

C++Builder Reviewers Guide

C++Builder で始める ビジュアル プログラミング

2020 <mark>年 11 月</mark> (バージョン 10.4 対応)

エンバカデロ・テクノロジーズ

バージョン 10.4 Sydney の概要や ビジュアルプログラミングの手法を解説した評価ガイド

目次

C++Builder の世界へようこそ!	1
お問い合わせ先	1
C++Builder 10.4 Sydney の新機能	
継続的なアップデート	2
メモリ管理の統合	2
IDE の機能強化	3
VCL の機能強化	3
FireMonkey の機能強化	4
C++言語の機能強化	5
RAD Server によるサービス指向アプリケーションの構築	5
無料からはじめよう!	7
個人またはスタートアップ企業の方は	7
企業ユーザーの方は	7
C++Builder のビジュアル開発を体験しよう	8
C++Builder を起動する	8
コンポーネント・UI パーツの配置	
レイアウトを調整する	11
ボタンのキャプションを設定する	12
イベントを使う	13
[戻る]ボタンに機能を実装する	15
URL を入力してページを表示する	17
表示されたページの URL とタイトルを表示する	19
ブラウズ履歴を表示する	20
新しいウィンドウでページを表示する	26
パーソナル Web ブラウザの完成	
デバッガを使ってみよう	
データベースを利用してみよう	
データベース接続を定義する TFDConnection	
TFDTable を使ってテーブルのデータセットを呼び出す	
ユーザーインターフェイスを設計する	
データフィールド設定	
DataSource とは	

レコードの移動と操作	40
データにコードからアクセスするには	41
ファイル保存用のダイアログを用意する	42
画像を保存するコードの実装	42
Windows 10 スタイルの画面にしてみよう	43

C++Builderの世界へようこそ!

このたびは、C++Builder 10.4 Sydney を評価いただき、誠にありがとうございます。C++Builder は、強 力なC++言語によるプログラミングをサポートしながら、Delphi や Visual Basic などでなじみのあるコン ポーネントによる開発を実現しており、効率的に UI を設計して C++プログラミングを進めることができ ます。C++Builder は、Windows 10、iOS 向けの真のネイティブアプリケーションの構築に加え、クラウ ドや IoT とつながるアプリケーションの構築をサポートしています。

最新の C++Builder を用いれば、新しい C++言語の強力な機能を用いながら、Windows 10 VCL コントロ ール、プラットフォームスタイル、ユニバーサル Windows プラットフォームサービスコンポーネントな どを用いて、すばやく Windows 10 対応のネイティブアプリケーションを作成できます。さらに、 FireMonkey フレームワークを用いることで、iOS モバイルプラットフォーム向けにもアプリケーション を展開することができます。

C++Builder は、接続性という点でも優れています。多様な RDBMS へのアクセスに加え、ビッグデータ、 NoSQL、クラウドサービス、エンタープライズサービスへの接続もサポートしており、サービス指向の アプリケーションを構築するために役立てることもできます。

このガイドはC++Builderの機能評価の出発点であることにご留意ください。C++Builderには、このガイドで紹介しきれないほどの数々の機能があります。エンバカデロでは、これらの機能を最大限に活用するのに役立つ補足的な情報、ビデオ、ウォークスルー、ガイドなどを用意しています。本製品の最新情報については、機能一覧、製品情報ページなどを参照してください。

お問い合わせ先

C++Builder の評価に関してご不明な点、ご質問などがございましたら、下記までお問い合わせください。

エンバカデロ・テクノロジーズ インフォメーションサービスセンター TEL:03-4540-4148 Email: japan.info@embarcadero.com

C++Builder 10.4 Sydney の新機能

C++Builder 10.4 Sydney では、最新の OS プラットフォームのサポートに加え、C++言語機能の強化、開 発生産性、効率性を強化などが加えられています。

継続的なアップデート

近年の C++Builder は、およそ年 1 回のメジャーリリースの他に、継続的に年数回のアップデートリリー スを実施しています。これは、日々変化する OS 環境への迅速な対応と、製品機能を常にブラッシュアッ プしていくという方針に基づくものです。

モバイル向けの OS は、頻繁なアップデートを行っており、ユーザーもこれらを積極的にインストールす る傾向にあるため、モバイル向けアプリを開発する場合には、最新 OS に追従するための開発環境のアッ プデートは欠かせません。一方、Windows OS は、安定的に同じバージョンを利用する傾向にあったため、 開発環境を継続的にアップデートするという考えは一般的ではありませんでした。しかし、Windows 10 は、年2回のアップデートがスケジュールされており、これらのアップデートでは大幅な機能拡張や仕様 変更が加えられています。つまり、Windows 10 向けのアプリケーション開発では、モバイル向けと同様 に、最新 OS 環境に追従する継続的な開発が必要となるのです。

エンバカデロでは、このような OS 環境の変化と開発需要に対応するため、上記のような継続的なアップ デートを実施しています。そして、開発者の負担を最小化するため、製品にアップデートサブスクリプ ション(年間保守)を標準添付しています。

開発者は、アップデートサブスクリプションサービスを継続することで、常に最新環境を利用すること ができます(同時に多数の旧バージョンへのアクセスも可能です)。

メモリ管理の統合

Delphi / C++Builder のメモリ管理機構は、すべてのサポートプラットフォーム(モバイル、デスクトッ プ、サーバー)で、従来型のオブジェクトメモリ管理の方式に統一されました。ARC(Automatic Reference Counting:自動参照カウント)と比較して、既存コードとの互換性が高く、コンポーネント、 ライブラリ、エンドユーザーアプリケーションで、よりシンプルなコーディングが可能になります。 ARC モデルは、文字列管理やインターフェイス型の参照向けに、すべてのプラットフォームで残されま す。この変更は C++コード内での Delphi スタイルのクラスの生成や定義が、ヒープに割り当てられる C++クラスと同様の通常のメモリ管理に従うこととなり、複雑性が大幅に低減することを意味します。

IDE の機能強化

10.4 では、IDE の Getlt パッケージマネージャを大幅に強化しました。パッケージをリリース日で並び変 えたり、インストールパッケージ、アップデートサブスクリプションユーザー向けのコンテンツ、アッ プデートが提供されているパッケージなどの基準でフィルタリングできるようになりました。

また、Getlt インストーラテクノロジーを使用した統合インストーラを導入しました。これにより、(イ ンターネット接続環境での)オンラインインストールと、(ISO イメージを用いた)オフラインインスト ールの双方で、単一のインストーラを使用するようになりました。オンラインとオフラインインストー ルのいずれでも、プログラミング言語とターゲットプラットフォームの選択を任意に選択可能。言語サ ポート、ヘルプリソースなど、インストールセットはいつでも追加/削除可能です。

VCLの機能強化

VCL(Visual Component Library)は、Delphi / C++Builder のファーストバージョンから利用されている Windows 向けのビジュアルコンポーネントです。VCL は、近年の Windows 環境の変化に対応し、数多く の機能強化が施されており、現在も進化を続けています。

ここ数年の大きな機能強化は、Windows 10 への対応と High DPI サポートの強化です。Windows 10 向けの新しい UI コントロールが追加されたほか、高解像度モニターでの表示などを改善しています。

以下は、10.4 で新たに追加された VCL の強化ポイントの一例です。

- High DPI における VCL スタイルの変更: High DPI および 4K モニターサポート向けに、VCL スタイ ルアーキテクチャが大幅に拡張され、VCL フォーム上のすべての UI コントロールは、フォームが表 示されるモニターの適切な解像度に合わせて自動的にサイズ変更されます。スタイル API もアップデ ートされ、High DPI スタイルをサポートしました。
- 新しい High DPI スタイル:多数のビルトインおよびプレミアム VCL スタイルを更新し、新しい High DPI スタイル モードをサポートすることで、あらゆるモニター向けの完成度の高いアプリケーションの設計が可能になりました。
- コントロールスタイル単位での VCL: 複数の VCL スタイルを、単一のアプリケーション内の異なる フォーム、あるいは同じフォーム上の異なるビジュアルコントロールに適用できるようになりまし た。これには、プラットフォームデフォルトテーマとの両立も含みます。その結果、スタイル設定 にさらなる柔軟性が加わるだけでなく、スタイルをサポートしていないサードパーティコントロー ルを、スタイル付き VCL アプリケーションで使用可能にする効果も得られます。

