Reviewer Guide

embarcadero®

RAD Studio XE7
機能評価ガイド

エンバカデロ・テクノロジーズ

2014 年 9 月

目次

概要 1
主な機能と開発者にとってのメリット 3
FireUI：マルチデバイスデザイナおよび UI コンポーネント 3
アプリケーション実現の一番の近道 4
マルチデバイスデザイナ 4
マルチデバイスデザイナの技術的詳細 5
FireMonkey ビヘイビアサービス API 6
新しい MultiView コンポーネント 7
FireMonkey のその他の機能強化 8
プラットフォームネイティブなコントロール 8
Android 向けの機能強化 8
パラレルプログラミングライブラリ 9
並列 For ループ 10
タスクとフューチャー 11
EMS（Enterprise Mobility Services）の概要 12
EMS のアーキテクチャ 13
すぐに使えるインフラストラクチャ、ユーザーの API、分析機能 13
開発ツールでのカスタマイズと使用する RDBMS の選択 14
HTTPS と InterBase ToGo によるセキュリティの確保 15
ホスティング先は自由（柔軟なライセンスプラン） 15
実際の EMS ソリューション開発の手順 16
EMS サーバー パッケージの作成 16
サーバーのデバッグ 18
EMS クライアントの作成 19
コンソールで提供される分析機能 21
EMS とエンタープライズデータアクセス 21
Bluetooth とアップテザリングを使った ガジェットやウェアラブルとの接続 23
アップテザリング 23
Bluetooth と Bluetooth LE 24
開発者の生産性向上に関する新機能 24
IDE の新機能 25
Object Pascal 言語の新機能 25
RTL の機能強化 26
データベースアクセスと FireDAC の機能強化 26
VCL の機能強化 27
RAD Studio XE7 追加情報 28
XE7 製品ラインナップ 28

概要

このたびは、RAD Studio XE7 を評価いただき、誠にありがとうございます。RAD Studio XE7 は、エンバカデロ・テクノロジーズが提供するフル機能のソフトウェア開発ソリューションで、Windows、Mac、iOS、Android 向けの真のネイティブアプリケーションの構築に加え、ガジェットやウェアラブルデバイスと接続するアプリケーションの構築もサポートしています。この評価ガイドでは、RAD Studio XE7 を使用した場合の主なメリットをいくつか解説し、マルチデバイスアプリケーションの構築にも Windows アプリケーションの構築にも利用できる新機能を紹介します。

現在、多くの企業が、Android や iOS といった OS を含むモバイルデバイスを筆頭に、さまざまなオペレーティングシステムやプラットフォーム上で、オフィスでも外出先でも使用できるアプリケーションを提供する必要に迫られています。しかし、これらのアプリケーションをプラットフォーム固有のツールでコーディングしメンテナンスしていくには、費用も時間もかかってしまいます。RAD Studio XE7 を利用すれば、開発者の生産性を向上させることができ、開発チームは単一のツール、単一のプログラミング言語、単一のフレームワークを使って Windows、Android、iOS、Mac OS X 向けのマルチデバイスアプリケーションを極めて短時間で開発できます。開発したアプリケーションは、実行時ランタイムにはまったく頼らずに、各サポート対象のプラットフォームで高いユーザーエクスペリエンスを提供するネイティブかつ安全なアプリケーションを作成することができます。

このようなマルチデバイス開発とネイティブコードへのコンパイルの組み合わせは他に類を見ないもので、特に XE7 では大幅な機能強化がなされています。その結果、最高のアプリケーションパフォーマンスを実現するうえで障害となるスクリプト言語や仮想マシンを使用せずに、すばやく開発できるようになります。

しかも、RAD Studio XE7 では新たに Bluetooth をサポートしたため、使用可能な何千ものガジェットやウェアラブルデバイスとやり取りできるアプリケーションの構築も実現しました。この Bluetooth サポートにより、XE6 で導入されたアップテザリングも拡張されました。アップテザリングは、モバイルデバイスに接続されたアプリケーションを容易に作成できる技術で、開発者は既存の Windows VCL アプリケーションをモバイルへと拡張できるようになりました。アップテザリングでは、Wi-Fi ネットワーク接続をサポートしていましたが、今回、Bluetooth もサポートするようになりました。

どのソフトウェア開発者もソフトウェア開発企業もモバイルデバイス、そして成長著しいガジェットやウェアラブルデバイスを活用したいというユーザーの需要を無視することはできませんが、一方で Windows プラットフォームで動作するミッションクリティカルなビジネス アプリケーションにも、継続して多額の投資が行われているのも事実です。モバイル アプリケーションは企業の成功に必要不可欠ですが、ミッションクリティカルな Windows アプリケーションを、最新の Windows バージョンや、よりモダンな UX（ユーザー エクスペリエンス）パラダイムへと更新することも急務です。

加えて、ハードウェアの進化に目を向けると、マルチコア CPU がデスクトップでもモバイルにも搭載されるようになり、非同期処理が標準的に利用されるようになってきました。XE7 では、新しい並列プログラミングライブラリが導入されており、マルチコア CPU を活用した並列処理に対応しています。このライブラリは、スレッドプールマネージャのほか、並列 "for" ループ、タスクスケジューリング、フューチャーなどの、すぐに使用できる機能を備えています。

RAD Studio XE7 は、PC、タブレット、スマートフォン、ガジェット、ウェアラブル向けの真のネイティブ アプリケーションを作成し、それらをすばやくマーケットに投入可能にします。さらに、Windows VCL アプリケーションへの投資をこれまでどおり続ける企業、あるいは Windows を中心とした新しいソリューションを構築する必要のある企業にとっても最適なアプリケーション開発スイートです。RAD Studio XE7 を使用する場合、管理対象は単一のコードベース、単一のチーム、単一のスケジュールですが、パフォーマンスとユーザー エクスペリエンスを犠牲にせずに複数のプラットフォームをサポートできます。

RAD Studio XE7 では、以下の 2 つのプログラミング言語をサポートしています。

- 強力な言語機能を利用できる C++ 言語（C++11 標準に対応しアップデート）
- より使いやすくなった最新の Object Pascal 言語（C# などで採用されている新しい言語機能も搭載）

この評価ガイドは RAD Studio XE7 の機能評価の出発点であることにご留意ください。RAD Studio XE7 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。巻末には、その他のリンク一覧を記載しています。

お問い合わせ先

RAD Studio XE7 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：03-4577-4520　Email： japan.info@embarcadero.com

主な機能と開発者にとってのメリット

RAD Studio（および C++Builder、Delphi）は開発ツール市場で長きにわたり揺るぎない存在感を示してきましたが、近年のマルチプラットフォームサポートにより、その機能は大幅に拡張されています。現在では、1 ダースあまりのターゲットプラットフォームをサポートするネイティブコンパイラ、広範なクロスプラットフォームランタイムライブラリ、エンタープライズクラスのデータベースアクセスサポート、従来からの VCL Windows UI ライブラリに加えて、マルチデバイスをサポートする FireMonkey（FMX）プラットフォームが提供されています。

これらの機能をすべて 1 つのドキュメントで詳細に説明しようとすると膨大な量になりますし、RAD Studio のオンラインドキュメントには、何千ページものチュートリアルと何万ページもの API ドキュメントが含まれています。したがって、本書では、中核となる機能のいくつかを主に取り上げ、バージョン XE7 で導入された新しい機能にフォーカスすることにします。この機能評価ガイドに加えて、旧バージョン向けの機能評価ガイドも、同じく参考になるでしょう。

FireUI：マルチデバイスデザイナおよび UI コンポーネント

RAD Studio のこれまでのバージョンでは、コンポーネントの位置揃えと相対位置設定を使ってビジュアルフォームを設計できることが、マルチデバイス向け開発の基礎となっていました。これにより、コンポーネントを、スマートフォン、タブレット、デスクトップといったさまざまなオペレーティングシステムやフォームファクタに適合できたほか、特定のデバイスごとに完全に別個の UI を作成することも可能でした。IDE では、さまざまなオペレーティングシステムのさまざまなフォームファクタやスタイルをプレビューできましたが、共通のフォームビューで行われた変更は他のすべてのデバイスビューに反映されました。それらのすべてについて、フォームストリームファイル（FMX）とオブジェクトインスタンスが 1 つしかなかったのはそのためです。

開発者は別個のフォームを使って UI を別々に作成することもできましたが、その場合は、UI の大部分はフォーム間で似ているにもかかわらず、別個のイベント処理コードが記述された複数のフォームを管理する結果になりました。

もう 1 つの問題は、一部のコンポーネントについては、ベースとなるプラットフォームに自動的に適合できないことでした。そのため、例えば、iOS と Android とでタブ位置を変得なければならないといった場合には、開発者がプラットフォーム固有のコードを記述して、それぞれのシナリオに対処しなければなりませんでした。

