

【B2】Delphi テクニカルセッション 「見た目で楽しい Delphi プログラ ミング」

会社名 株式会社シリアルゲームズ 名前 細川 淳



- VCL / FireMonkey のスタイルの触りを紹介します
- VCL スタイル

 TStyleManager
- FireMonkey スタイル
 - TStyleBook





VCL スタイル

VCL スタイル

- Delphi XE 以前
 - TThemeServices
 - OS が提供する「テーマ」を利用するクラス
 - Windows XP Style (Luna など)に対応するために追加されました
 - ※XE2 以降でも依然として利用可能
- Delphi XE2 以降
 - TStyleManager
 - VCL / FireMoneky に新しく追加された「スタイル」を利用するクラス
 - スタイルを利用すると、OSのビジュアル(見た目)から解放されます
 →特にマルチプラットフォームである FireMonkey において重要



VCL スタイルの一番簡単な使い方





VCL スタイルを 適用する デモンストレーション



VCL スタイルを プログラムから使う

VCL スタイル - TStyleManager

- TStyleManager.TrySetStyle
 - スタイルを設定するメソッド
 - とりあえずこのメソッドさえ知っていればスタイルの適用は 可能

procedure TForm1.Button1Click(Sender: T0bject);
begin

```
TStyl eManager. TrySetStyl e('Amakrits'); // スタイルの名前
end;
```



VCL スタイル - TStyleManager

• TStyleManager.StyleNames

- 使用可能なスタイルを返すメソッド

```
procedure TForm1.Button2Click(Sender: TObject);
var
```

StyleName: String;

begi n

```
for StyleName in TStyleManager. StyleNames do
```

```
ListBox1. Items. Add(StyleName);
```

end;

```
名前をリストアップして、それを適用します
```

procedure TForm1.Button3Click(Sender: T0bject); begin

```
// SetStyle は内部で TrySetStyle を呼び出し、失敗すると例外をあげる
```

```
if (ListBox1.ItemIndex > -1) then
```

TStyl eManager. SetStyl e(Li stBox1. Items[Li stBox1. ItemIndex]); end;





VCL スタイルを プログラムから 適用する デモンストレーション



VCL スタイルを自分で 作ってみる

VCL スタイル - VCL スタイルデザイナ

自分でスタイルを作成する場合、VCL スタイルデザイナを使います。





VCL スタイル - VCL スタイルデザイナ

- スタイルは基本的に各部分部分を表す画像で提供され ます
 - ゲームのように一枚の画像からパーツを切り出して使います
- 設定項目が多い!
 - 自分のスタイルをゼロから作ろうと思っても挫折します
 - 既存のスタイルを修正していくと良いでしょう



VCL スタイル - VCL スタイルデザイナ



VCL スタイル - VCL スタイルデザイナ







VCL スタイル スタイルデザイナの デモ

VCL スタイル - 自作スタイルの使い方





VCL スタイル - 自作スタイルの使い方

TStyleManager.LoadFromFile
 スタイルファイルを読み込むメソッド

procedure TForm1. Button2Click(Sender: TObject);
begin

TStyleManager.LoadFromFile('C: ¥Temp¥TestStyle.vsf');

TStyl eManager. TrySetStyl e('TestStyl e'); //スタイルデザイナで設定した名前 end;



VCL スタイル - 自作スタイルの使い方

TStyleManager.IsValidStyle

 スタイルファイルの正当性チェック&スタイル情報取得

<pre>procedure TForm1.Button2Click(Sender: T</pre>	ſObj ect);	
Var Stal sinfer TStal sinfer		
Styrernro: IStyrernro;		
begi n		
if		
(TStyleManager.IsValidStyle('C:¥Tem	<pre>mp¥TestStyle.vsf', StyleInfo))</pre>	
then		
with StyleInfo do		
ShowMessage(
Name + sLineBreak +	Project1	
Author + sLineBreak +		
AuthorEMail + sLineBreak +	TestStyle	
AuthorIIRI + slineBreak +	HOSOKAWA Jun	
	jun@serialgames.co.jp	
version	1.0	
);		
end;	ОК	