- 新しい VCL コンポーネント: Edge Browser コントロール: TEdgeBrowser を用いれば、VCL アプリ ケーションで、Chromium ベースの新しい Microsoft Edge WebView2 を使用できます。これにより、 よりモダンでセキュアな HTML エンジンを利用できるようになります。従来の TWebBrowser コンポ ーネントは、既存の Internet Explorer ベースのエンジンと Edge ブラウザを、必要に応じて動的に切 り替えできるようになります。
- 新しい VCL コンポーネント: TTitleBarPanel および CustomTitleBar プロパティ:新しい TTitleBarPanel コントロールと TForm.CustomTitleBar プロパティにより、VCL フォームのネイティ ブ Windows タイトルバーのカスタマイズが可能になりました。これにより、Office、Explorer、 Google Chrome などのアプリケーションと同様の、最新の拡張タイトルバーを構築できます。
- 新しい VCL コンポーネント:マルチ解像度をサポートするイメージコンポーネント:新しい TVirtualImageがマルチ解像度/DPIスケーリングをサポート。従来のTImageコンポーネントと置き 換えることで、高品質なイメージスケーリングと表示が可能になります。
- Windows API のアプデート: RAD Studio が提供する優れたプラットフォーム統合をさらに向上させるため、多くの API 宣言を拡張し、さらに追加も行いました。

FireMonkey の機能強化

FireMonkey は、Delphi / C++Builder でマルチデバイスアプリケーションを開発する際に利用するクロス プラットフォームコンポーネントフレームワークです。

10.4 では、Professional、Enterprise、Architect のいずれのエディションでも、FireMonkey によるモバイ ル開発をサポートしています。単一コードにより、複数プラットフォーム向けのネイティブ開発をサポ ートしているので、プラットフォームごとに複数の言語やツールを習得する必要がなく、劇的な開発効 率化が可能になります。

10.4 では、FireMonkey に関連して以下のような機能強化が施されています。

- Metal API のサポート: macOS プラットフォーム(Delphi) で、古い Quartz API や OpenGL (Apple で廃止予定) ではなく、Metal API に対してアプリケーションを構築できるようになりました。これ により、今後の要件へのスムーズな移行と、画面描画のより良いパフォーマンスが提供されます。
- 最新の iOS SDK のサポートに加え、組み込みの IDE サポートを通じ、Apple の起動画面のストーリ ーボード要件にも対応できます。

- 10.4 では、Windows プラットフォームでのスタイル付き TMemo コンポーネント用の新しい FMX 実 装が含まれており、IME のサポートが改善され、追加の機能強化が行われています(macOS プラッ トフォームは 10.4.1 からサポート)。
- iOS 用 TWebBrowser コントロールは、WKWebView API を使用して実装されました。
- Media Player コントロール の macOS 実装は、AVFoundation の使用により可能となりました。
- FireMonkey フレームワークを使用する開発者は、サポート済みオペレーティング システムの最新バ ージョンをターゲットとすることができます。

C++言語の機能強化

C++Builder には、従来からの Win32 向けクラシックコンパイラ(bcc32)と、Clang ベースの新しいコ ンパイラの双方が搭載されています。10.4 Sydney は、C++17 をサポートしており、マルチプラットフォ ームに対応するべく、各プラットフォーム向けにコンパイラが用意されています。

- C++ライブラリサポートの拡張: 10.4 では、さまざまな人気の C++ライブラリを C++Builder 向けに 移植し、C++Builder 内で利用できるように最適化しました。これには、すでにサポートされている Eigen などに加え、ZeroMQ、SDL2、SOCI、libSIMDpp、Nematode などのライブラリが含まれます (Getlt パッケージマネージャからダウンロード可能)。
- C++向け Win64 デバッガとリンク: 10.4 では、新しい Windows 64-bit C++向けデバッガが導入され ました。LLDB をベースとした新しいデバッガは、64-bit アプリケーションのデバッグ時の安定性を 大幅に改善し、C++や Delphiの文字列型のほか、STL コレクション(std::vector や std::map など) など、複雑なデータ型の評価やインスペクトをサポートしています。新しいデバッガでは、従来と は異なるデバッグ情報のフォーマットを使用しています。これにより、より安定したデバッグが可 能となり、従来よりも強力なデータの評価/インスペクトを実現しています。
- ツールチェーンのパフォーマンスと品質の向上: Dinkumware による多数の STL の改善といくつかの 主要な RTL メソッドとその周辺が改善されました。これは、汎用 C++ライブラリとの互換性を改善 させるために行われた作業をベースとしています。また CMake サポートにおける改善および多数の 品質および安定性における改善されています。

RAD Server によるサービス指向アプリケーションの構築

RAD Server は、REST/JSON ベースのカスタム API を実装するための中間サーバー機能を提供します。開 発者は、従来の Delphi / C++Builder でも使い慣れたデータアクセスコンポーネントやロジックコンポー ネントを用いて、容易にサーバーサイドアプリケーションを実装できます。





RAD Server を用いれば、モバイルクライアントとバックエンドシステムを結び付け、業務システムにモ バイルクライアントを加えることが可能になります。さらに、新たにエンバカデロの製品ファミリーに 加わった Web 開発ソリューション Sencha のバックエンドとしても RAD Server を利用できるため、既存 の Delphi / C++Builder アプリケーションを Web に拡張する手段としても有効となります。

無料からはじめよう!

C++Builder は、エンバカデロが提供する C++向けビジュアル開発ツールです。その歴史は古く、ボーラ ンドが提供してきた Turbo C や Borland C++のコンパイラがベースとなっています。Delphi と同じよう なビジュアル開発を C++で実現したいという要望に応えて 1997 年に最初のバージョンがリリースされて 以降、多くの C++開発者に支持されてきました。

Windows 向けの C++開発ツールとして登場した C++Builder は、その後、マルチデバイス向け開発ツー ルへと進化し、現在では、Windows、iOS 向けのネイティブアプリケーション(※)を開発できます。

C++Builder による開発を体験するには、無料から始めることができます。30 日トライアル版をダウンロードすれば、C++Builder のすべての機能を 30 日間試用することができます。また、個人またはスタートアップ企業の方であれば、Community Edition をダウンロードすることで、Professional 版相当の機能を使って開発を始められます(商用開発には制限があります)。

個人またはスタートアップ企業の方は

以下のページから C++Builder Community Edition をダウンロードしてください。

https://www.embarcadero.com/jp/products/cbuilder/starter

C++Builder Community Edition には、Windows、iOS 向けアプリケーション(※)を単一の C++コード ベースから開発できるマルチデバイス開発機能、ローカルデータベースアクセス機能などが搭載されて います。個人開発者または 5 名以下の開発者の企業で利用(企業の年間売上が 5,000 US ドル未満、また は個人開発者の場合、作成したアプリケーションの年間売上が 5,000 US ドル未満の場合に限る)するこ とができます。

※ Android および macOS については、32bit アプリケーション開発のみの限定的なサポートが提供されています。

企業ユーザーの方は

Community Edition の利用規定に該当しない企業ユーザーの方は、以下のページから RAD Studio トライアル版をダウンロードしてください。

https://www.embarcadero.com/jp/products/rad-studio

RAD Studio トライアル版は、Delphi / C++Builder のすべての機能を 30 日間試用することができます。

C++Builder のビジュアル開発を体験しよう

初めて C++Builder に触れる方は、複雑な C++による UI 設計を行うことなく、「コンポーネント」と呼 ばれるパーツをドラッグ&ドロップで配置するだけで簡単に UI を構築できることに驚くかもしれません。 C++Builder は、ビジュアル操作で簡単にアプリケーションを開発できるツールですが、C++言語の機能 を制約するものではありません。開発者は、コンポーネントによる生産性と C++言語のパワーを両立し た開発が可能なのです。

はじめに、C++Builder によるビジュアル C++プログラミングの手法を理解するために、簡単なパーソナル Web ブラウザを作ってみましょう。

C++Builder を起動する

C++Builder、あるいは RAD Studio のアイコンをダブルクリックして統合開発環境(IDE)を起動します。 C++Builder によるプログラミング作業は、この統合開発環境で行います。ユーザーインターフェイスの 設計、コーディング、ビルド、デバッグ、実デバイスへの配置など、すべての操作をこの環境で実行で きます。

新規にアプリケーションを作成するには、 [ファイル(F) | 新規作成(N) | Windows VCL アプリケーション - C++Builder]を順に選択します。



コンポーネント・UI パーツの配置

右下の「ツールパレット」から TPanel をドラッグし、画面中央のフォーム上にドロップします。ツール パレットの検索バーに「Panel」と入力すると、探しやすくなります。