RAD Studio XE7 では、こうした問題を解決する新機能が搭載されています。これらの新機能を紹介する前に、RAD Studio による開発における全般的な利点を見ていきましょう。

アプリケーション実現の一番の近道

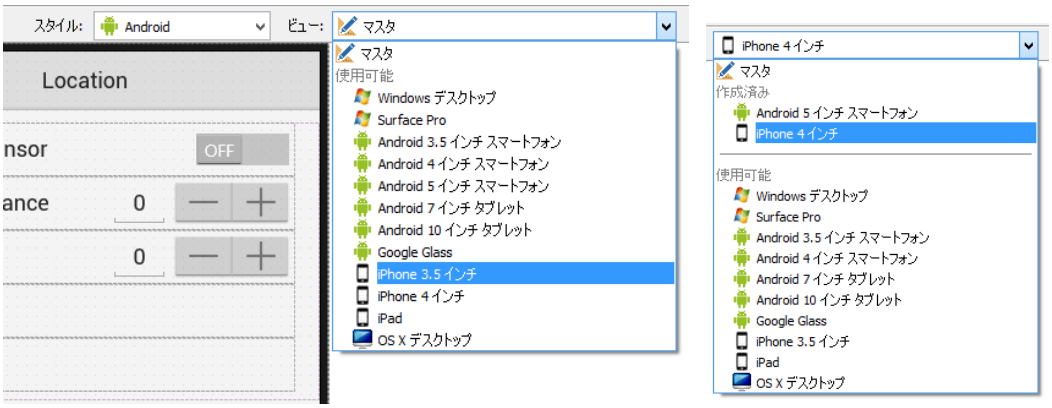
RAD Studio を利用すれば、実際に動作するプロトタイプを作成し、ユーザーのフィードバックを取り込んで、アプリケーションをマーケットにすばやく投入することができます。大半のビジュアルプロトタイピング環境では、外見が同じだけで実際には動作しないモックアップが作成されるだけで、使って体験できるプロトタイプは生成されません。つまり、顧客とチームメンバーはアプリケーションのコンセプトの実体験を得ることはできないのです。それに加えて、開発者もプロトタイピングののちに本開発に進む際には、再びゼロから開発を始めなければなりません。

RAD Studio でのビジュアル開発では、実データまたはプロトタイプ用のデータを扱うビジュアルモックアップを、開発者や設計者がコードを記述せずにすばやく作成し、実際のターゲットデバイス（PC、スマートフォン、タブレット）に配置したり、Windows や Mac 上でシミュレートしたりできます。これにより、顧客とチームメンバーは、はるかに正確で印象的なプロトタイプを体験することができます。RAD Studio で作成するプロトタイプでは、実際の開発で使用するフレームワークオブジェクトを使用するので、開発者は労力を無駄にせずにプロトタイピングから本開発に進むことができ、開発時間とリソースの節約につながります。

マルチデバイスデザイナ

XE7 の画期的な新機能として、さまざまな種類のデバイスやフォームファクタ向けに最適化可能な単一の共有 UI を構築できる方法が導入されました。XE7 では、共有 UI をすばやく作成したら、デスクトップフォームファクタとモバイルフォームファクタなどの複数のビューを使って、それぞれのデバイス向けに最適化を行うことが可能になりました。まず、中核となる UI を「マスター」ビューとして作成します・マスタービューは、サポートされているあらゆるプラットフォーム向けにスタイル設定できます。あるデバイスやオペレーティングシステムに特化したビューを作成する際に、デバイスの種類ごとに UI コンポーネントを（さらにそれ以外の非ビジュアルコンポーネントさえも）配置したり、プロパティを変更したりして、デバイス固有のカスタマイズができるようになりました。

次の図には、使用可能な定義済みデバイスフォームファクタのリストが表示されています。これらのリストのいずれのフォームファクタもアクティブにすることができ、そのビューがいったんアクティブになると、図の右側にあるように、それが先頭セクションに表示されます。



また、図の左側で、マスタービューの UI スタイルを選択していることにも注意してください。この図の場合は［Android］を選択しており、設計画面におけるマスタービューの表示が Android デバイスのスタイルになっています。

特定のビューで生じた差異は、2 番目の FMX ファイルに保存されます。このファイルには、特定のビューとマスタービューの差分のみが格納されます。つまり、マスターに変更を加えることで、その変更内容は、対象の値をオーバーライドしていない各ビューに自動的に反映されます。一方、各ビューでは、コンポーネント個々のプロパティやコンポーネント全体をマスター値に戻すこともできます。

つまり、マスターには、すべてのビューに反映されるべき変更をいつでも加えることができ、特定のビューには、そのビューにのみ最適化された変更を加えることができるのです。デバイスごとにレイアウトをカスタマイズできるだけでなく、プラットフォームごとに必要に応じてカスタムスタイルを読み込むこともできます。

それでも、必要な「フォームクラス」またはオブジェクトインスタンスはすべてのフォームファクタにわたって 1 つだけで済むので、開発者は単一のソースファイルを作成し、すべての共通 UI イベントをその 1 つの共有ソースファイルで管理することができます。

この機能により、デバイス固有のビューをカスタマイズ／最適化できるので、真に単一ソースのマルチプラットフォーム開発ソリューションとなり、コード中に、プラットフォーム固有の IFDEF 指令を記述しなければならない状況はいっそう少なくなります。これらのビューにおいては、そのコンポーネントのプロパティの値、位置、表示／非表示は、それぞれ異なる可能性があり、イベントに関連付けられているハンドラさえ、ビューによって異なることがある点に注意してください。

この画期的なデザイナは、現在開発者が使用可能な他のあらゆるモバイルフォームデザイナよりもはるかに強力で、しかも、デスクトップ OS とモバイル OS の両方を網羅しています。

マルチデバイスデザイナの技術的詳細

技術的観点から言えば、最終的に、さまざまなプラットフォームやフォームファクタ用の複数の FMX ファイルができあがることになりますが、コンパイラは、その最適化処理により、特定のプラットフォーム用のビューのみリンクします。これにより、たとえば、iOS アプリケーションには、iPhone 用のビューと iPad 用のビューがコンパイルされ実行可能ファイルに含まれますが、Android 固有のファイルは含まれません。

次のように、実際のコードでは、メインのフォームリソースファイルとビュー固有の差分がすべてリストアップされていますが、コンパイラでは、$R 指令の新しいプラットフォームオプションに基づいて、リソースを選別します。

```
implementation

{$R *.fmx}
{$R *.LgXhdpiPh.fmx ANDROID}
{$R *.iPhone4in.fmx IOS}
```

これらの「差分リソースファイル」はどのようなものなのでしょうか。非常に簡単な例を次に示します。

```
inherited PForm_iPhone4in: TPForm_iPhone4in
  ClientHeight = 548
  ClientWidth = 320
  inherited ToolBar1: TToolBar
    Size.Width = 320.000000000000000000
    Size.Height = 44.000000000000000000
    inherited Label2: TLabel
      Align = Top
      ...
    end
  end
end
```

XE7 には、普及している多数のモバイルデバイス、タブレット、デスクトップの各フォームファクタ向けの定義済みビューが付属しているほか、独自のビューを追加することもできます。社内で使用されている特定のデバイスをターゲットにする必要がある場合、開発者がそれを、ピクセル数やピクセル密度のほか、画面のサイズ、向き、縦横比などを含め、特定のフォームファクタを IDE に追加することができます。追加された新しいビューは、定義済みビューと一緒に表示されます。

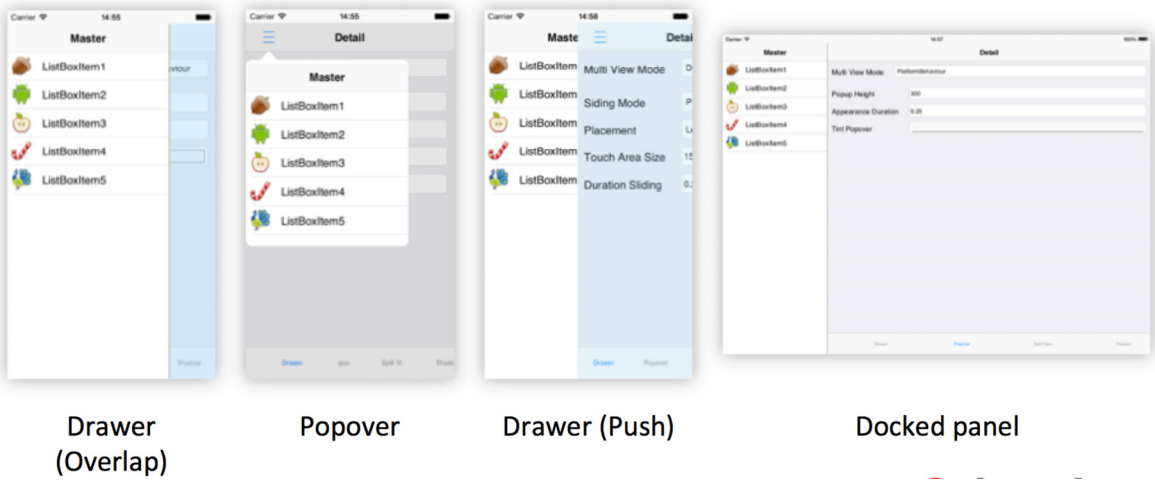
FireMonkey ビヘイビアサービス API

エンバカデロが FireUI として提供している新機能はこれだけではありません。2 つ目の機能は、プラットフォーム固有の情報をビジュアルコンポーネントに提供する FireMonkey ビヘイビアサービス API の導入です。この機能を使用することで、開発者がコードを記述することなくターゲットデバイスの種類ごとに UI 要素の位置を自動調整できます。たとえば、タブの位置が Android では画面の上部なのに対して iOS では画面の下部になるといった調整を自動化できるのです。

