本語が入っていると正しく認識しない ため、英語で指定する。

#### • EditExit

OnAsyncExit イベントに EditExit を代入することで、フォーカスが抜けた 際に EditExit イベントが実行されるよ うになる。このため、EditExit イベン トの引数は OnAsyncExit と同じものを 設定しておく。

#### (4) HookEventsイベントの コーディング

コンポーネント内で JavaScript を利 用したイベントを実行させることができ る ScriptEvents プロパティを利用して、 キー押下時に、数値入力と無関係のキー 押下を無効にする処理を実装する。

これを実現するため、HookEvents イ ベントにおいて、ScriptEvents プロパ ティに対して JavaScript を設定する。 なお、本稿では JavaScript 言語の詳細 説明については割愛するが、参考文献や 解説している Web サイトが多く存在す るので、比較的修得がしやすい言語であ る。

ソースの記述については、ソース3を 参考にしていただきたい。【ソース3】

#### (5) ValueとTextの読み取り/ 書き出し時イベントの設定

今回は Edit の値を数値として保持さ せるため、既存の Text を読み取り専用 プロパティに変更し、新たに数値型 (Double)の Value プロパティを作成 して値の入出力を行う。

記述例については、ソース4を参考に していただきたい。【ソース4】

#### GetText

GetText は Text の値を読み取るため のイベントで、他のクラスから Text を 呼び出すと、このイベントを通して値が 渡される。Text ではなく inherited Text と記述することで、画面にセット されている文字列をそのまま返すことが できる。

#### GetValue

GetValue は Value の値を読み取るた めのイベントで、GetText と同様に他 のクラスから Value を呼び出すと、こ のイベントを通して値が渡される。

#### SetValue

SetValue は逆に Value に値を書き込 むためのイベントで、内部保管用の FValue 変数および画面の Text にセッ トされた数値を代入する。

## (6) EditExitイベントのコーディング

EditExit 手続きを使用することで、 Editからフォーカスが抜けた時に、 Textに入力した文字列を数値に変換し て内部保管値に代入する処理を行う。

その際、コピー&ペースト等によって 数値でない文字列がセットされた状態で フォーカスが抜けた場合には、背景色を 赤に変更し、内部保管値に初期値(0) を代入する。

また、数値の判定に成功した際は、背 景色をもとに戻す処理を記述している。 ただし、無色を示す clNone を BGColor プロパティにセットしても色の変更が行 われないため、他の色(今回の場合ウィ ンドウ色)を指定する必要がある。

ソース記述については、ソース5を参 考にしていただきたい。【ソース5】 以上で、TIWNumEdit コンポーネン トのコーディングは完了である。

### (7)インストールと画面への実装

作成したコンポーネントを実際に使用 するには、パッケージを新規作成してイ ンストールを行う。手順に沿って説明し よう。

#### ①パッケージを新規作成する

メニューの [ファイル | 新規作成 | そ の他] を選択し、新規作成ウィンドウよ り「パッケージ」を選択する。【図 8】 パッケージを作成したら、プロジェク トを任意の提示に欠前を付けて促在す

トを任意の場所に名前を付けて保存す る。(今回作成したユニットと同じディ レクトリ内が望ましい。) 今回は SamplePackage.bplとして保存する。

なお、既存のパッケージに今回作成し たユニットを追加する場合は、ここの手 順は不要である。

#### ②パッケージにユニットを追加

次に、メニューの [プロジェクト | プ ロジェクトに追加 ...]を選択し、今回作 成したユニットをパッケージに追加す る。【図 9】

#### ③インストール

図9のプロジェクトマネージャ内にあ る SamplePackage.bpl を右クリックし、 再構築(ビルド)を行う。

再構築が正常に行えたことを確認した ら、再度 SamplePackage.bpl を右クリッ クし、「インストール」を選択すると、 コンポーネントのインストールが実行さ れる。

インストールが正常に完了すると、ダ イアログが表示される。【図 10】

#### ④画面への組み込み

インストールが完了したら、VCL for the Web の新規プロジェクトを作成す ると、デザイン画面のツールパレットに TIWNumEdit が追加される。【図 11】

TIWNumEdit を選択して画面内をク リックすると、図 12 のような項目が配 置 される。見た目はカスタム前の TIWEdit と異なっているが、TIWEdit と同じようにプロパティやイベントの設 定を行うことが可能である。【図 12】