同様に、TWebBrowser をドラッグして、フォーム上の別の位置にドロップします。フォームは、以下の ようになります。



頻繁に使うコンポーネントは、簡単な方法で追加することもできます。フォーム上に配置した、TPanel (自動的に「Panel1」という名前が振られています)を右クリックすると、クイックメニューが表示さ れます。 embarcadero[®]

ここから、[コントロールの追加(S)|ボタン]を選択すると、Panel1 の上にボタンを追加することがで きます。



ボタン(Button1)を配置したら、再び Panel1 を右クリックして、 [コントロールの追加(S) | 編集] を 選択して、Panel1 の上に Edit1 を追加します。

ここで左上の「構造」ペインを確認してみましょう。Button1 と Edit1 は、Panel1 の下の階層(子項 目)に位置していることが分かります。



Panel のように、別のコンポーネントをその上に置くことのできるコンポーネントを「コンテナ」といい ます。コンテナの上に置かれたコンポーネントは、「構造」ペインでは、コンテナの子項目として表示 されます。Panel を移動させると、その上に置かれた、Button1、Edit1 は一緒に移動します。コンテナ を用いることで、複数のコンポーネントをグループ化してレイアウトすることができます。 もし、「構造」ペインで、**Button1** や **Edit1** が Panel の下に位置していない場合には、Panel の子項目に はなっていないということです。これでは、レイアウトをする上で困ったことになるので、修正してお く必要があります。次のように「構造」ペインで操作をしてください。

- Edit1 を Panel1 の上にドラッグアンドドロップします
- Button1 を Panel1 の上にドラッグアンドドロップします

この操作は、必ず「構造」ペインで行います。フォーム上でコンポーネントを移動させても、親子関係 は変わらないことに注意してください。

レイアウトを調整する

コンポーネントは、XY 座標で配置されています。しかし、実際のアプリケーションでは、ウィンドウの 大きさを変えても、UI パーツは、適切にレイアウトされますよね。このようなレイアウトの調整を行う には、レイアウトを調整するためのプロパティを設定します。

「プロパティ」とは、コンポーネントの動作や外観などをカスタマイズできるさまざまなパラメータで す。画面左下の「オブジェクト インスペクタ」には、コンポーネントに用意されたさまざまなプロパテ ィの一覧が表示されています。C++Builder では、コードを記述しなくても、プロパティを設定するだけ で簡単にコンポーネントの外観や振る舞いをカスタマイズできるのです。

では、最初に Panel1 のレイアウトを調整してみましょう。「構造」ペインで、Panel1 をクリックしま す。すると、「オブジェクト インスペクタ」の「プロパティタブ」内に、Panel1 のプロパティが表示さ れます。ここから、「Align」を見つけ、右側の値列のドロップダウンリストをクリックし、画面上部 (alTop)を選びます。



同様の操作で、Button1、Edit1、WebBrowser1の「Align」プロパティも設定します。

Button1	
項目	値
Align	左側(alLeft)
Edit1	
項目	值
Align	画面中央(alClient)
WebBrowser1	
項目	值
Align	画面中央(alClient)

以上で、フォームは次のような外観になります。

🐯 Form1		
Button 1	Edit1	

フォームの大きさを変更しても、Button1、Edit1 は適切な場所に配置されていますね。

ボタンのキャプションを設定する

プロパティは、ボタンのキャプションを設定するためにも使えます。Button1 をクリックして選択状態 にしたら、次のように「Caption」プロパティを設定します。

Button1	
項目	值
Caption	戻る

この操作で、フォームのボタンのキャプションは、「戻る」に変わります。C++Builder では、プロパティの変更を設計フォームで直ちに確認できるため、直感的な開発が可能なのです。

Som Torm Torm Torm Torm Torm Torm Torm To		
	オブジェクト インスペクタ	₽ ×
	Button1 TButton	\sim
	プロパティ イベント	Q
	Action Align I alLeft AlignWithMargin: False Anchors [akLeft,akTop,akBottom] BiDiMode bdLeftToRight Cancel False Caption 戻る CommandLinkHi	
	> Constraints (TSizeConstraints) Cursor crDefault	\sim
	クイック編集… ビジュアルにバインド… すべての項目が表示されています	

さて、この[戻る] ボタンの機能はどのように実装するのでしょうか?ここで登場するのが「イベント」 です。でも急がずに、簡単なところから始めてみましょう。

イベントを使う

embarcadero[®]

通常、Web ブラウザは、起動するとホームページに設定されたページが表示されます。フォーム上に配置した TWebBrowser には、設定されたホームページを表示する機能が用意されています。ホームページは、Windows のインターネットオプションで設定されている URL です。

では、このホームページの表示は、いつ、どのように呼び出すのでしょうか?

「プロパティ」は、コンポーネントの状態を設定したり、参照することができます。例えば、Width プロ パティは、そのコンポーネントの幅を表します。この値を参照すれば現在の幅が分かりますし、値を変 更すれば実際に幅が変わります。

一方、コンポーネントの機能を呼び出すためには、「メソッド」が用意されています。TWebBrowser の GoHome メソッドは、「ホームページを表示しなさい」という指令をコンポーネントに送ります。GoHome メソッドを呼び出すコードを記述することで、ホームページが表示されるというわけです。 では、いつこれを呼び出せばいいのでしょうか?

そのために用意されたものが「イベント」です。コンポーネントには、複数のイベントが用意されてい ます。例えば、ボタンをクリックすると「OnClick」イベントが発生します。マウスが上を通過すると 「OnMouseOver」イベントが発生します。このように、コンポーネントに対してなされた「何か」によ って、そのイベントが発生するのです。

そして、このイベントに対して「イベントハンドラ」と呼ばれるプログラムコードを記述することができます。これにより、さまざまなカスタムコードを定義できるのです。

今回は、フォームが作られたときに、ホームページを表示させます。つまり、Form の「**OnCreate**」イベ ントです。

これを行うには、Form1 をクリックして「オブジェクト インスペクタ」の「イベント」タブ内の 「OnCreate」の右横の空白部分(値列)をダブルクリックします。

💀 Project1 - RAD Studio 10.4	4 - Unit1.cpp		デフォルトレイアウト 🗸	
ファイル 編集 検索	表示 リファ	ウタリング プロジェクト 実行 コンス	ポーネント ツール タブ ヘルプ	
) 🗳 ~ 🔒 📮	🗟 📷 📄 🕨 × 🖬 🖬	▮⊊⊑⊑	
構造	4 ×	ウェルカム ページ Unit1.cpp ・		
File & Compared to the second seco		Term1 展る		
オブジェクト インスペクタ Form1 TForm1 プロパティ イベント OnCloseQuery OnConstrainedR OnContextPopu OnCreate OnDblClick OnDeactivate OnDestroy	v ∓ × • • ^			-

値列に FormCreate と入力され、コードエディタが表示されます。これがイベントハンドラです。イベントハンドラの必要なコードは C++Builder が自動的に生成してくれるので、{と}で囲まれた中身だけを記述すればいいのです。ここでは、次のように WebBrowser1->GoHome();と、一行記述します。

void _ {	fastcall TForm1	.::F	ormCrea	te(TObject *Sender)		
Web	oBrowser1->GoHome	e();				
}						
	構造	۲× ۱	ウェルカム ページ	Unit1.cpp $ imes$		\sim
	11 🗄 😰 🦑 щ	á	💐 ~ 🖶 ~	V TForm1::FormCreate V	Q	
		ner) :r)	; evoid 19 20 	fastcall TForm1::FormCreate(TObject *Sender) Browser1->GoHome();		^

間違ったイベントをダブルクリックして、必要のないイベントハンドラができてしまっても気にしない でください。C++Builder は、空白のイベントハンドラを自動的に削除してくれます。手作業で削除する と、自動生成したコードと一致しなくなる恐れがあるので、やめたほうがいいでしょう。

[戻る] ボタンに機能を実装する

では、この要領で〔戻る〕ボタンにも機能を実装してみましょう。

「戻る」という機能は、TWebBrowser の GoBack メソッドを使うだけです。このように、コンポーネントには、あらかじめいろいろな機能が用意されているので、コード量は各段に少なくなりますよね。これも C++Builder の利点のひとつなのです。

先ほど、コードエディタを使ったので、設計フォームは隠れてしまいました。再び設計フォームを表示 するには、画面右下の「デザイン」タブをクリックします。 embarcadero

Button1([戻る]ボタン)を選択したら、オブジェクトインスペクタの「イベント」タブ内にある 「OnClick」の値列をダブルクリックします(もう、ボタンをクリックしたときの処理を実装するために、 OnClick イベントを選択したことはお分かりですね)。