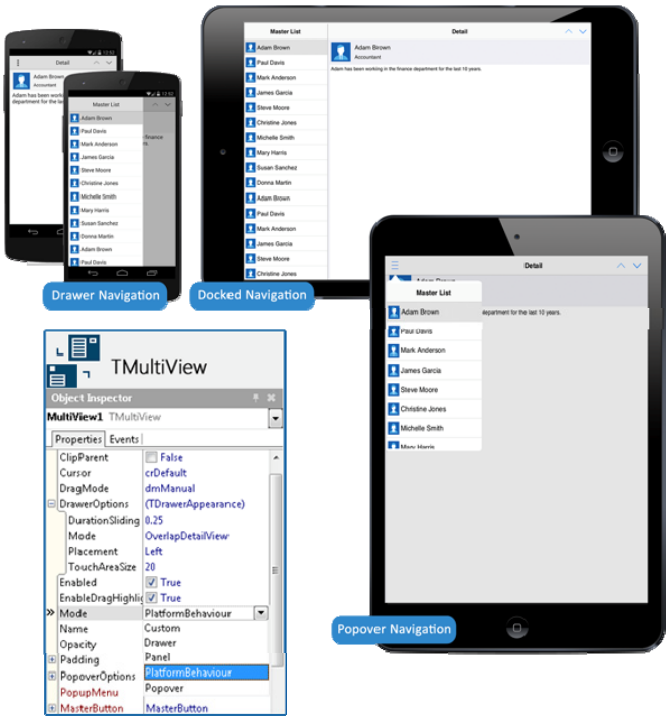
開発者は、コンポーネントのプロパティに対してプラットフォームごとに特定の値を設定するのではなく「PlatformDefault（プラットフォームデフォルト値）」を選択するだけでよいのです。

新しい MULTIVIEW コンポーネント

FireUI の機能として最後に紹介するのは、MultiView という新しいコンポーネントです。これは、デバイスのサイズと向きに応じて、ポップアップ、ポップオーバー、ドローワーのように動作できるコンポーネントです。ひとつの MultiView コンポーネントに同じデフォルト設定をした状態で、iOS のさまざまなフォームファクタ、画面向きで、どのように表示されるのかを、次の図に示します。



次の図は、MultiView コンポーネントの設定方法と、その結果、Android でどのように表示されるのかを示しています。



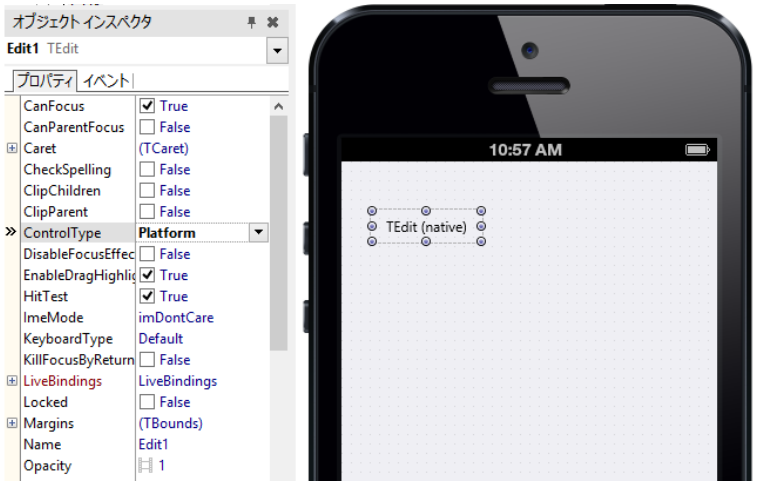
FireMonkey のその他の機能強化

新しいデザイナと、マルチデバイス開発用のいくつかの拡張機能は、XE7 の中核的な新機能となります。しかし、その他にも、特定プラットフォーム向けのものも含め、FireMonkey ライブラリの中核的な機能強化がいくつか施されています。

実は、FireMonkey では、マルチデバイス対応の単一ソースを実現できるとはいえ、すべてのプラットフォームに共通する最低限の標準機能で妥協しているわけではありません。FireMonkey には、プラットフォーム固有の機能も幅広くサポートしており、他のマルチデバイス対応フレームワークでは直接使用することのできないネイティブ対応のコンポーネントや 3D コントロールセットなども提供されています。

プラットフォームネイティブなコントロール

プラットフォーム統合の改善および強化の例として、iOS プラットフォーム上の TEdit コントロールが新しいプレゼンテーションアーキテクチャをサポートするように書き直されました。このアーキテクチャでは、コントロールで使用するプレゼンテーションが標準の FireMonkey スタイルのプレゼンテーションかネイティブプラットフォームコントロールのプレゼンテーションかを選択できます。後者の場合は、実行時にライブラリがネイティブプラットフォームコントロールをインスタンス化しレンダリングします。このコントロールのプロパティと動作は、それに対応する FireMonkey スタイルのコントロールと同じです。その結果、単一ソースによる開発を断念せずにプラットフォームコントロールを使用できることになります。



これと同じプラットフォームネイティブサポートは iOS 上の Calendar コンポーネントでも使用できます。

ANDROID 向けの機能強化

主に Android 向けの開発に的を絞った機能強化には、次のものがあります。

- Google Play Services SDK の Google Mobile Ads API を使用するように改良された TBannerAd コンポーネント。AdMob API をベースにした前バージョンからの移行は完全にシームレスで、同じコンポーネントで iOS プラットフォーム上の iAd もサポートしています。

- Android 4.4 KitKat での没入型フルスクリーンモードと Android スプラッシュ画面の IDE レベルでのサポート
- Android APK への Java ライブラリの追加を「プロジェクトマネージャ」レベルでサポート
- プッシュ通知と広告機能を Android マニフェストレベルで構成するためのプロジェクトオプション（従来はマニフェストファイルを直接編集）
- Java Android クラスの Object Pascal インターフェイスを生成する新規ツール Java2OP（Java To Object Pascal）。これはダウンロードで別途入手可能
- Google Glass に特化したスタイルと設計時画面により、現行の Google Glass サポートをさらに強化（モバイルアプリケーションが次世代のウェアラブル デバイスに対応可能）
- iOS および Android での ListView コントロールのプルリフレッシュ（Pull-to-Refresh）機能
- デスクトッププラットフォームでの複数モニタの管理の向上

パラレルプログラミングライブラリ

Delphi（Object Pascal）および C++による RAD Studio での開発における中核的な特長の 1 つは、生成される実行可能ファイルがコンパイル済みのコードであるということです。IDE には、Win32、Win64、Mac OS X、iOS ARM、Android ARM、さらには Mac OS X 上の iOS シミュレータをターゲットとする数多くの異なるコンパイラが搭載されています。

ARM コンパイラと Intel CPU 向け C++コンパイラの一部は LLVM アーキテクチャに基づいています。つまり、ARM プラットフォームでは、エンバカデロのコンパイラは Apple の Xcode と同じバックエンドジェネレータを使用するのです。

一方、Android 向けには、NDK ベースのアプリケーションを生成するいくつかのソリューションの 1 つを提供しています（SDK Java ライブラリへのバインディングを完全にサポート）。ネイティブバイナリには、実行速度の高速化、開発者による制御の強化、セキュリティの向上（ランタイムに依存しない場合）、リバースエンジニアリングの危険性の減少など、多数の利点があります。

実行速度の観点では、今日のデスクトップコンピュータやモバイルデバイスがマルチコアであることを生かしたマルチスレッドアプリケーションを作成できることからも、ネイティブコンパイラは有利です。バックグラウンドスレッドを使用する 1 つの理由は、（モバイルではますます一般的になってきている）リモートのインターネットリソースへの接続のような低速の操作をアプリケーションで実行する必要がある場合に、UI の応答性を高めることです。ただし、非同期コードやマルチスレッドアプリケーションの作成は、多くの開発者にとって一筋縄ではいかない場合があります。