VCL スタイル スタイルファイル 読み込み デモ



VCL スタイルを使用する コントロールの作成

VCL スタイル - コントロールの作成

ー番単純な方法は Paint メソッドや WM_NCPAINT, WM_PAINT に応えて自分で実装 します。

```
ノーマル状態のボタンを描画する例
procedure TForm1. PaintBox1Paint(Sender: T0bject);
var
 DC: HDC;
 R: TRect:
begi n
 DC := PaintBox1. Canvas. Handle:
 R := PaintBox1. ClientRect;
 StyleServices.DrawElement(
    DC.
    StyleServices. GetElementDetails(tbPushButtonNormal),
   R):
 StyleServices.DrawText(
   DC.
    StyleServices. GetElementDetails(tbPushButtonNormal),
    'TEST CAPTION'.
    R.
    TTextFormatFlags(DT_CENTER or DT_VCENTER or DT_SINGLELINE),
    StyleServices.GetStyleFontColor(sfButtonTextNormal));
end:
```

Developer Camp

VCL スタイル - コントロールの作成

- 一般的に、VCLスタイルの描画は TStyleHook を使い ます。
 - TStyleHook とは TStyleManager から呼び出されコント ロールの描画を担います。
 - WM_PAINT や WM_NCPAINT などを「フック」して、それ らの応答をコントロールに変わって実行します。
 - ただし、OS 標準スタイル時の描画には使われません
 - 標準のスタイル時は、従来と同様に、コンポーネントの Paint メソッドなどで処理されます。



VCL スタイル - TStyleHook の実例

unit uTestControl;

interface

uses

Winapi. Windows, Vcl. Controls, Vcl. ExtCtrls, Vcl. Graphics, Vcl. Themes;

type

```
TTestControl = class(TPanel)
private
class constructor Create; // クラスコンストラクタは initialization に追加するのと同じ
end;
```

implementation

type

```
TTestControlStyleHook = class(TStyleHook)
protected
  procedure Paint(Canvas: TCanvas); override;
public
  constructor Create(iWinControl: TWinControl); override;
end;
```



VCL スタイル - TStyleHook の実例

{ TTestControl }

class constructor TTestControl.Create; begin

// StyleHook とコントロールを結び付けます

TCustomStyl eEngi ne. Regi sterStyl eHook(TTestControl, TTestControl Styl eHook); end;

{ TTestControlStyleHook }

constructor TTestControl Styl eHook. Create(iWinControl: TWinControl);
begin

inherited;

Overri dePaint := True; // Paint メソッドをフックしますよ、という意味 end;

procedure TTestControlStyleHook.Paint(Canvas: TCanvas); begin

```
// TTestControl の Paint の代わりにこちらが呼ばれる
```

```
(ノーマル状態のボタンを描画する例と同様のため中略)
end;
```

end.



VCL スタイル - TStyleHook の実例

Windows 標準時	
Form2	TTestControl が継承している TPanel の標準の見た目が描画 される







VCL スタイル TStyleHook デモ

Embarcadero Developer Camp

FireMonkey スタイル

FireMonkey スタイル

- FireMonkey のスタイルは VCL スタイルとはまったく異なります。
 - 個々のコントロールがスタイルを持っている
 - TStyledControl を継承したコントロール
 - 個別に設定できる
 - スタイルは TStyleBook で管理される
 - TStyleBook で定義済みのスタイルリソースを読み込むと スタイルを一括で変更できる



FireMonkey スタイル

- 一番簡単な使用方法は…
 - Form に TStyleBook コンポーネントを貼る
 - TStyleBook コンポーネントの「Resource」プロパティをクリックしてスタイルエディタを開く
 - スタイルエディタから定義済みスタイルを読み込む
 - 「適用して終了」を押す