アイル 編集 検索 表示 リア799リング プロジェクト 裏行 コンポーネント ツール ダブ ヘルブ !! こ こ こ こ こ こ こ こ こ こ こ こ こ こ こ こ こ こ こ	noise the studio 10.4 - Unit1.cpp		デフォルトレイアウト 🗸	오 哏~ ? - ㅁ ×
Image: Stand Stan	ファイル 編集 検索 表示 リフ	ァクタリング プロジェクト 実行 コンボーネント ツール タブ ヘ		
構造 9 × クエルカムページ UnitLopp Project1.cbproj - プロジェクト + × プレ 日 ふ や ●		120 115 110 ► × 12 × 11 ■ 12 12 12 12 14 1	✓ ● ✓ Windows 32 ビット ✓	
$ \begin{array}{c} \hline D & \Box & \Diamond & \bigcirc & & & & & & & & & & & & & & & &$	構造 ¥ ×	ウェルカムページ Unit1.cpp ●	×	Project1.cbproj - プロジェクト # ×
マジョンパトインパング キ × Button 1 TButton プロ/b イントト の Action DropDownMenu Images LiveBindras OnContestPopup OnDragOver OnDragOver	Image: Second secon	Coming Contraction		 ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○
JULY IV IV Action CustomHint DropDownMenu Images Standard V. LiveBindings LiveBindings OnClick OnClick OnClick System OnConstRPopup OnDrogOver OnDrogOver Dialogs	オブジェクトインスペクタ サ× Button1 TButton ~			Project1 モデルビ データェ マルチデ
OnDropDownClick OnEndDock OnEndDock OnEndDrag クイング資産」、ジュアルにプインド シャロー	Action CustomHint DropDownMenu Images > LiveBindings OnCitick OnContetPopup OnDragDrop OnDragDrop OnDragDworClick OnEndDrag クイック対象重、ビジュアルにパインド、			/fLvyh # × Standard A Standard A Additional A Win32 System Win31 Dialogs Data Access Data Access > Data Access Data Access > Datasnap Server Xmin
オペイの酒目が表示されています UnitLopp Tunitin デザイン 層歴 シビッを時間の	すべての項目が表示されています		Unit1.cpp Unit1.in デザイン 履歴	> LiveBindings

Button1Clickのイベントハンドラには、以下のようにコードを記述します。



ところで、 [戻る] ボタンを押したときに、戻るページがない(履歴がない)場合にはどうなるでしょうか?この場合はエラーが発生してしまいますが、一般的には、そのようなことのないように履歴がないときには、 [戻る] ボタンが押せないように無効になっていますよね。

もちろん C++Builder でもそのように設定することができますが、今回は割愛してアプリケーションの完成を急ぎましょう。興味のある方は、あとで実装してみてください。

URL を入力してページを表示する

多くのブラウザがそうであるように、フォーム上部に配置した Edit1 に URL を入力して [Enter] キーを 押すと、そのページが表示されるようにしようと思います。

Edit1 でキーが押されたときに発生するイベントは、OnKeyDown です。このイベントハンドラには、押 されたキーの情報が渡されます。Enter キーかどうかは、あらかじめ定義された値「vkReturn」を使って 調べます。では、この値はどこで定義されているのでしょうか?

ちょっと視野を広げて、C++Builder のソースコード全体を眺めてみましょう。今回作成したアプリケー ションのユニットは Unit1.cpp と言うファイル名ですが、cpp ファイルとは別に Unit1.h ファイルとい う名前のファイルもあり 2 つのファイルで構成されています。

#ifndef Unit1H #define Unit1H //-----#include <System.Classes.hpp> #include <Vcl.Controls.hpp> #include <Vcl.StdCtrls.hpp> #include <Vcl.Forms.hpp> #include <SHDocVw.hpp> #include <Vcl.ExtCtrls.hpp> #include <Vcl.0leCtrls.hpp> //----class TForm1 : public TForm { __published: // IDE で管理されるコンポーネント private: // ユーザー宣言 // ユーザー宣言 public: ___fastcall TForm1(TComponent* Owner); }; //---------extern PACKAGE TForm1 *Form1; //-----#endif

Unit1.h ファイル中には、インクルード指定とクラスの定義が記述されています。

embarcadero[®]

#includeは、他のユニットや C++ソース内容を取り込む場合に利用します。その後に書かれている class TForm1はフォームの宣言部です。つまり、先ほどビジュアル操作で設計していたフォームの定義 です。Unit1.cpp では、Unit1.h に記述された class 宣言をもとに、他のユニットからアクセスできな い実装部を記述しています。

インクルード指定に戻りましょう。vkReturn が定義されているのは、System.UITypes.hpp です。この ファイルはインクルード指定に含まれていませんので、この値を使えるようにするために、#include <System.UITypes.hpp> を1行追加します。



設計フォームを表示して Edit1 を選択し、「OnKeyDown」イベントを設定します。Edit1KeyDown のイベ ントハンドラを次のように記述します。

```
void __fastcall TForm1::Edit1KeyDown(TObject *Sender, WORD &Key, TShiftState Shift)
{
    if (Key == vkReturn) {
        WebBrowser1->Navigate(Trim(Edit1->Text));
    }
}
```

ここまでで Web ブラウザの基本的な動作を実装することができました。簡単でしたね。

ではもう少しだけ機能を追加して、より本物に近づけてみましょう。

表示されたページの URL とタイトルを表示する

URL を入力する Edit1 は、現在の URL を表示するボックスとしても機能しなければなりません。そのため、ページが表示されたら、その URL を Edit1 に設定し直すという動作が必要です。これを実装するのにちょうどよいイベントが、TWebBrowser に用意されています。OnDocumentComplete です。

WebBrowser1 を選択して、「OnDocumentComplete」イベントのイベントハンドラを作成します。



作成されたイベントハンドラ WebBrowser1DocumentComplete に、以下のコードを記述します。

```
void __fastcall TForm1::WebBrowser1DocumentComplete(TObject *ASender,
        IDispatch * const pDisp, const Olevariant &URL)
{
        Edit1->Text = WebBrowser1->LocationURL;
}
```

もうひとつ実装すべきなのは、タイトルの表示です。Web ページにはタイトルが定義されていますが、 多くの Web ブラウザでは、ウィンドウのタイトルに、ページのタイトルを表示するようにしています。

embarcadero[®]

これを行うには、TWebBrowser の OnTitleChange イベントを使います。WebBrowser1 を選択して 「OnTitleChange」イベントのイベントハンドラを作成、次のようにコードを記述します。

さて、ここで新しい要素 this が出てきました。this は自分自身を表す変数です。ここでいう自分自身 とは、TForm1 です。

今編集している Unit1.cpp の宣言部である Unit1.h には、TForm1 というクラスが宣言されています。 TForm1 は、現在作成しているフォームです。C++Builder では、コンポーネントフレームワークを使うこ とで、他の C++開発ツールとは異なり、複雑なコードを組むことなく、ユーザーインターフェイスを構 築できます。

TForm1 は、TForm というからっぽのフォームを継承しています。Web ブラウザの機能を実装するために、 ウィンドウの基本的な機能を作らなくてよかったのは、TForm の機能を利用してきたからです。ここで詳 細に説明することはしませんが、C++Builder のコンポーネントフレームワークの構成を深く知るには役 立つ知識です。

コードの説明に戻りましょう。this->Caption という記述は、自分自身の Caption プロパティという意 味になります。TForm の Caption プロパティは、ウィンドウのタイトルを表しますので、ここに渡され た Text(新しいタイトル)を設定すれば、ページのタイトルを表示することになるのです。

ブラウズ履歴を表示する

簡単な1行コードを記述し、フォーム画面タイトル表示することができました。さらに本格的ブラウザ のようなアプリに仕上げていきしょう。次はブラウザにアクセスした URL の履歴をメモリに保存し、履 歴ボタンをクリックすることで過去アクセスしたリストを呼び出します。C++の標準ライブラリを使っ て機能を実装していきますが、前章のようなシンプルな1行加えたようなコードでは実現できません。 少し複雑になりますが履歴機能を作っていきましょう。



まず TButton を Panel1 上に配置し、履歴ボタンとします。



履歴ボタンのプロパティを設定します。

Button2

項目	值
Align	alRight
Caption	履歴

履歴リストを表示するために新しいフォームを作成します。メニューから[ファイル(F)|新規作成(N)| VCLフォーム]を選択すると、中央のフォームデザイナー画面に新たにフォーム(**Unit2.cpp**)が作られ ます。