VCL や FireMonkey を使用した Delphi（Object Pascal）および C++アプリケーションの開発者が、この種の開発を簡単に行えるように、XE7 では、新しいパラレル（並列）プログラミングライブラリを導入

しました。これは、マルチコア CPU を十分に活用したアプリケーションを作成してパフォーマンスを飛躍的に向上させることと、非同期コードをはるかに容易に開発できるようにすることを目的としたものです。

新規の System.Threading ユニットを使用することで、以下が可能になります。

- 並列 "for" ループ
- タスクのスケジューリング
- スレッド プールのセルフチューニング
- フューチャー

並列 FOR ループ

これらの機能を実際のコードでどのように使用すればよいのでしょうか。（かなり単純なシナリオで）これらの機能を示す Object Pascal コードサンプルを、以下にいくつか示していきましょう。

最初のコードには、従来の for ループと、（無名メソッド呼び出しに基づいた）同等の並列 for ループがあります。これらは、多数の数の集合に対して独立に計算を実行します。

```
for I := 1 to Max do
begin
  if IsPrime (I) then
    Inc (Tot);
end;

TParallel.For(1, Max, procedure (I: Integer)
  begin
    if IsPrime (I) then
      InterlockedIncrement (Tot);
  end);
```

このコードは、デスクトップでもモバイルでも、4 コア CPU の場合、容易に 3 倍速いパフォーマンスで実行することができます。実際、Windows 関数 InterlockedIncrement へのネイティブ呼び出しを、クラスメソッド TInterlocked.Increment プラットフォーム非依存の呼び出しに置き換えるだけで、同様のコードをマルチプラットフォームアプリケーションで記述できます。

タスクとフューチャー

このユニットのもう 1 つの機能は、タスクを定義できる TTask クラスです。既存プール内のスレッドへのタスクの割り当てはライブラリによって自動的に行われるので、次のコードに示すように、スレッドを直接使用する場合よりも開発が容易で、最適化かつ抽象化されます。

```
var
  aTask: ITask;
begin
  aTask := TTask.Create (procedure ()
    begin
      sleep (3000); // 3 seconds
      ShowMessage ('Hello');
    end);
  aTask.Start;
```

これは本当に瑣末なデモですが、非常に理解しやすいでしょう。もっと複雑なシナリオでは、複数のタスクを作成し、それらの全部または一部が完了するのを待つことがあります。以下にもう 1 つコードサンプルを示します。

```
var
  tasks: array of ITask;
  value: Integer;
begin
  Setlength (tasks ,2);
  value := 0;

  tasks[0] := TTask.Create (procedure ()
    begin
      sleep (3000); // 3 seconds
      TInterlocked.Add (value, 3000);
    end);
  tasks[0].Start;

  tasks[1] := TTask.Create (procedure ()
    begin
      sleep (5000); // 5 seconds
      TInterlocked.Add (value, 5000);
    end);
  tasks[1].Start;

  TTask.WaitForAll(tasks);
  ShowMessage ('All done: ' + value.ToString);
```

ここで強調しておきたい最後の機能は、フューチャーを定義できることです。フューチャーは、ウィキペディアによれば「変数の読み取り専用プレースホルダビュー」です。コードでは、必要時（またはバックグラウンドスレッドが使用可能になる場合は必要になる前）に、実際の値を計算するためのタスクをフューチャーに関連付けます。以下の簡単なコードサンプルでご覧いただきましょう。

```
var
  OneValue: IFuture <Integer>;
  Total: Integer;
begin
  OneValue := TTask.Future<Integer>(function: Integer
    begin
      Result := ComputeSomething;
    end);
  ...  // omitted
  Total := OneValue.Value + ...
```

このように、並列ループに基づいたマルチコアの有効活用とタスクやフューチャーを用いたバックグラウンド処理が簡単に行えるようになったことで、開発者は、新規および既存の VCL ならびに FireMonkey アプリケーションの処理を容易に高速化できます。

EMS（Enterprise Mobility Services）の概要

RAD Studio では、ネイティブライブラリや完全統合された Indy ソケットライブラリを使って、強力なインターネット接続機能が常に提供されてきました。最近のバージョンでは、次のような機能を容易に利用できるコンポーネントが追加されました。

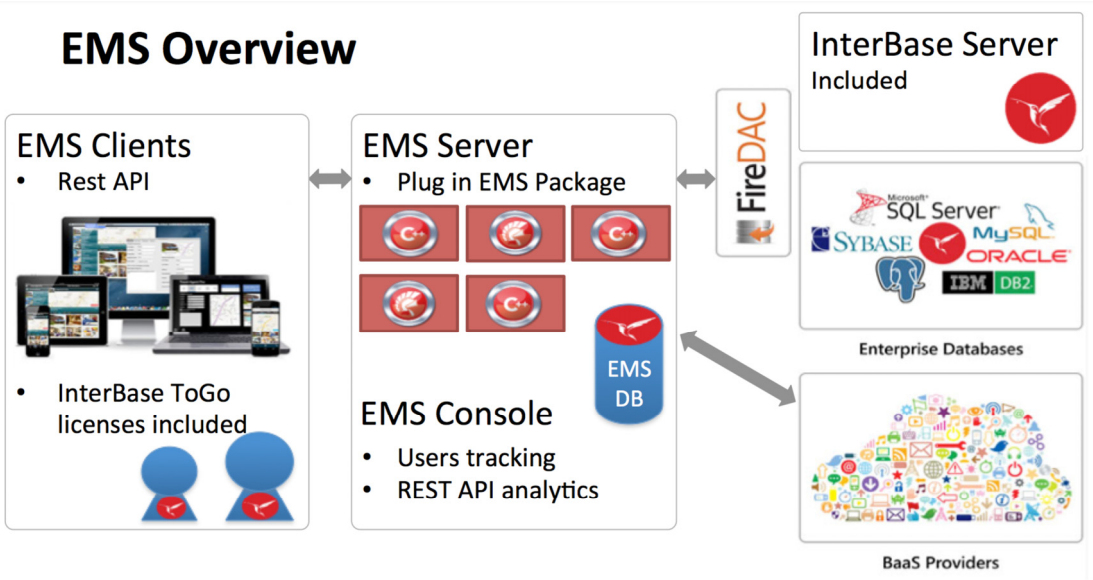
- クラウドサポート（Amazon Web Services と Microsoft Azure が対象）
- REST 呼び出しサポート（統合 URL 処理、権限付与、XE5 で導入された JSON 処理を含む）
- REST コンポーネントに基づいた BaaS（Backend as a Service）接続レイヤー（Parse や Kinvey などのサービスへの即時接続を提供、App42 サポートも別途利用可能）

多層開発に関しては、RAD Studio では、従来から DataSnap アーキテクチャを提供してきました。これは、もともと COM ベースのソリューションでしたが、その後 TCP/IP、HTTP、HTTPS、REST に移行しており、モバイルアプリケーションからのデータベースアクセスに適合しています。DataSnap は、カスタムサービスをプログラミングするための低レベル SDK と見なすことができます。この SDK は、多層開発向けのすばらしい基盤を提供するとはいえ、開発者はソリューションを何から何まで構築する必要があります。

これに対し、新たに提供される EMS（Enterprise Mobility Services）は、REST ベースの MEAP（Mobile Enterprise Application Platform）ミドルウェアスタックで、すぐに利用できるという特徴があります。サーバーはあらかじめコンパイルされており、基本的なユーザー管理サービスと認証サービスが組み込まれています。また、広範な REST API 呼び出しの解析機能により、ユーザーおよびユーザーとシステム間のやり取りを追跡するための Web ベースの REST コンソールが付属しています。

EMS のアーキテクチャ

次の図は、EMS ソリューションの概要を示しています。



バックエンドでは、既存のエンタープライズデータベース、EMS に組み込まれている InterBase サーバー、任意の BaaS プロバイダクラウドのいずれかを使用できます。中間層では、EMS は次の 3 つのモジュールで構成されています。

- EMS サーバーそのもの（開発者により作成されたカスタムパッケージを読み込んで、REST API やデータベースアクセスのホストになります）
- 別個の EMS コンソール Web サーバー（分析機能を提供します）
- InterBase を使用する EMS データベース（ユーザー構成と分析機能のログ データをホストし、EMS サーバーと EMS コンソールの双方から使用されます）

クライアント側には、任意のカスタム EMS サーバーREST API を呼び出すことができるアプリケーションがあり、暗号化されたデータベースがデバイス（またはデスクトップ）上に含まれています。