⑤ライブラリパス設定と実行

作成したカスタムコンポーネントを使 用したプロジェクトのコンパイルが通る ようにするには、Delphi/400 開発環境 のライブラリパスの設定を行う必要があ る。

メニューの [ツール | オプション] を 選択し、今回作成した SampleNumEdit. pas の保管先ディレクトリをライブラリ パスの一覧に追加する。【図 13】

ここまでの手順で、今回作成した TIWNumEditを使用する準備はすべて 完了である。

アプリケーションにコンポーネントを 組み込んでコンパイルすれば、 TIWNumEditの機能動作を確認するこ

ノーン	
	ソース1 宣言部の記述
	unit SampleNumEdit;
	uses
	System.SysUtils, System.Classes, Vcl.Controls, IWVCLBaseControl, IWRaseControl. IWRaseHIMIControl. IWControl, IWCompEdit,
	type
	TIWNumEdit = class(TIWEdit) HookEventsで使用するため、 private usesに追加 usesに追加
	iP//Valce ヨョー/ FValue: Double; procedure SetValue(const Value: Double); // Yalueに書き込む時の処理
	function GetValue: Double; // Valueが読み取られる時の処理 function GetText: String; // Textが読み取られる時の処理
	/ <i>Protected 宣言 )</i> procedure HookEvents(AContext: TIWPageContext40;
	<u>AScriptEvents: TIWScriptEvents): override:</u> // <i>JavaScriptの設定</i> procedure EditExit(Sender: TObject; EventParams: TStringList); nublic
	( <i>Public 宣言 )</i> <u>constructor Create(AOwner: TComponent); override;</u> // コンボーネント生成時処理
	published <u>(Published 宣言)</u> 「property Value: Double read GetValue write SetValue: // 教値として値保持
	property Text: String read GetText; // 競拝のTextを読み取り専用にする end;
L	writeを指定しないことで、 Textが読み取り専用になる
ソー:	72
	ンーフッ Create イベントの記述
[	constructor TIWNumEdit.Create(AOwner: TComponent);
	inherited: Alignment := taRightJustify; // 右揃え(Version2009以降で対応)
	Font.FontName := 'MS GOTHIC'; // フォント名(英語で指定)  Font.Size := 10: // フォントダイズ
1	QnAsyncExit := EditExit: // QnAsyncExitのなイミングでEditExitを実行
	OnAsyncExit     ニ EditExit:     // OnAsyncExitのタイミングでEditExitを実行       Self-BGColor     ニ Cl Window:     // 資景色の初期値(ウインドウ色)       FValue     ニ O;     // 実数値の初期値
	OnAsyncExit       := ÉditExit;       // OnAsyncExitのタイミングでEditExitを実行         Self.BGColor       := [I]indow;       // 賞景色の初期値(ウインドウ色)         FValue       := 0;       // 画面に表示する文字列の初期値         end;
	OnAsyncExit       := clititxit;       // OnAsyncExitのタイミングでEditExitを実行         Self-BGColor       := clitindow;       // 賞意色の初期値(ウインドウ色)         FValue       := 0;       // 実数値の初期値         inherited Text       := '0';       // 画面に表示する文字列の初期値         end;
	OnAsyncExit       := ÉditExit;       // OnAsyncExitのタイミングでEditExitを実行         Self-BGColor       := clWindow;       // 賞曼の初期値(ウインドウ色)         FValue       := 0;       // 運動に表示する文字列の初期値         end;       // 画面に表示する文字列の初期値
	OnAsyncExit := ÉditExit; // OnAsyncExitのタイミングでEditExitを実行 Self.BGColor := clWindow; // 賞景色の初期値 FValue := 0; // 実数値の初期値 inherited Text := '0'; // 画面に表示する文字列の初期値 end;
	OnAsyncExit     := ÉditExit:       Self-BGColor     := cl\#indow:       // 賞曼色の初期値     // 賞曼色の初期値       Inherited Text     := '0';       // 画面に表示する文字列の初期値
) ソー:	OnAsyncExit     ジョーを見付した       Self-BGColor     ジョーのは       「Yalue     ジョーのは       Inherited Text     ジョーのは       Image: Self-BGColor     Image: Self-BGColor       Image: Self-BGColor     Image: Self
ן יע-:	OnAsyncExit       := EditExit:       // OnAsyncExitのタイミングでEditExitを実行         Self+.BGColor       := Cl\\\  # そのの初端(ウインドウ色)         // 定意のの初端(ウインドウ色)       // 定意塗(のの初端)         inherited Text       := '0';         // 画面に表示する文字列の初期値         end;         X3         ソース3 HookEventsイベントの記述         procedure TI\\NumEdit.HookEvents(AContext: TI\\PageContext40; AScriptEvents);
 : ערבי	OndernetXit       ジョーを見付tEXiti         Self+BGColor       ジョーにし;         (アValue       ジョーの;         (mherited Text := '0';       // 資意色の初期値 (ウインドウ色)         (原義値の初期値)       // 演意のの期値         (mherited Text := '0';       // 画面に表示する文字列の初期値         end;       ()       ()         73       ()       ()         Y-A3 HookEventsイベントの記述       ()         procedure TIWNumEdit.HookEvents(AContext: TIWPageContext40;       ()         AScriptEvents: ITWScriptEvents);       ()         const       (/)       ()         (/)       ()       ()         (/)       ()       ()         ()       ()       ()         ()       ()       ()         ()       ()       ()         ()       ()       ()         ()       ()       ()         ()       ()       ()         ()       ()       ()         ()       ()       ()         ()       ()       ()         ()       ()       ()         ()       ()       ()         ()       ()       ()         ()       () <td< td=""></td<>