まず Unit2.cpp フォームのプロパティを設定しましょう。

Form2	
項目	值
BorderStyle	bsDialog
Caption	履歴
Width	265
Height	274





Unit2.cpp(Form2)にURL 履歴を表示するため TListBox コンポーネントを配置します。

配置された TListBox のプロパティ設定を行います。Form2 画面全体に ListBox 表示したいので、Align を 変更しましょう。

ListBox1

項目	值
Align	alClient

フォームデザイナー画面の上部タグをいったん Unit1.cpp に切り換え、さらにメニューから[ファイル (F) | 使用するユニット(U)](Alt+F11)を選択すると、以下のダイアログが表示されます。

🐵 使用するユニット 🛛 🔪	<
Q	
Project1PCH1.h (\\Mac\Home\Documents\Embarcadero\Studio\Projects)	
Unit2.pas (\\Mac\Home\Documents\Embarcadero\Studio\Projects)	
追加先: ・ ヘッダー ・ ・ ソース のK キャンセル	

ダイアログ内のリスト Unit2.cpp を選択し、追加先で「ヘッダー」を選択したら [OK] ボタンをクリックします。これにより、Unit1.cpp から Unit2.cpp を参照できるようになります。

続いて C++コードを記述していきます。まず、フォームデザイナー上部のタブを切り換え、Unit1.cpp を選びます。次に画面下のタブで、Unit1.h を選択します。



C++ヘッダーが記述されたエディタ画面に切り替わります。#include という記述が先頭付近にたくさん書かれています。その最後に以下の2行を追加します。



記述した vector と memory は、C++の標準ライブラリの一部の型を呼び出すための宣言です。

embarcadero

次に履歴を収納するための動的配列 vector<T>型を宣言してみましょう。エディタ画面の中央行あたり に private:と書かれた行があります。その直下に以下のコードを記述します。

```
private: // ユーザー宣言
    std::vector<UnicodeString> History;
```

以上で、C++コードを記述するための準備は完了です。

それでは Unit1.cpp イベントハンドラ内のコードを記述していきましょう。前章では TWebBrowser の OnTitleChange イベントで 「OnTitleChange」イベントのイベントハンドラを作成しフォームタイトル を表示しました。

その続きのコードを次のように記述します。

続いてブラウザ URL 履歴を呼び出せるように、履歴ボタンのイベントハンドラを追加しましょう。

ボタンイベントハンドラの作成手順は同様です。履歴ボタンを選択し、オブジェクトインスペクタから イベントタグで OnClick 項目をダブルクリックします。履歴画面を呼び出すためのコードは、次のように 記述します。

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
    std::unique_ptr<TForm2> fHistory(new TForm2(this));
    fHistory->Position = poMainFormCenter;
    fHistory->ListBox1->Clear();
```

```
std::vector<UnicodeString>::iterator iter;
for (iter = History.begin() ;iter != History.end(); iter++) {
     fHistory->ListBox1->Items->Add(*iter);
}
fHistory->ShowModal();
}
```

少し難しいコードですが、Unit2.cppのフォームデザイナーで作成した ListBox1 が配置されたフォームが表示され、リスト内に過去にアクセスした URL の履歴が入ったウィンドウを表示します。

ここでは、C++特有のコードを使っています。include <memory>で呼び出した unique_ptr<TForm2>型 はスマートポインタの一種で、必要なくなったメモリを自動破棄してくれる型です。TForm2 を新たに作 っていますが、必要がなくなった時点でメモリは解放されます。

もう1箇所は、最初にユーザー宣言した vector<UnicodeString> History に保存した内容を ListBox1 に追記していくコードです。

アプリケーションを実行し、履歴ボタンをクリックすると、図のように URL 履歴を表示するウィンドウ が現れます。このウィンドウはフォームなので、「X」ボタンで終了します。



新しいウィンドウでページを表示する

TWebBrower には、標準的な Web ブラウザ機能の多くがあらかじめ用意されているので、ブラウザ上で マウスを右クリックして [新しいウィンドウで開く] といったメニューを選択することもできます。こ の動作はまだ定義していません。このままでは、別のブラウザを表示してしまうので、新たにもうひと つフォームを表示するように記述してみましょう。

「TForm1 はクラスである」と説明しました。クラスのインスタンスが、実際に表示されているフォーム です。アプリケーションが起動すると、最初に TForm1 のインスタンスがひとつ作成され、表示されます。 この部分のコードは記述していませんが、C++Builder が自動的に作成してくれているのです。

```
#include <vcl.h>
#pragma hdrstop
#include <tchar.h>
//-----
                            _____
USEFORM("Unit1.cpp", Form1);
//------
int WINAPI _tWinMain(HINSTANCE, HINSTANCE, LPTSTR, int)
{
   try
   {
       Application->Initialize();
       Application->MainFormOnTaskBar = true;
       Application->CreateForm(__classid(TForm1), &Form1);
       Application->Run();
   }
   catch (Exception & exception)
   {
       Application->ShowException(&exception);
   }
   catch (...)
   {
       try
       {
           throw Exception("");
       }
       catch (Exception & exception)
       {
           Application->ShowException(&exception);
```