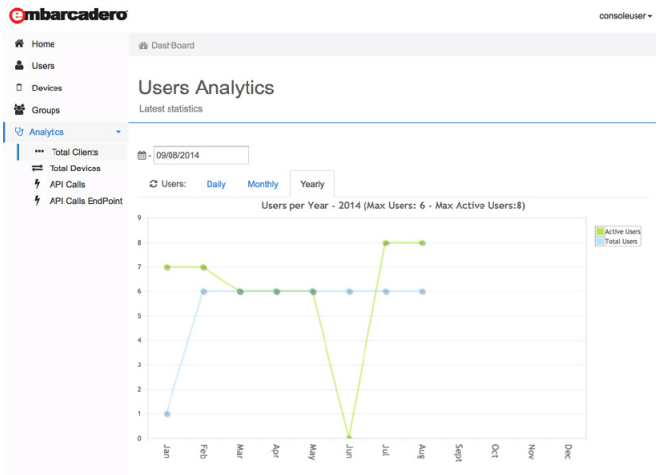
すぐに使えるインフラストラクチャ、ユーザーの API、分析機能

何から何まで開発者が作成するようなカスタムサーバーとは異なり、EMS には、ソリューションとすぐに使える状態のインフラストラクチャが用意されているため、開発と構成にかかる時間を短縮できます。

サーバーでは、ユーザー管理、接続されているデバイスの追跡、API の使用状況の監視をサポートしています。HTTP ベースの別個のコンソールを使用すれば、何が行われているのかをリアルタイムに追跡できます。

そのため、サーバー側の開発者は、Object Pascal か C++のどちらかの言語を使ってパッケージを作成し、（位置または URI が一致する）特定のリソースを登録し、対応する要求を処理するコードを書くだけです。単純なシナリオでは、カスタム Web サービス API を実装するコードを書くだけで済みます。それに加えて、社内のリレーショナルデータベース内のデータにアクセスしてモバイルユーザーに返すための使いやすいコンポーネントも用意されています。

もう 1 つの重要な機能は、ユーザー、ユーザーのデバイス、ユーザーから呼び出されている API を追跡するための広範囲にわたるリアルタイム分析機能です。API の全体的な使用状況を確認し、個々のリソースについて詳しく調べ、個々の HTTP 呼び出しさえ参照できるので、ユーザーである顧客がアプリケーションをどう使用しているか、また、どのサービスやデータソースを主に扱っているかを追跡できます。



開発ツールでのカスタマイズと使用する RDBMS の選択

EMS は、エンバカデロの RAD Studio、Delphi、C++Builder、Appmethod などのモバイルおよびデスクトップ向け開発ツールで利用できる、優れたバックエンドソリューションです。エンバカデロの開発ツールでは、特化したウィザードやすぐに使えるコンポーネントにより、モビリティサービスへのアクセスを簡単に行えるため、ソリューション全体（サーバーパッケージとクライアント呼び出し）の開発はきわめてスムーズで、ツールによく統合されており、単一の開発ツールだけでも実現可能なくらいです。

中間層とデバイスの両方でのデータベースアクセスについては、きわめて強力かつ柔軟性の高い汎用エンタープライズデータベースアクセス層である FireDAC が用意されています。EMS では、FireDAC がサポートするすべてのデータおよびデータベースを対象とすることができますが、特にプロジェクト用にデータをホストする場合を考慮して、InterBase サーバーライセンスが付属しています。これは、サービスそのもののデータの格納に使用しているのと同じデータベースです。つまり、柔軟性が非常に高く、非常に強力な RDBMS が自由に使える環境が用意されているわけですが、Oracle、MS SQL Server、

Sybase、MySQL、Informix、その他 FireDAC でサポートされている多数のデータベースの中から選択することもできます。

HTTPS と InterBase ToGo によるセキュリティの確保

大半の企業がそうであるように、セキュリティを重視する場合には、その解決策も用意されています。EMS は、Windows Server OS で動作する Microsoft IIS に ISAPI DLL としてインストールされます。IIS を HTTPS サービスとしてインストールし、クライアント上で HTTPS サポートを有効にするだけで、使用するデバイスからサーバーへの通信はすべて安全に行われます。ファイアウォールを有効にし、サービスとデータベースサーバー間の接続のセキュリティを確保したら、準備完了です。

それでも、データをデバイス上にキャッシュし格納して、接続解除時でもアプリケーションを使用可能にしたり、帯域幅が低い状況でのパフォーマンス向上などを考えている場合は、デバイス上のデータのセキュリティも確保しなければなりません。InterBase ToGo Edition は、ローカルのモバイル InterBase エンジンですが、ここで述べたセキュリティ機能も提供しています。EMS では、InterBase ToGo もソリューションの一部になっています。

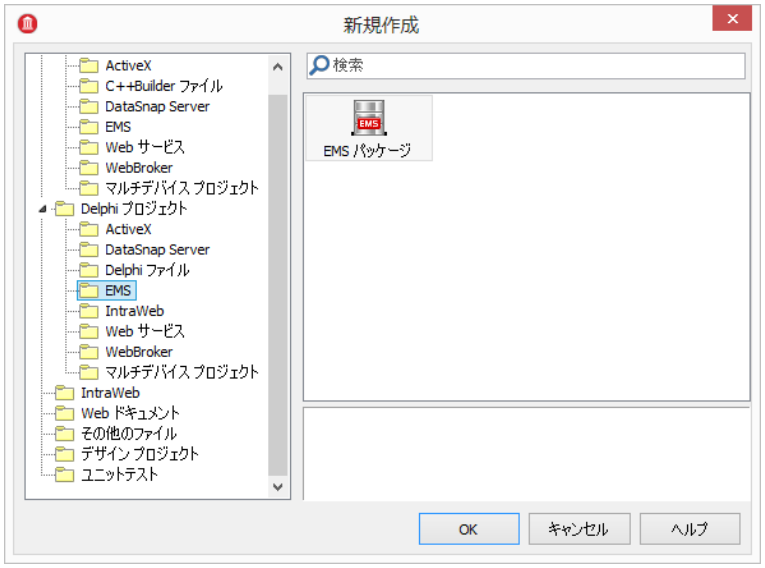
ホスティング先は自由（柔軟なライセンスプラン）

EMS は MEAP（Mobile Enterprise Application Platform）カテゴリに分類され、社内かカスタムクラウドのどちらかにホストされるようになっています。社内ホスティングの場合は、データベースに直接アクセスし、社内ファイアウォールで完全に保護されます。一方、カスタムクラウドホスティングの場合は、一般にスケーラビリティが向上し帯域幅が増えることになります。インストール可能なサーバーが提供されるため、料金の支払いが必要でかつパフォーマンスを制御できない専有のカスタムホスティングソリューションを使用しなければならない理由はありません。ホスティングを変更する必要があるときはいつでも、何の問題もなく、ソリューションの構成を自由に変更できます。

EMS は、中間層のセキュアソリューションでの API ホスティングとデータソースアクセス機能を提供します。また、パッケージの一部として、バックエンドデータを格納するための RDBMS と、モバイルデバイスにインストールするための暗号化をフルサポートしたローカル RDBMS が含まれています。柔軟な運用ライセンスオプションを用意しており、使用しているデバイスに関係なく、エンタープライズモビリティサービスにアクセスするユーザーの数に基づいた運用ライセンスを購入できます。