FireMonkey スタイル 簡単な設定方法

🐻 Unit 1	オブジェクト インスペクタ			
	StyleBook1 TStyleBook	▼		
Form1	ל אין			
	BindingName FileName Name StyleBook1			
StyleBook1	StyleName Tag 0	スタイルエディ	タを開く	
🕑 Unit1.pas				
Unit1 2 FMX.Types				
読み込み デフォルトの読み込み 上書き保存		<u>לעל</u>	適用 適用して閉	じる キャンセル
閒< 💙			ound: TRectang	₽⊙ × ₽⊙ ×
🕥 🗸 🕌 « Program Files (x86) 🕨	Embarcadero 🕨 RAD Studio 🕨 9.0 🕨 Redist 🕨 styles 🕨 f	Fmx + + Fmxの検索	Actangle	≞⊚×
整理 マ 新しいフォルダー		\$\$\$ ▼ □	olorAnimation	x
Air.Style			nerGlowEffect	× Box
× oskic∧り ↓ ダウンロード Amakrits	Style		ext	₽ ⊙ ×
AquaGrap	hite.style		ffect c Theyout	× ₽⊙×
□ Blend.Sty □ Blend.Sty □ dark.style			y/: TLayout	₽ ⊙ ×
FMX.Plat	orm.iOS.style		arStyle: TLayout	₽ ⊙ ×
Carl FMX.Platt	orm.Mac.style			
■ ドキュメント FMX.Platt	orm.Win.style			
■ ピクチャ Goldensi	ipine.scyle			
	2.Style			
Light.Sty	e			
Mac.Style	Style			
MacGraph	ite.Style 中美文ユフカ	イルた問ノ		
🗣 ネットワーク 📄 RubyGrag	hite.style 上我,所のヘア	イルを開く		
Windows	/.Style			
ファイル名(N):		FireMonkey Style	-	
		聞く(0) キャンセノ		
		-		
			スタイルの作成	
▶ ● ■ 9564:17 挿入	\コード \ デザイン / 履歴 /	N=		/
■ メッセージ				8



FireMonkey スタイル - コンポーネント毎

- 個々のコンポーネントごとに編集する方法は…
 - コンポーネントを右クリックして
 - 「カスタムスタイルの編集」「デフォルトスタイルの編集」の どちらかを選ぶ
 - スタイルエディタが開く
 - 「適用して終了」を押す

※個々のコンポーネントのスタイルを編集すると、自動的に TStyleBook コンポーネントのインスタンスが生成される



FireMonkey スタイル - コンポーネント毎

Buttion1 ● 編集(W) → コントロール(X) → LiveBinding の新規作成(L) → グリッドに合わせる(G) ● 継承元の値に戻す(I) ● 位置合わせ(A) 作成順序(N) リポジトリに追加(R) エディタで表示(V) マーススト FMX(F) カスタム スタイルの編集(Y) デフォルト スタイルの編集(Z) ●	スタイルエディタを開く	
オブジェクト インスペクタ <名前無し> TGlowEffect ・ プロパティ イベント BindingName Enabled False SGlowColor Name Opacity Softness StyleName Tag 0 Trigger IsFocused=true	♥Units ■ Linits ■ Joints ● Joints </th <th>P 通用 適用して閉じる キャンセスル ● background: TRectangle ● ○ × TRectangle ColorAnimation × TColorAnimation × TColorAnimation × TColorAnimation × TRectangle ● ○ × text: TText TGIowEffect ×</th>	P 通用 適用して閉じる キャンセスル ● background: TRectangle ● ○ × TRectangle ColorAnimation × TColorAnimation × TColorAnimation × TColorAnimation × TRectangle ● ○ × text: TText TGIowEffect ×



FireMonkey スタイル - コンポーネント毎の例

フォーカス時に使われる TGlowEffect の色を変えてみる

オブジェクト インスペクタ 🛛 🛛 🛛				
<名前無し> TGlowEffe	ect 🔽			
プロパティ イベント				
BindingName				
Enabled	False			
GlowColor	#82005ACC			
Name				
Opacity	0.9			
Softness	0.2			
StyleName				
Tag	0			
» Trigger	IsFocused=true			

GlowEffect の色

GlowEffect が発動するトリガー ほかにも IsMouseOver などがある

Form1		Form1	
Button1	GlowColor を「Crimson」(\$DC143C) に変更	Button1	
	e	mbarcadero Developer Camp	34