```
}
}
return 0;
}
```

このコードは通常意識する必要はありません(画面右側の「プロジェクト マネージャ」で 「Project1.exe」を選択して右クリックし [ソースの表示(V)] を選択すると表示されます)。

「新しいウィンドウを開く」という動作を実装するには、フォームをもうひとつ、つまり TForm1 をもう ひとつ作成する必要があります。これには new 演算子を呼び出します。 [新しいウィンドウで開く] メ ニューが選択されると、TWebBrowser の「OnNewWindow3」イベントが発生します。これに対応するコ ードを記述するために、WebBrowser1 の OnNewWindows3 イベントにイベントハンドラを記述します。

記述するコードは、新しいフォームを作成し、これを表示します。このためには、作成したフォームを 変数に代入する必要があります。C++では、関数内のローカル変数はスコープ内に宣言できます。ここ では、newForm というローカル変数を、イベントハンドラのスコープ内に宣言します。

イベントハンドラのコード全体は、以下のようになります。

ここまでの作業が完了したら、[ファイル | すべて保存]でユニット、プロジェクトファイルを保存し ておきましょう。これまでファイル名として Unit1、Project1 などのデフォルトの名前を使ってきました が、BrowserUnit、MyBrowser などの名称を付けておくといいでしょう。

パーソナル Web ブラウザの完成

以上で、おおよそ Web ブラウザらしい動作が実装できました。F9 キーを押すか、[実行(R)|実行(R)]メ ニューを選択して、プログラムを実行します。



しかし、まだまだ検討の余地は残っているようです。例えば、Java Script エラーが発生したときの処理な ど。また、多くの Web ブラウザでは、メインフォームという概念はなく、いくつもウィンドウを開いて、 どの順番に閉じても構わないという動作をします。しかし、今回の実装では、最初のウィンドウをメイ ンウィンドウとしており、複数のウィンドウを開いているときに、メインを閉じると、すべてのウィン ドウが閉じられ、アプリケーションが終了してしまいます。

あらゆる操作を想定してプログラムを記述することは、困難なことかもしれません。しかし、 C++Builder にはさまざまなエラーハンドリングのしくみが用意されているので、比較的容易にエラーに 対応するコードを実装することができます。

デバッガを使ってみよう

C++Builder の統合開発環境には、ソースコードレベルでアプリケーションをステップ実行したり、変数 の内容を監視できるビジュアルデバッガが搭載されています。デバッガを使えば、プログラムが意図し たとおりに実行されているか、変数に正しく値が設定されているかを確認し、容易にバグを取り除くこ とができます。

作成した Web ブラウザアプリケーションをデバッグ実行してみましょう。メインメニューで[実行(R)| 実行(R)]を選択します。

ソースコード中のイベントハンドラ TForm1::Edit1KeyDown の最初の行の左余白をクリックして、ブレ ークポイントを設定します。アプリケーションで、Edit1に何か入力すると、ブレークポイントを設定し た行で実行が停止します。



監視式に、引数として渡された Key などの変数を追加すると、現在どのような値がイベントハンドラに 渡されているかを確認することができます。

データベースを利用してみよう

C++Builder と Delphi には、データベースアクセスのための強力な機能が搭載されています。これは、 Delphi が元々Oracle データベースにアクセスするアプリケーションの開発を想定して開発されたことに も由来します。

最新バージョンでは、FireDACと呼ばれるマルチデバイス対応の共通データベースアクセスフレームワー クが用意されており、多様なデータベース(Oracle、SQL Server、IBM Db2、Sybase、InterBase、 MySQL、PostgreSQL、Access、MongoDB など)に高速かつ直接、ネイティブアクセスできます。



今回は、ご覧のような「魚図鑑」データベース(古くから Delphi や C++Builder のサンプルとしておなじ みのものです)を使って、FireDAC によるデータアクセスの方法を紹介します。他のデータベースを使う 場合も、基本的な手順は変わりません。

BDE などの古いデータベースアクセスの方法をご存じの方は、おなじみのデータベースを利用するアプ リケーションの作成を通して、最新の FireDAC でどのようにデータアクセスを行うのか、その基本的な 手順を理解できると思います。

データベース接続を定義する TFDConnection

[ファイル(F) | 新規作成(N) | Windows VCL アプリケーション - C++Builder] を選択し、新規アプリケー ションを作成します。

ツールパレットから、FireDAC カテゴリにある、TFDConnection コンポーネントをフォーム上にドラッ グ&ドロップします。



配置した TFDConnection (FDConnection1)をダブルクリックすると、FireDAC 接続エディタが表示されるので、「ドライバ ID」項目で、接続するデータベースを選択します。今回は、「IB」(InterBase)を選びます。

🔞 FireDAC 接続エディタ -	[FDConnection1]	-		×
■ ドライバまたはオーバーラ	イドする接続定義の名前を選択してから	、パラメータをセ	ットアップし	ます
定義 オブション 情報	SQL スクリプト			
ドライバ ID(D):			\sim	
接続定義名(N):	DS FB		_^Ì	
ታスト(1)	18 MIDLite			
パラメータ	Mongo MSAcc			
	MSSQL		~	
		OK	キャンセ	2) (C)

ドライバ ID を選択すると、選択したドライバに応じて必要な設定パラメータが表示されます。InterBaseの場合、この中で設定が必要となるのは、Database、User_Name、Password、CharacterSet です。

三義 オブション 情報	SQL スクリプト			
ライバ ID(D):	IB			\sim
卷流定義名(N):				~
テスト(T)	ウィザード (W)	デフォルトに戻す(F	2) ヘルプ(ト	l)
パラメータ	値		デフォルト	1
DriverID	IB		IB	_
Pooled	False		False	
Database				
User_Name				
Password				
MonitorBy				
OSAuthent				
Protocol	Local		Local	
Server				
Port				
SQLDialect	3		3	
RoleName				
CharacterSet	NONE		NONE	
GUIDEndian	Little		Little	
ExtendedMetadata	False		False	
OpenMode	Open		Open	

Database には、データベースファイルの場所を指定します。C++Builder をインストールする際にサンプ ルファイルもインストールしていると、デフォルトで以下の場所に、今回使用する InterBase のデータベ ースファイルが保管されています。

C:\Users\Public\Documents\Embarcadero\Studio\21.0\Samples\Data\dbdemos.gdb

各項目は以下のように設定します。

項目	
Database	C:\Users\Public\Documents\Embarcadero\Studio\21.0\
	Samples\Data\dbdemos.gdb
User_Name	sysdba
Password	masterkey
CharacterSet	UTF8

embarcadero[®]

設定が完了したら、[テスト]ボタンをクリックして接続テストを行います。接続に成功すれば、設定 は完了です。[OK]をクリックして、接続エディタを閉じます。

定義 オプション 情報	SQL スクリプト				
ドライバ ID(D):	IB			~	
培结空差々かい					
テスト(T)	ウィザード(W)	デフォルトに戻す	(R) /	、 、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、	
パラメータ	値		デフォルト		
DriverID	IB		IB		
Pooled	False	~	False		
Database	C:¥Users¥	Public¥Documents			
User_Name	sysdba				
Password	masterkey	1			
MonitorBy					
OSAuthent					
Protocol	Local		Local		
Server					
Port					
SQLDialect	3		3		
RoleName					
CharacterSet	UTF8		NONE		
GUIDEndian	Little		Little		
ExtendedMetadata	False		False		
OpenMode	Open		Open		

このほかに、FDConnection1 に対して、いくつかのプロパティを設定します。

FDConnection1	
項目	值
Connected	True
LoginPrompt	False

これで、ユーザー名/パスワードを入力するログインプロンプトを表示することなく、アプリケーション起動と同時にデータベースに接続します。

TFDTable を使ってテーブルのデータセットを呼び出す

データベースのテーブルを扱うのは、TFDTable コンポーネントです。同じようなコンポーネントに TFDQuery があります。TFDQuery は、テーブルではなく、クエリー(SELECT 文)によってデータセッ トを取得する点が異なります。 embarcadero

ツールパレットから、TFDTable コンポーネントを選択し、フォーム上にドラッグ&ドロップします。

ウェルカムページ Unit1.cpp ×	~	Project1.cbp	roj - プロジェクト	₽ ×
Pormation i Pormation i Portuge i Aleman i Alema		Project Log Project Scope	(c)	↓ ~
		Project1.c	モデルビュー データェク	マルチデバ
		パレット		₽ ×
			imes fdt	Q
		V FireDAC		
		E TFDTr	ansaction	
		E TFDTa	bleAdapter	
	\neg	TFDTa	ble	

オブジェクト インスペクタで、配置した TFDTable(FDTable1)に対して、次のようにプロパティを設 定します。

FDTable1

項目	值
TableName	BIOLIFE
Active	True

TFDTable を配置したときに、このコンポーネントの Connection プロパティは、自動的に FDConnection1 に設定されているはずです。これにより、TableName プロパティを設定しようとすると、 テーブルの一覧を取得し、下図のようにドロップダウンリストでテーブルの一覧を表示します。