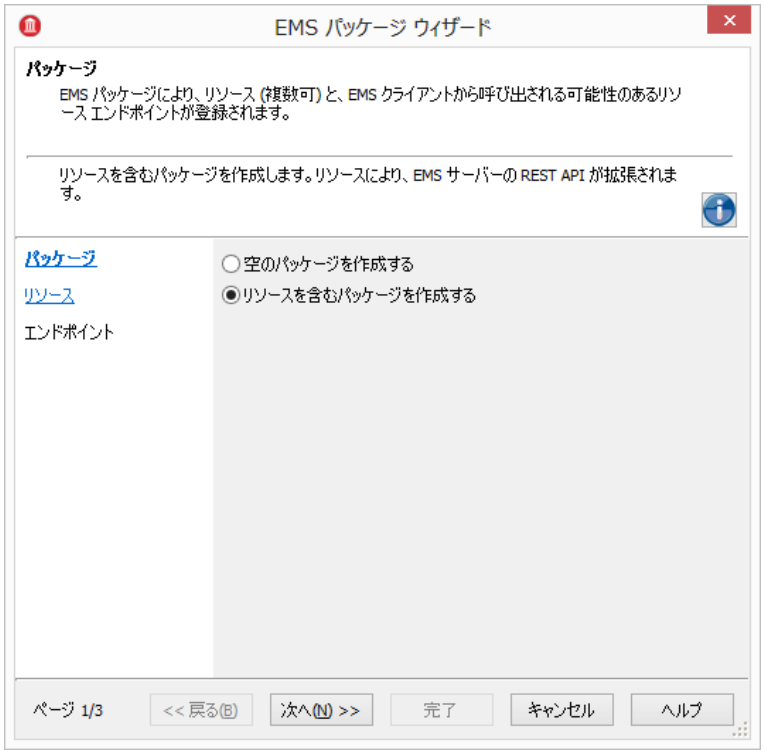
実際の EMS ソリューション開発の手順

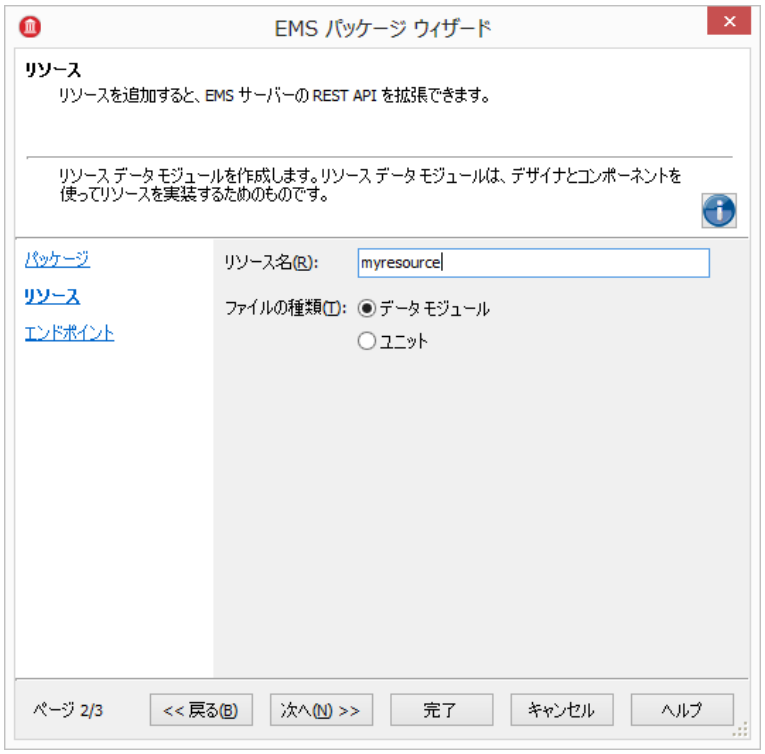
こうした概要を踏まえて、EMS ソリューションを構築する手順を見ていきましょう。サーバー側については、RAD Studio の「新規作成」ダイアログボックスから［EMS パッケージウィザード］を使用します。

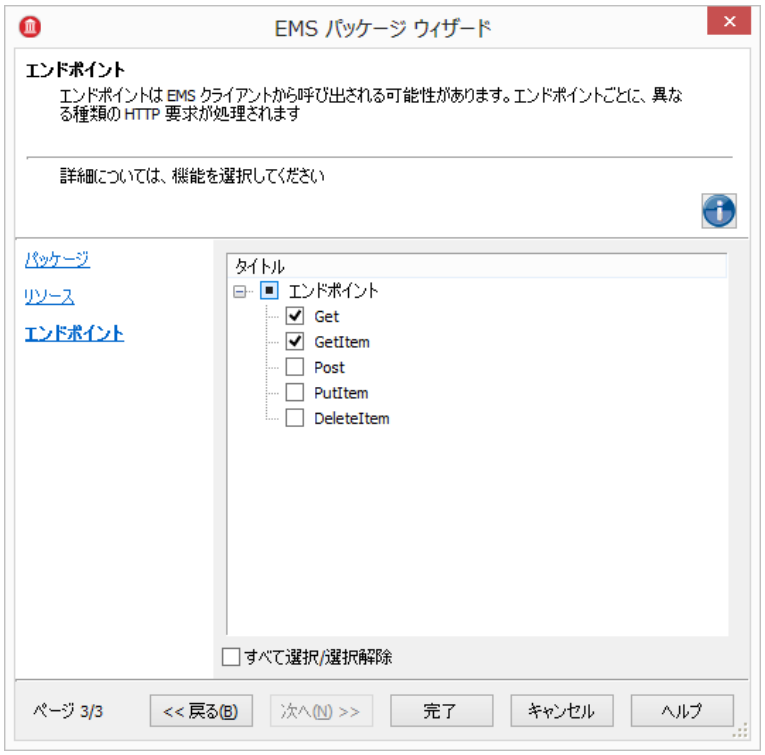


EMS サーバーパッケージの作成

ウィザードの指示に従って、以下のように EMS 拡張機能パッケージを作成します。ウィザードによって、そこに最初のリソースが自動的に追加されます。





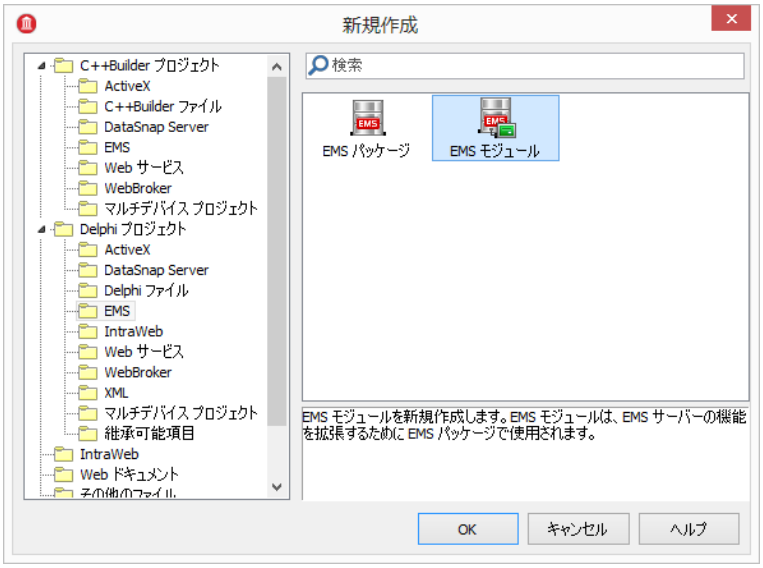


この手順では、データモジュール構造とリソース名（リソースのマッピング先の URL）を選択し、（異なる URL および HTTP アクションにマッピングされる）5 つの定義済み操作、つまりエンドポイントから 2 つを選んでいます。その結果、以下のようなデータモジュール（データアクセスコンポーネントなどの非ビジュアルコンポーネントを配置する非ビジュアルコンテナ）のコードが生成されます。

```
type
  [ResourceName('myresource')]
  TMyresourceResource1 = class(TDataModule)
  published
    procedure Get(const AContext: TEndpointContext;
      const ARequest: TEndpointRequest;
      const AResponse: TEndpointResponse);
    [ResourceSuffix('{item}')]
    procedure GetItem(const AContext: TEndpointContext;
      const ARequest: TEndpointRequest;
      const AResponse: TEndpointResponse);
  end;
```

リソース名は属性としてクラスに追加されます（これは Object Pascal の場合で、C++の場合は別のマッピングメカニズムが使用されます）。また、2 番目の Get 操作のサフィックスも同様で、これにより後続の URL セグメントが item コンテキスト情報にマッピングされます。Get 操作はリソース（またはリソースのリスト）用で、2 番目の Get 操作は単一項目用である、という考え方です。

2 つ目のモジュールを同じ EMS パッケージに追加できることに注意してください。実際、EMS パッケージを作成する際には、［新規作成］ダイアログボックスの同じページに、2 つ目の選択肢が追加されています。



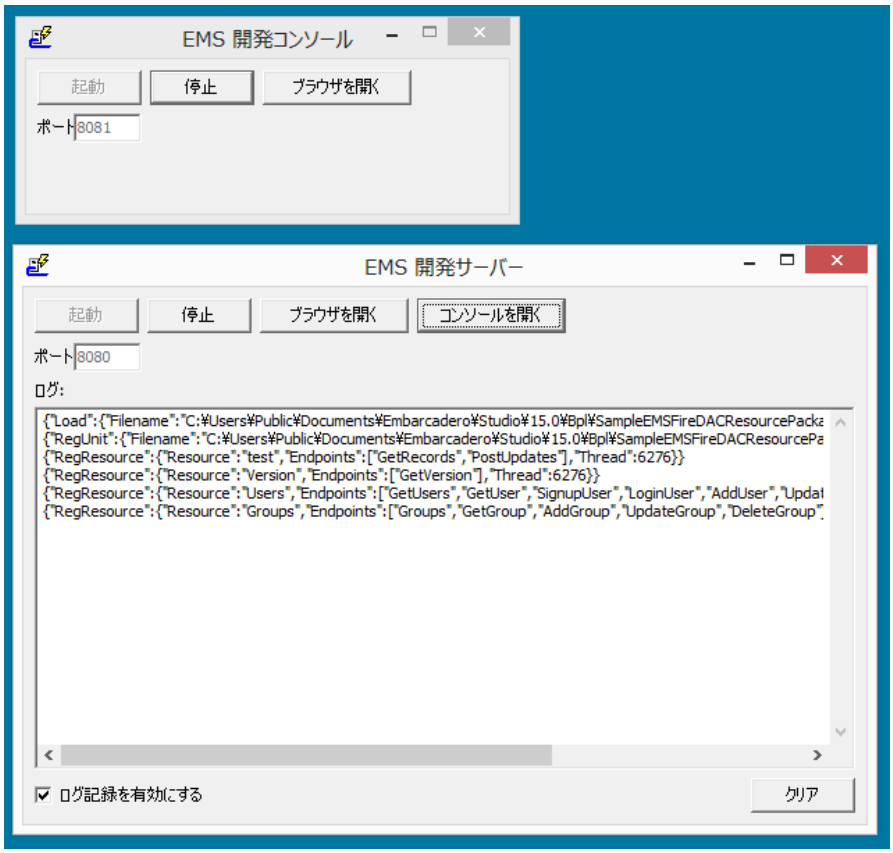
製品に付属している一連のデモの Object Pascal/Database/EMS フォルダと CPP/Database/EMS フォルダに、完全な EMS サーバー（およびクライアント）デモが 2 つあります。