FireMonkey スタイル - StyleLookup

- コンポーネントの StyleLookup プロパティを設定すると デフォルトのスタイルを変更できます。
 - StyleLookup プロパティは、このコントロールが使うスタイ ルの名前です。
 - このプロパティが設定されると FindStyleResource メソッドによって適切なスタイルが設定されます。



FireMonkey スタイル - StyleLookup

• TPanel の StyleLookup を変更して見た目をボタンのようにしてみます。

+	ゴミークト インフベ	200	🛃 Unit1.pas		• •• ••		ſ	
			Unit1 EMX.Types	オ	ブジェクト インスペ	(クタ) 🛛		g Unit1.pas
Pa	IPanel			Pa	nel1 TPanel		L	😼 Unit1 ڮ FMX.Types
	プロパティ イベント		Form1					
»	Alian	alNone	0 0 0		プロパティ イベント			Form1
	BindingName			»	Align	alNone		• • • • • • • • • • • • • • • • • • •
	CanClip	True			BindingName			
	ClipChildren	E False	•		CanClip	True		
	ClipParent	False			ClipChildren	Faise		button
	Cursor	crDefault			ClipParent	False		
	DesignVisible	True	• • •		Cursor	crDefault		
	DragMode	dmManual			DesignVisible	✓ True		ه و
	Enabled	True			DragMode	dmManual		
	EnableDragHighlight	✓ True			Enabled	✓ True		
	Height	100			EnableDragHighlight	✓ True		
	HelpContext	0			Height	100		
	HelpKeyword				HelpContext	0		
	HelpType	htContext			HelpKeyword			
	HitTest	✓ True			HelpType	htContext		
	IsDragOver	False			HitTest	✓ True		
	IsFocused	False			IsDragOver	False		
	IsMouseOver	False			IsFocused	False		
	IsVisible	✓ True			IsMouseOver	False		
÷	LiveBindings	LiveBindings			IsVisible	True		
	Locked	False		±	LiveBindings	LiveBindings		
±	Margins	(TBounds)			Locked	False		
	Name	Panel1		±	Margins	(TBounds)		
	Opacity	1			Name	Panel1		
÷	Padding	(TBounds)			Opacity			
	PopupMenu		C	±	Padding	(TBounds)		
÷	Position	(TPosition)			PopupMenu			
	RotationAngle	I O		±	Position	(TPosition)		
÷	RotationCenter	(TPosition)			RotationAngle			
Ð	Scale	(TPosition)		±	RotationCenter	(TPosition)		
	StyleLookup				Scale	(TPosition)		
	StyleName			7	StyleLookup	Buttonstyle		
	TabOrder	15			StyleName			
	Tag	0				15		
	Visible	True True			lag			
	Width	120			Visible	M Irue		
					Width	120		

Embarcadero Developer Camp

FireMonkey スタイル デモ

Embarcadero Developer Camp

FireMonkey スタイル そのほか

FireMonkey スタイル

- スタイルの個々の要素をプログラムから変更することができます(本当は)
- 現状では小細工しないと変更できません
 - Delphi XE2 Update4 で確認(QC100275 既に Closed)
 - これについては、EDN TeamJapan の高橋さんのブログ記事が詳しいです
 - http://blogs.embarcadero.com/teamj/2011/11/11/2589/

上記記事より引用

FireMonkeyのコンポーネントのFindStyleResourceメソッドを呼び出して、スタイルにア クセスしようとしますが、なぜかnil(見つからなかったということ)が返ってきてしまいま す。

この問題(現象)はQualityCentralにも報告されたのですが、トップレベルのスタイルの 場合、親クラスである「TStyleControlクラスのprotectedなメンバであるFResourceLink」 にアクセスする必要がある(As Designed, 仕様)という結果になりました。



Combarcadero Developer Camp

付録:UI 設計の指針

UI 設計の指針

- VCL / FireMonkey スタイルを利用することで OS の見た目から解放されます。
- しかし、OS の UI から解放されるわけでありません。そのため OS で提供されている一般的な操作が犠牲になる場合があります。
 - 例として
 - Flash (Air)
 - Win / Mac / Linux で動作するが、一般化のためにコンテキストメニューが廃止されている
 - Unitiy
 - Android / iOS などで動作するが、一般化のために「戻るボタン」による
 Activity 遷移がサポートされていない
- FireMonkey の場合は特に OS の提供する UX の犠牲を最小限にとど める必要があります。
- VCLはWindowsアプリケーションとなりますが、見た目が自由になった からといって、独自すぎるUIを提供すべきではありません。