 	OnAsyncExit := collExit: 少くでEditExitを実行         Self-BGOolor := collFindow:         // 資景色の初期値(ウインドウ島)         // 演員合の初期値         inherited Text := '0':         // 画面に表示する文字列の初期値         end:             Z3             Y-Z3 HookEventsイベントの記述             Procedure TINumEdit-HookEvents(Acontext: TIWPageContext40;         AScriptEvents: ItWscriptEvents);         const         // のからいたまでは、         // のからいたまでは、         // のからいたまでは、         // のからいためには、         // のからいためには         // のからいた         // のからいためには         // パーの         // のからの         // のからいためには         // のからの         // のからの         // のからいためには         // のからいためには         // のからの         // のから
	Y-A3 HookEventsイベントの記述 procedure TIWNumEdit.HookEvents(AContext: TIWPageContext40; ASCriptEvents: IWScriptEvents); const         Y-A3 HookEventsイベントの記述 procedure TIWNumEdit.HookEvents(AContext: TIWPageContext40; ASCriptEvents: IWScriptEvents); const         // の格型の構成         // の格型の構成         // 回面に表示する文字列の初期後
 ソー:	Ondesyndefxit := EditExit: Salf_BG0lor := clWindow: // ア書色の初解値(ウインドウ色) // 変数値の初解値 // 面面に表示する文字列の初期値             Z3             Z3             Y-Z3 HookEvents/ベントの記述             Procedure TIWNuEGit.HookEvents(AContext: TIWPageContext40; AScriptEvents: TIWScriptEvents);         const             // のれるvnoExitののまかっとした。             // スカロシスティントの記述             // アロスの部分値(ウインドウ色)             // アロスのをした。             // アロスの部分値             // スカロシスティントの記述             Procedure TIWNuEGit.HookEvents(AContext: TIWPageContext40;             AScriptEvents: TIWScriptEvents);         const             // ノカの能なアののスクリプトイベント         // ノカの能なアのが用を指定する回数         script=* = String.fromCharGode(event.keyCode);             ·/ アelurn true;             ·/ * return true;
; ソー;	Image: Solar := cl\#indow:       // OnAsyncExit@タイミングでEditExite要行
; ソー;	Image: Self. Book Contract Content in the second seco
; //—;	ImagenicExit       ImagenicExitOge       シックでEgitExitExite要行 度合の初期値(ウインドウ色) // 変数値の初期値         Fyalue       ImagenicExitOge       シックでEgitExitExitE         end;       ImagenicExitOge       シックでEgitExitExitE         Procedure       ImagenicExitOge       シックでEgitExitExitE         Procedure       ITWEGItExitExitE       ImagenicExitOge       シックでEgitExitExitExitExitExitExitExitExitExitEx
; ע;	Image: Self 1: Bdo: Solor :::::::::::::::::::::::::::::::::::
; y;	Impaired a synce field text i: ::::::::::::::::::::::::::::::::::
; ソー;	Imperiate Section       Imperiate Section       Imperiate Section       Imperiate Section         Y=Z3       HockEvents       Imperiate Section       Imperiate Section       Imperiate Section         end:       Imperiate Section       Imperiate Section       Imperiate Section       Imperiate Section         Y=Z3       HockEvents       HockEvents       Imperiate Section       Imperiate Section         end:       Imperiate Section       Imperiate Section       Imperiate Section       Imperiate Section         Y=Z3       HockEvents       HockEvents       Imperiate Section       Imperiate Section       Imperiate Section         procedure       IIMPERIate Section       Imperiate Section       Imperiate Section       Imperiate Section         procedure       IIMPERiate Section       Imperiate Section       Imperiate Section       Imperiate Section         fill       Cold       Section       Section       Imperiate Section       Imperiate Section         fill       Section       Section       Section       Imperiate Section       Imperiate Section         fill       HockEvents       For Section       Imperiate Section       Imperiate Section       Imperiate Section         fill       Section       Imperiate Section       Imperiate Section       Imperiate
; y;	Imperiation of the synces
; ソー;	Imperiation of the second state second state of the second state of the se
; ;	ZA3         Y-X3 HookEvents/(X) - is c [4] index::
; ソー;	Total control is a control is control is control is
; ;	Total sector is a constrained of the sector is consector is a constrained of the sector is a
; //—;	Outson 25 (I       I I I I I I I I I I I I I I I I I I I