サーバーのデバッグ

EMS パッケージの作成が完了したら、サーバーのスタンドアロン版（開発ツールに付属）を使って、それをテスト（およびデバッグ）できます。この EMS サーバーとそれに対応する EMS コンソールはテス

ト用のデスクトップアプリケーションですが、実際の配置は ISAPI DLL を通じて行われます（運用ライセンスを別途購入）。

これら 2 つの開発用サーバーを次の図に示します。



EMS クライアントの作成

以上でサーバーが稼働したので、それに対するクライアントアプリケーションを少なくとも 1 つ作成する必要があります。現在、IDE にはそれ用のウィザードが用意されていない代わりに、BaaS（Backend as a Service）アーキテクチャと統合された専用のコンポーネントがあります。つまり、コードの一部を Parse や Kinvey の使用から EMS の使用に（逆もまた同様）切り替えることができるということです。

通常、クライアントには、次のような EMSProvider とそれに対応する権限付与コンポーネントがあります。

```
object EMSProvider1: TEMSProvider
  ApiVersion = '1'
  URLHost = 'localhost'
  URLPort = 8080
end
object BackendAuth1: TBackendAuth
```

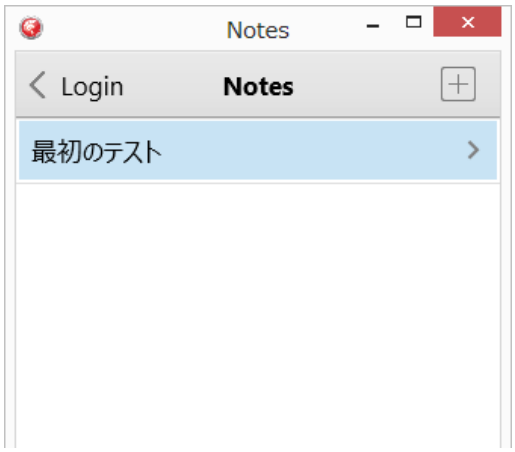
```
  Provider = EMSProvider1
  LoginPrompt = False
  UserDetails = <>
end
```

ここでは、サーバーへのアクセス全般とユーザー認証機能を構成しています。これら 2 つのコアコンポーネントには、呼び出す必要がある操作（またはカスタム REST API）ごとに BackendEndpoint コンポーネントを追加できます。各エンドポイントには、リソース参照（これもやはり実際の HTTP URL にマッピングされます）と、場合によってはさらにいくつかのパラメータがあります。2 つの例を以下に示します。最初の例には追加パラメータはなく、2 つ目の例には URL セグメントパラメータがあります。

```
object BackendEndpointGetNotes: TBackendEndpoint
  Provider = EMSProvider1
  Auth = BackendAuth1
  Params = <>
  Resource = 'Notes'
end

object BackendEndpointGetNote: TBackendEndpoint
  Provider = EMSProvider1
  Auth = BackendAuth1
  Params = <
    item
      Kind = pkURLSEGMENT
      name = 'item'
      Options = [poAutoCreated]
    end>
  Resource = 'Notes'
  ResourceSuffix = '{item}'
end
```

さらにプロパティがあれば、開発者がデフォルトの Get 以外の HTTP 操作を選ぶことができます。次の図に示すのは、Windows で動作するマルチデバイスクライアントアプリケーションです（これは、iOS、Android、Mac OS X でも動作します）。



コンソールで提供される分析機能

これでサーバーが稼働し、それを使用するクライアントも動作しているので、次の図のように、コンソールにアクセスして使用状況の統計情報を調べることができます（管理者以外の唯一のユーザーである場合は、かなり限られた情報になります）。

EMS とエンタープライズデータアクセス

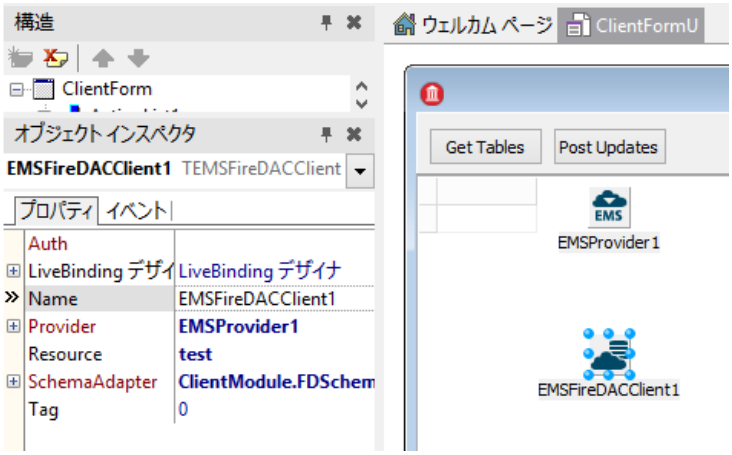
このコアアーキテクチャに加えて、EMS では、任意のエンタープライズデータベースを対象に、FireDAC データ アクセスコンポーネントにマッピングされるクライアントおよびサーバーアプリケーションの開発を具体的にサポートしています。テーブルやクエリにマッピングしデータをデスクトップクライアントやモバイルクライアントにキャッシュできるだけでなく、EMS では、更新バッチの送信やマスター／詳細構造の管理を、ほとんどコードを書くことなく実装できます。

EMS と FireDAC の統合は、主に FDSchemaAdapter コンポーネントを使って行われています。このコンポーネントにより、マスター／詳細構造を含め、標準のデータアクセスおよびデータ更新操作の広範なカスタマイズが可能になります。たとえば、次の例（SampleEMSFireDACResourcePackage デモからの抜粋）では、数行のコードを使ってマスター／詳細関係の JSON 構造を返すことができます。

```
var
  oStr: TMemoryStream;
begin
  oStr := TMemoryStream.Create;
  try
    qCustomers.Open;
    qOrders.Open;
    FDSchemaAdapter1.SaveToStream(oStr, TFDStorageFormat.sfJSON);
```

```
    // Response owns stream
    AResponse.Body.SetStream(oStr, 'application/json', True);
  except
    oStr.Free;
  end;
```

このようなデータを、クライアント側の EMS アプリケーションからはどのように取得するのでしょうか。複雑な処理が多少絡んでくることを想定し、XE7 では、コードを大幅に簡略化できる EMSFireDACClient という特定のクライアント側ヘルパーコンポーネントが導入されています。このコンポーネントとプロバイダを使った設計時の画面を、次の図に示します。



構成には、リソース名（または URL の一部）とクライアント側の FDSchemaAdapter を指定する必要があります。このコンポーネントは、（クライアント上にデータを格納するための）2 つのメモリテーブルコンポーネントと、情報を伝達するパイプの役割を果たす TableAdapters という 2 つの専用コンポーネントに接続されます。設計時の構成は単純ではありませんが、マスター／詳細構造の取得と更新の返信に必要なすべてのクライアント側コードを、次のようなたった 2 行にまとめることができるというプラス面があります。

```
EMSFireDACClient1.GetData;
EMSFireDACClient1.PostUpdates;
```

つまり、冒頭の説明やここで示した実際のコードからもわかるとおり、EMS には、開発に簡単かつすぐに使えるクロスプラットフォーム向けの中間層ソリューションが用意されています。これは、いつでもカスタム REST API をホストできるほか、モバイルプラットフォームからエンタープライズデータベースにアクセスするためのパイプの役割を果たすことができるようになっています。