UI 設計の指針 - 一般的論としての UI

• 視線は左から右、上から下に流れる



上記原則に反するが、Windows は MacOS との類似を避けるためにOK ボタンを左にした FireMonkey の場合は、MacOS に合わせるか、Windows に合わせるかはメインターゲットによる。 もしくは、プログラムで動的に変更する

42

Developer Camp

参考として Android は 2.x までは左側、3.x からは右側に変更されている

UI 設計の指針 - 一般的論としての UI

- 視覚効果は感覚に合わせる
 - グレーアウトされていれば押せない
 - MouseOver で凹むボタンなど言語道断
 感覚としてボタンは押されると凹むもの

- 曖昧なアイコンは使わない

グレーアウトされているボタンは押せない	Form1 Edit1 简正 追加 OK Cancel	
マウスオーバーするとグレーが解除されたり、 ポップアップしたりして押せる事をアピールする 変な動作がウェブにあふれていますが、間違い。 このソーユンをソリソソーのとんきとのソソソコンは ・音がOFFになる ・音がONになる どっち?	グレーアウトされているボタンは押せない。 マウスオーバーするとグレーが解除されたり、 ポップアップしたりして押せる事をアピールする 変な動作がウェブにあふれていますが、間違い。	このアイコンをクリックすると起きるアクションは? ・音がOFFになる ・音がONになる どっち?



UI 設計の指針 - 一般的論としての UI

- その UI で正しいかどうか考慮する
 - TComboBox & ListBox
 - TLabel & TImage
 - TMainMenu と TPopupMenu
 - etc.

都道府県を選択してください OK Cancel 都道府県を選択してください 千葉県 OK Cancel 都道府県を選択してください 東京都 神奈川県 丁葉県 埼玉県 栃木県、 グンマー 〇K Cancel	🥝 正解はどれ? 📃 💷	×
都道府県を県まで入れて入力してください OK Cancel 都道府県を選択してください 千葉県 OK Cancel 都道府県を選択してください 東京都 神奈川県 千葉県 「黄豆県 埼玉県 低赤木県、 グンマー OK Cancel	都道府県を選択してください	
OK Cancel 都道府県を選択してください 都道府県 OK Cancel 都道府県を選択してください 都道府県 都道府県を選択してください 東京都 神奈川県 「葉県県 埼玉県 栃木県 ウンマー 山梨 OK Cancel	都道府県を県まで入れて入力してください	
都道府県を選択してください 「葉県 OK Cancel 都道府県を選択してください 東京都 神奈川県 埼玉県 栃木県 ヴンマー 山梨 OK Cancel	OK Cancel	
千葉県 都道府県 OK Cancel 都道府県を選択してください どの UI カ 都道府県を選択してください 「葉県 埼玉県 「 ガンマー 」 山梨 OK	都道府県を選択してください	
 都道府県を選択してください 東京都 神奈川県 千葉県 埼玉県 栃木県 グンマー 山梨 	千葉県 OK Cancel	■ 都道府県の どの UI がご
東京都 神奈川県 千葉県 埼玉県 栃木県 グンマー 山梨	都道府県を選択してください	
千葉県 埼玉県 栃木県 グンマー 山梨	東京都神奈川県	<u> </u>
「 「 「 「 「 「 「 「 「 」 「 」 「 」 「 」 」 「 」 」 」 」 」 」 」 」 」 」 」 」 」	· 千葉県 塔王県	=
グンマー 山梨 OK Cancel	栃木県	
	リンマー	-
	OK Cancel	

都道府県の選択の場合、 どの UI が正解だと思いますか?



UI 設計の指針 - その他の事

ターゲット UI はターゲット層で決まる



色覚異常への対応



それぞれが判別可能か?

ユーザビリティテスト

独りよがりな UI になっていないか?