	オブジェクトイン	スペクタ	4	×
	FDTable1 TFDTab	le		\sim
	לעולדי אינ	ント		Q
	LocalSQL			^
	MasterFields			
	MasterSource			
	Name	FDTable1		
	ObjectView	✓ True		
> ResourceOption		(TFDBottomResourceOptions)		
	SchemaAdapte			
	SchemaName			
l	TableName	BIOLINE	\sim	
	Tag	ANIMAES		h
	Transaction	BIOLIFE		
	UpdateObject	COUNTRY		ľ
	フィールド エディタ… と	CUSTOLY		
	クイック編集…	EMPLOYEE		
	すべての項目が表示	INDUSTRY		
		ITEMS		



ユーザーインターフェイスを設計する

データセットの準備ができたら、次にユーザーインターフェイスを作成していきます。データセットを 表形式で表示、編集できる便利なグリッドコンポーネントを使いましょう。

ツールパレットから TDBGrid を選択し、フォーム上にドラッグ&ドロップします。



配置した TDBGrid (DBGrid1) に対して、以下のように Align プロパティを設定します。

DBGrid1	
項目	值
Align	alRight

Align プロパティを alRight に設定することで、フォーム左側の残りの部分に詳細データを表示するための領域を確保できました。ただ、ウィンドウの大きさによって、データの表示可能領域は変わってきてしまいます。そこで、左右の表示領域の大きさをユーザーが自由に変更できるようにしようと思います。

これを実現するのが、TSplitter コンポーネントです。TSplitter を使えば、左右の表示領域の分割位置をマウス操作で自由に変更できます。TSplitter は、フォームの一方の端に接しているコントロールと、それ以外のクライアント領域を占めているコントロールとの間に配置されます。ユーザーがスプリッタを移動すると、フォームの端に接しているコントロールのサイズが変化します。これによって、このフォームのクライアント領域が変化し、それに応じてクライアント領域の残りの部分を占めるコントロールのサイズが変化します。



ツールパレットから TSplitter を選択し、フォーム上にドラッグ&ドロップします。



配置された TSplitter (Splitter1)の Align プロパティを次のように設定します。

Splitter1		
項目	值	
Align	alRight	

これで、Splitter1は、右側のコントロール(DBGrid1)にくっつきました。

Splitter1の左側にはTPanelを配置します。最終的には、この上にいくつものUIコントロールを配置して、詳細データを表示するようにします。ツールパレットから TPanel を選択し、フォーム上にドラッグ&ドロップします。



配置した TPanel (Panel1)のプロパティを次のように設定します。

Panel1			
項目	值		
Align	alClient		
ShowCaption	False		

以上で分割線(スプリッタ)を持つユーザーインターフェイスが作成できました。後は詳細データを表 示するコントロールの配置です。これには、データベースのフィールドデータに関する便利な機能を使 ってみましょう。

データフィールド設定

TFDTable がデータベースのテーブルを表すのに対し、テーブルの各フィールド(列)は、TField(およびその派生クラス)によって表されます。TField は、2種類の方法で作成できます。

- **静的フィールド**:設計時に「フィールドエディタ」を使って定義する
- 動的フィールド:実行時に自動生成(TFDTable およびその他のデータセットの Fields プロパティ や FieldByName メソッドを使ってアクセス)

設計時に「フィールドエディタ」を使って定義すると、各フィールドのプロパティやイベントを定義す ることができます。例えば、Alignment プロパティを使ってデータの整列方法を変更したり、OnChange イベントを使って、フィールドデータが変更された時の処理を記述することができます。

「フィールドエディタ」では、フィールドを定義するだけでなく、定義したフィールドデータを表示す るコントロールをドラッグ&ドロップで簡単にフォーム上に配置することができます。今回は、この方 法で、詳細データを表示するコントロールを配置することにします。



フォーム上の FDTable1 を選択し、右クリックして [フィールドエディタ] メニューを選択します。



「フィールドエディタ」が表示されるので、右クリックしてメニューを表示し、 [すべてのフィールド を追加] を選択します。



すると、次のように BIOLIFE テーブルのすべてのフィールドが追加されます。



「フィールドエディタ」または「構造」ペインで、フィールドを選択すると、そのフィールドのプロパ ティやイベントが、「オブジェクト インスペクタ」に表示されます。これらの値を設定することで、設 計時にフィールドに関する設定やカスタマイズが可能になります。 embarcadero[®]

さて、ここからがマジックのようですが、「フィールドエディタ」にリストされているフィールドをす べて選択し、設計フォーム右側に位置する Panel1 の上にドラッグ&ドロップしてください。

Referent		
SPECIES_NO	<u>)</u>	
90020		
CATEGORY		
Triggerfish	× I	FSConnection 1 FDTable 1
COMMON_NAME		$\mathbf{\lambda}$
Clown Triggerfish		
SPECIES_NAME		
Ballistoides conspicillum		
LENGTH_CM_		
50	1	Form1.FDTable1 ×
LENGTH_IN		
19.6850393700787		SPECIES_NO
NOTES		CATEGORY COMMON NAME
Also known as the big spotted triggerfield. Inhabite outer reef		SPECIES_NAME
areas and feeds upon crustaceans		LENGTHCM LENGTH_IN
and mollusks by crushing them with		NOTES
eaters, and divers report seeing	DataSource1	GRAPHIC
GRAPHIC		
- ALARCERE		

たったこれだけの操作で、各フィールドに対応するラベル(TLabel)と入力ボックス(TDBEdit)、メモ (TDBMemo)、画像(TDBImage)が追加されます。これらのコントロールとデータセットをリンクす るための TDataSource も、**DataSource1** として追加されます。

DataSource とは

ドラッグ&ドロップで配置したコントロールには、データベースの実データが表示されています。この ように、設計時に実データを表示できるのも C++Builder の特長のひとつです。いちいちアプリケーショ ンを実行しなくても、データを表示するのに最適なレイアウトを選べますから便利ですね。

さて、先に配置した DBGrid1 には、まだデータが表示されていません。その理由は、DataSource プロパ ティが設定されていないからです。DataSource プロパティには、データセットとの仲介を行う TDataSource を指定します。今回は、BIOLIFE テーブルを表す FDTable1 に DataSource1 が結びつけら れています(DataSource1 は、先ほどのドラッグ&ドロップ操作で自動生成されました)。

DBGrid1の DataSource プロパティにも、同じ値を設定します。





レコードの移動と操作

グリッドとは異なり、詳細データでは、現在のレコードの情報のみが表示されます。レコードを移動さ せたり、データの更新などを行うには、データセットに対して、レコードの移動や更新を行うメソッド を呼び出す必要があります。

こうした処理をボタンに割り当て、イベントハンドラやアクションをいちいち定義するのは面倒です。 そこで、データセットの主要な操作をまとめたボタンセットのコンポーネント TDBNavigator を使います。 TDBNavigator を用いれば、データセットにリンクされた TDataSource を指定するだけで、上記のような 操作を実装できます。



ツールパレットから TDBNavigator を選び、フォーム上の DBImage1 画像の下あたりに配置します。

次のようにプロパティを設定すれば、ボタンが動作します。

DBNavigator1	
項目	值
DataSource	DataSource1

データにコードからアクセスするには

C++Builder では、ビジュアルコンポーネントとデータアクセスコンポーネントを結び付け、ユーザーイ ンターフェイスとデータをリンクし表示させることができます。コードを記述する必要がまったくない ことに驚かれたかもしれません。

しかし、実際のプログラムでは、データを操作したり、複雑な処理を実装する必要があるでしょう。 C++Builder のコンセプトは、簡単でありながらも高度なことも実現できるようにすること。実は、ここ で使用してきたデータアクセスコンポーネントについても、コードによって直接操作できるのです。こ の手法を用いれば、プログラム内で直接レコードのデータを取得したり、操作することもできます。

ここでは、データアクセスコンポーネントをプログラムコードによってアクセスし、表示中の画像を、 ファイルとして保存できるようにしてみましょう。

ファイル保存用のダイアログを用意する

まずは保存ダイアログコンポーネント TSaveDialog を配置します。ツールパレットの「Dialog」から TSaveDialog を選び、フォーム上の DBImage1 画像の下あたりに配置します。



配置した TSaveDialog (SaveDialog1) のプロパティを次のように設定します。

SaveDialog1	
項目	
DefaultExt	bmp
Filter	ビットマップファイル *.bmp
Options	[ofOverwritePrompt,ofHideReadOnly,ofEnableSizing]
	ofOverwritePrompt:既存のファイルを上書きするかどうかを尋ねます
	ofHideReadOnly:ダイアログから[読み出し専用ファイルとして開く]
	チェック ボックスを削除します
	ofEnableSizing:ダイアログのサイズを更できるようにします

画像を保存するコードの実装

画像のダブルクリックに応答して画像ファイルを保存するイベントハンドラを実装します。DBImage1 を 選択し、オブジェクトインスペクタで「OnDblClick」イベントを設定します。