# 5.まとめ

近年、Web アプリケーションやスマー トデバイス用アプリケーションへの需要 が高まり、C/S アプリケーションだけ ではなく、Web アプリケーションでの システム構築が多くなってきている。

今回は Web のカスタムコンポーネン ト開発手法を説明したが、カスタムコン ポーネントを開発できるようになれば、 さらなる Web 開発の効率化が見込める だろう。

Web アプリケーション開発に難しさ を感じておられる開発者の皆様にとっ て、本稿が新たなアプリケーション開発 の一歩となる内容であれば幸いである。

Μ

ソース4			
ソース4 GetText/GetValue/	SetValueの記述	他のユニット等から「Text」プロパティが	
function TIWNumEdit.GetT begin Result := inherited Te	ext: String; xt: // 画面のJextの文字値を渡す	参照されると、この関数を通じて値を返す	-
end;		他のユニット等から「Value」プロパティが	
begin Result := FValue; // 4	alue: Double; 実際に値を保管している変数FValueの	参照されると、この関数を通じて値を返す の <i>値を返す</i>	
end;	Value(appet Value: Dauble):	他のユニット等から「Value」プロパティに	-
inherited Text := Floa	tToStr(Value); // 画面のTextに代 変形である年間、イリス変換FValuat	値が代入されると、この関数を通じてセットする え され た 値 を 表示	
end;			-
			-
			-
V-75			
	<b>ب</b>		-
ソース5 EditExit手続きの記 <u>procedure TIWNumEd</u> it.Edi	亚 tExit(Sender: TObject; EventPara	ams: TStringList);	
var dCheck: Double; // 値。	<i>变換用変数</i>	TryStrToFloat関数は、実数変換成功時は	-
if (not TryStrToFloat)	(inherited Text, dCheck)) then <	」 Trueが返り、dCheckに変換値が代入される 変換失敗時はFalseが返る	-
Self.BGColor := clRe FValue := 0:	ed; // 入力値が数値でない場合。 // 実際に値を保管している#	、 背景色を赤に変更 変数F/a/weiこのを代入	
end else	,,		-
Self.BGColor := clWi FValue := dCheck;	ndow; // 入力値が数値の場合、背影 // 実際に値を保管している影	景色をウインドウ色に変更 変数FValueに入力値を代入	
end; end;			-
			-
			-
আও			
			-
図8 パッケージの新規作成			
S 新式売TFDX	Q. 検索	×	-
ActiveX			-
Delphi ファイル ロ IntraWeb	ビー 「「」」 パネル サービス アプリケー	「International Section Control Section Contr	
·····ピコ Web サービス ·····ピコ WebBroker	アプリケーション ション Me	letropolis UI ア ップアグリケーション	-
WebSnap			
Profiling Web ドキュメント	MDI アプリケーション SDI アプリケーション VCI	1. Metropolis UI ア VCL フォーム アプリケ プリケーション ーション	-
		n 🚔	-
	コンソール アプリケー ダイナミックリンクライ	パッケージ ソース DLL ウィザー	
			-
			-
			-
	専用のセッションでユーザーインターフェイスなし ンを作成します。サービスは、サーバー上で使用	ババックグラウンドで動作する Windows サービス アブリケーショ 目する場合や、コンピュータ ユーザーの介入なしにプロセスが動	
	11年する必要がある場合に通しています。		
		OK キャンセル ヘルプ	-
			-
			-