```
イベントハンドラのコード全体は次のように記述します。
```

```
void __fastcall TForm1::DBImage1DblClick(TObject *Sender)
{
    // レコードから項目を取得し、SaveDialogにファイル名として設定する
    UnicodeString FileName = FDTable1->FieldByName("COMMON_NAME")->AsString;
    SaveDialog1->FileName = FileName;

    // ダイアログを表示して、画像の保存先を指定する
    if (SaveDialog1->Execute()) {
        // 画像ファイルを保存する。
        DBImage1->Picture->SaveToFile(SaveDialog1->FileName);
    }
}
```

このコードでは、レコードから画像ファイルと画像ファイル名を取得し保存しています。

'COMMON_NAME'フィールドから値を取得して保存先のファイルとし **DBImage1** の Picture プロパティの SaveToFile メソッドを呼び出せば、SaveDialog1 で指定した保存先に画像が保存されます。

この結果、レコードから取得したデータをローカル保存先に画像として保存することができました。

Windows 10 スタイルの画面にしてみよう

以上で「魚図鑑」アプリケーションは完成ですが、最後にウィンドウの表示スタイルをカスタマイズしてみましょう。C++Builder には、スタイルと呼ばれる UI スタイル変更のしくみが用意されており、あらかじめ用意されたスタイルを切り替えるだけで、表示をモダンなイメージにしたり、ダークで落ち着いた感じにしたりと、カスタマイズできます。

今回は、Windows 10 スタイルを適用してみましょう。メインメニューから[プロジェクト|オプション] を選択し、「プロジェクト オプション」ダイアログを表示します。 embarcadero

ここで、左側のツリーから「アプリケーション」下の「表示」を選択します。そして、「カスタムスタイル」項目で、「Windows 10」を選択し、その下の「デフォルトスタイル」で「Windows 10」を選択し ます。

🤒 Project1.exe のプロジェクト オプショ	> (Win32 - Debug)	٩	×
 C+++ リンカ 出力 	^ 表示		
警告 V Delphi コンパイラ	アプリケーションの設定		
コンパイル ドンドと警告	915UD		
リンク	ヘルプ ファイル(E)		
出力 - しん++		参照(B)	
ディレクトリと条件定義 Turbo アセンブラ	カスタム スタイル		
ディレクトリと条件定義	Ruby Graphite	^	
警告	Sapphire Kamri		
ビルドイベント	Silver		
ビルド順序	Sky		
	Slate Classico		
表示	Smokey Quartz Kamri		
マニフェスト	Tablet Light		
アイコン	TabletDark		
フォーム	Turquoise Gray		
バージョン情報	Windows10		
◇ テバッガ	Windows10 Blue		
シンボル テーフル	Windows10 Dark		
環境ブロック	Windows10 Green		
◇ パッケージ	Windows10 Purple		
実行時パッケージ	Windows10 SlateGray	\sim	
✓ プロジェクト プロパティ			
一般) / / Ca-(P)		
Getit 依存パッケージ			~
◇ 配置	Windows10	\sim	
プロビジョニング			J
	(保存) キャンセル	ヘルプ	

[OK] ボタンをクリックして、ダイアログを閉じます。

[実行 | 実行] メニューを選択してアプリケーションを実行すると、次のように Windows 10 スタイルで UI が表示されます。

SPECIES_NO SPE 90020 CATEGORY Triggerfish Cownroupcilum ENVITH_ON_ 59 LENCITH_IN 18VITH_ON_ 59 LENCITH_IN 18VITH_ON_ 59 LENCITH_IN 10455033707077 NOTES Also how on a the big spotted sportfall tenth up are various eaters, and diver report seeing GRAPHIC	CIES_NO CATEGORY 90020 Triggerfish 90030 Snapper 90050 Wrasse 90050 Wrasse 90070 Angefish 90080 Cod 90090 Scorpionfish 90100 Butterflyfish 90110 Shark 90120 Ray 90130 Cod 90150 Sculpin 90150 Sculpin 90150 Spadefish 90170 Shark	COMMON LIAME Clown Troggerfah Red Emperor Gast Naoni Wrasse Blue Angelfah Lunartal Rockood Freefah Ornate Butterfrifah Swell Shark Bat Ray California Morey Lingcod Cabecon Adamic Spodefah
90020 CATEGORY Trogerfish COMMON_WAVE COmmonserfish SPECIES_NAVE BalandorSecondulum LENGTH_CM_ 50 LENGTH_CM_ 50 LENGTH_CM_ 50 LENGTH_CM_ 60 CATEGORY 60 CATEG	90020 Triggerfah 90030 Snapper 90050 Wrasse 90070 Angeffah 90080 Cod 90090 Scorpionfah 90100 Butterflyfah 90110 Shark 90120 Ray 90130 Eel 90140 Cod 90150 Sculpin 90150 Sculpin 90150 Spadefah 90170 Shark	Clown Triggerfah Red Emperor Glant Maori Wrasse Blue Angeffah Lunartal Rockod Frefah Ornate Butterftyfish Swel Shark Bat Ray California Moray Lingcod Cabezon Alantic Spadefah
ATEGORY Trogerfish COMMON LAWE COMMON LAWE COMMON LAWE COMMON LAWE COMMON LAWE Ballstodes compticitum ENGTH_CM	90030 Snapper 90050 Wrasse 90070 Angelfah 90080 Cod 90090 Scorpionfah 90100 Butterfyfish 90110 Shark 90120 Ray 90130 Eel 90140 Cod 90150 Sculpin 90160 Spadefah 90170 Shark	Red Emperor Giant Maori Wrasse Blue Angelfah Lunartall Rockcod Firefish Ornate Butterflyfish Swell Shark Bat Ray Calfornia Moray Lingcod Cabezon Atlantic Spadefish
Inggerfsh COMMON JAVAE COMMON JAVAE COMMON JAVAE COMMON JAVAE Debitodes congolum ENGTH_CM	90050 Wrasse 90070 Angelfish 90080 Cod 90090 Scorpionfish 90100 Butterflyfish 90110 Shark 90120 Ray 90130 Eel 90140 Cod 90150 Sculpin 90150 Spadefish 90170 Shark	Giant Maori Wrasse Blue Angelfish Lunartali Rockcod Firefish Ornate Butterflyfish Swell Shark Bat Ray Calfornia Moray Lingcod Cabezon Atlantic Spadefish
MACULAR Common Usake Common	90070 Angefish 90080 Cod 90090 Scorpionfish 90100 Butterflyfish 90110 Shark 90120 Ray 90130 Eel 90140 Cod 90150 Sculpin 90160 Spadefish 90170 Shark	Blue Angelfish Lunartal Rockcod Firefish Ornate Butterflyfish Swell Shark Bat Ray California Moray Lingcod Cabezon Atlantic Spadefish
UMUUT_vare: Join Triggerfsh. TECES_JUME Bistodes corgodum SNGTL_CMS0 ENCTL_INS0	90080 Cod 90090 Scorpionfish 90100 Butterflyfish 90110 Shark 90120 Ray 90130 Eel 90140 Cod 90150 Sculpin 90160 Spadefish 90170 Shark	Lunartail Rockcod Firefish Ornate Butterflyfish Swell Shark Bat Ray California Moray Lingcod Cabezon Atlantic Spadefish
CRES_NAME eleficities comparison NGTH_OM	90090 Scorpionfish 90100 Butterftyfish 90110 Shark 90120 Ray 90130 Eel 90140 Cod 90140 Sculpin 90160 Spadefish 90170 Shark	Firefish Ornate Butterflyfish Swell Shark Bat Ray California Moray Lingcod Cabezon Atlantic Spadefish
HeatIng SetUp NCTHOM	90100 Butterflyfish 90110 Shark 90120 Ray 90130 Eel 90140 Cod 90150 Sculpin 90160 Spadefish 90170 Shark	Ornate Butterflyfish Swell Shark Bat Ray California Moray Lingcod Cabezon Atlantic Spadefish
NGTH_CM	90110 Shark 90120 Ray 90130 Eel 90140 Cod 90150 Sculpin 90150 Spadefish 90170 Shark	Swell Shark Bat Ray California Moray Lingcod Cabezon Atlantic Spadefish
NCH_CM	90120 Ray 90130 Eel 90140 Cod 90150 Sculpin 90160 Spadefish 90170 Shark	Bat Ray California Moray Lingcod Cabezon Atlantic Spadefish
50 NGTH_JN 15.6650393700787 TTES like Involutes at the big spotted rigger fishInvolutes outler reef rigger fishInvolutes outler reef rises and fields upon utsleacens atters, and divers report seems Advert LebchThere is even atters, and divers report seems	90130 Eel 90140 Cod 90150 Sculpin 90160 Spadefish 90170 Shark	California Moray Lingcod Cabezon Atlantic Spadefish
NGTH_IP. I9-6850/93700767 TES Iso known as the big sported genfal, Invalued souther reaf of mail labels, They are vanadious stars, and divers report seeing APHIC	90140 Cod 90150 Sculpin 90160 Spadefish 90170 Shark	Lingcod Cabezon Atlantic Spadefish
19.6850393700787 TES to known as the big spotted gogerfish. Inhabits outer reef and moluske by cushing them with were ful keth. They are violations kethy, and diversing in the sethy ArHitC	90150 Sculpin 90160 Spadefish 90170 Shark	Cabezon Atlantic Spadefish
TES TES as the bigs potted goeffah. Inholds sufer receives of endulately oruphing them with week liteth. They are voradous stars, and dwars report seeing APHIC	90160 Spadefish 90170 Shark	Atlantic Spadefish
ics is nown as the big spotted goeffin. Inhabits outer reef es and fleed, your outscaren werful tests. They are voracious tera, and divers report seeing APHIC	90170 Shark	
a workin. The hadron government of the set of the set o		Nurse Shark
effect and relations of the second se	90180 Ray	Spotted Eagle Ray
d moluska by cuthing them with here, and dwars report seeing APHIC	90190 Snapper	Yellowtail Snapper
White Contract seeing	90200 Parrotfish	Redband Parrotfish
	90210 Barracuda	Great Barracuda
APHIC	90220 Grunt	French Grunt
	90230 Snapper	Dog Snapper
	90240 Grouper	Nassau Grouper
	90250 Wrasse	Bluehead Wrasse
	90260 Jack	Yellow Jack
	90270 Surfperch	Redtail Surfperch
	90280 Croaker	White Sea Bass
	90290 Greenling	Rock Greenling
	90300 Wrasse	Senorita
	90310 Smelt	Surf Smelt
$\Box \lhd \rhd \bowtie + - \bigtriangleup \checkmark \times \circlearrowright$		

レコードの移動、データの更新などができることを確認してみましょう。

embarcadero[®]

Embarcadero、Embarcadero Technologies ロゴならびにすべてのエンバカデロ・テクノロジーズ製品またはサービス名は、 Embarcadero Technologies, Inc.の商標または登録商標です。その他の商標はその所有者に帰属します。