Bluetooth とアップテザリングを使った ガジェットやウェアラブルとの接続

RAD Studio を使用する多くの開発者がまだ解決できていないもう 1 つの課題は、大規模な Windows VCL アプリケーションがマーケットに既にある状況で、この新しいモバイルの世界に参入する方法を見つけることでした。RAD Studio では、最近の数バージョンでマルチデバイス機能を提供してきたとはいえ、従来のアプリケーションを完全にモバイル プラットフォームに移植する具体的な計画を思い描くことは、一部のユーザーには困難でした。現実解として、移植の必要はありませんし、移植すべきでもありません。一般的な Windows VCL アプリケーションには、モバイル プラットフォーム上に移植して利用できるようにすべき機能よりも、はるかに多くの機能が含まれているからです。

アップテザリング

このような理由で、RAD Studio XE6 には、「アップテザリング」が導入されたのです。これは、既存の Windows VCL アプリケーションにモバイルデバイスを「つないで（テザリング）」、特定の機能を提供するものです。アップテザリングの重要な機能の 1 つは、ローカルエリアネットワーク上のコンパニオンアプリケーションの自動検出です。アップテザリングにより、アプリケーションではアクションやコマンドを公開（およびモバイル デバイスをリモコンとして使用）できるほか、（モバイル デバイスを画像やセンサー情報などのデータ入力ツールとして使って）データを受信できます。

この機能が使用可能になってから 6 ヶ月以上経ちますが、この間、画像やバーコードのキャプチャからデバイスや PC の制御まで、この機能を使用するさまざまなシナリオをお客様が考え出されていることがわかりました。アップテザリングは、モバイルからデスクトップへの方向に限られているわけではなく、それとは逆の方向でも機能しますし、デスクトップアプリケーション間のやり取りにも使用できることに注意してください。

RAD Studio XE7 では、次のようないくつかの新機能をアップテザリングに追加して、青写真を完成させつつあります。

- 異なるサブネットにあるデバイスやコンピュータの自動検出とそれらへの接続のサポート
- 指定された IP アドレスへの直接通信のサポート
- アップテザリングへの Bluetooth サポートの統合（これにより、Wi-Fi 経由でなく、ペアになったデバイスを使ってモバイルからデスクトップへ直接通信できます）

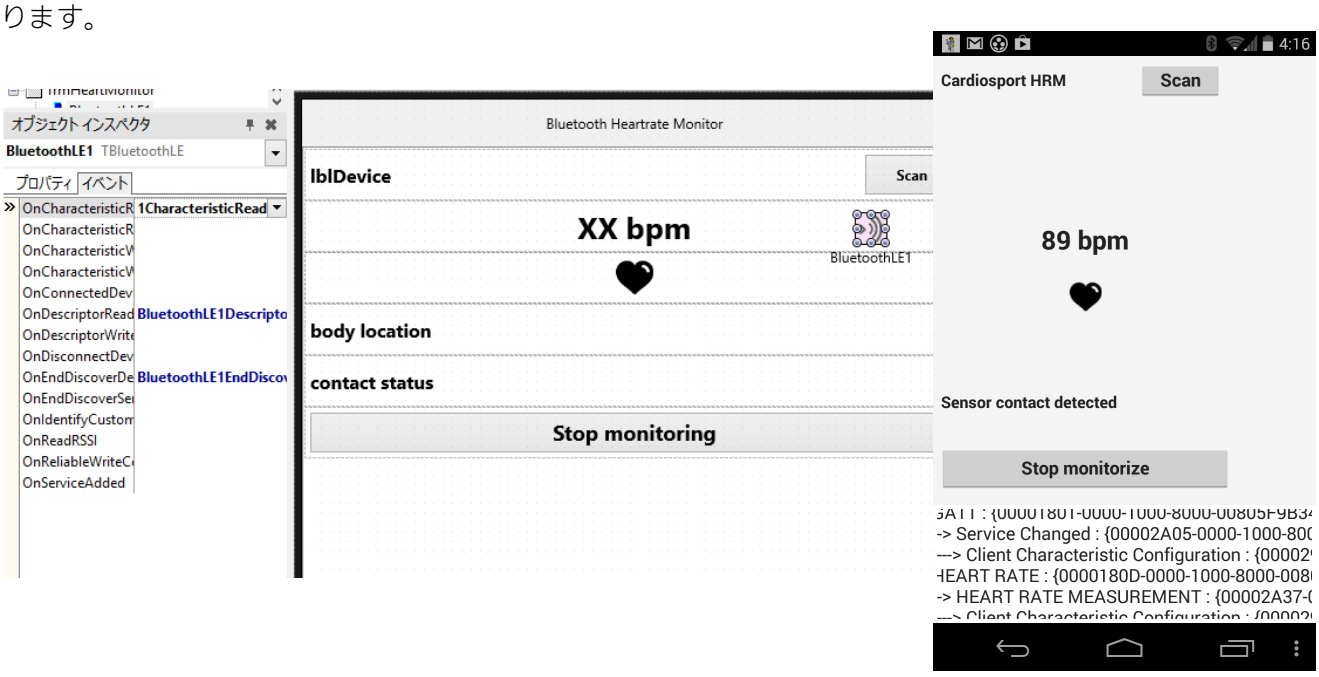
さらにすばらしいのは、コードにたった 1 つのプロパティを設定するだけで、Wi-Fi から Bluetooth に切り替えて、これらの新機能を活用できることです。

BLUETOOTH と BLUETOOTH LE

アップテザリングモデルのこのような機能拡張を実現するため、RAD Studio XE7 では、Classic Bluetooth と Bluetooth LE（Low Energy）を幅広くサポートしています。どちらのモデルの Bluetooth API も、それらをサポートしているプラットフォームから公開されています。Bluetooth LE を使用するには Mac OS X、iOS、Android の最近のバージョンが必要であるのに対して、Classic Bluetooth の使用は iOS 上では制限されています。

従来の Bluetooth はアップテザリングアーキテクチャに統合されているため、コマンドの送信やデータの共有は非常に簡単ですが、Bluetooth LE は、FireMonkey と VCL の両方で使用可能な特定の非ビジュアルコンポーネントにより、使いやすくなっています。LE デバイスでは、一般にデータ接続機能が制限されており、固有のプロトコルが用意されています。

Bluetooth LE サポートの例として、非常に簡単な心拍数監視モバイルアプリケーションを作成しました。これは、世の中に多く出回っているいくつかのウェアラブル健康管理機器に接続できます。次の図は、BluetoothLE コンポーネントを使ったデモアプリケーションの設計時と Android で実行したときの様子です。このアプリケーションには、ガジェット検出、管理、データ処理、表示などのコードが 50 行ほどあります。



開発者の生産性向上に関する新機能

ここまで紹介してきたものは、RAD Studio を初めて使用する開発者にとっても、既に使用している開発者にとっても重要な新機能ですが、その他にも、ターゲットプラットフォームやライブラリが何であれ、開発者が日常業務で活用できるいくつもの重要な機能強化が施されています。XE7 では、製品のほとんどの分野で改良が施されているほか、品質とパフォーマンスも向上しています。

IDE の新機能

これらの機能強化のうち、多くは RAD Studio の統合開発環境（IDE）に関するものです。IDE は Windows にホストされますが、直接またはプラットフォームアシスタントサーバー（PAServer）として知られる「ツールチェイン・リモーティングアーキテクチャ」を経由して、Microsoft、Apple、Google のプラットフォーム SDK ツールとシームレスに統合することができます。PAServer を Mac で動作させることにより、開発者は、Apple SDK の操作（コード署名からデバイス配置やデバッグまで）を、Windows で動作する IDE から直接呼び出すことができます。

XE7 の新しい PAServer マネージャは Mac OS X のトレイアプリケーションであり、これを使って PAServer（iOS および Mac OS X アプリケーションを開発するためのプラットフォームアシスタントアプリケーション）の複数のインスタンスを起動し管理できます。

IDE の機能強化のもう 1 つのグループは、バージョン管理の統合に関するものです。XE7 では Subversion サポートを大幅に改良したほか、ローカルリポジトリとの統合に限定されるものの、Git バージョン管理システムが初めて IDE に統合されました。

OBJECT PASCAL 言語の新機能

Object Pascal 言語は、Java や C# と似た機能を備えた強力なプログラミング言語で、非常に堅牢かつ使いやすいと考えられており、Windows での開発では今でも非常に人気があります。最近の数年間で、この言語には、ジェネリックス、無名メソッド、リフレクション、クラスおよび組み込み型ヘルパなど、多数の機能が追加されました。

XE7 では、Object Pascal 言語に、配列の管理に関する重要な改良がいくつか行われています。定数配列と動的配列をうまく組み合わせて、一方をもう一方に代入できるようになりました。また、動的配列を文字列のように連結したり、任意の型の動的配列に対して Insert や Delete のような共通の RTL 関数を使用したりできるようになりました。

これらの新しい言語機能を強調した短いコードサンプルを次に示します。

```
var
  di: array of Integer;
  i: Integer;
begin
  di := [1, 2, 3];    // initialization
  di := di + di;      // concatenation
  di := di + [4, 5];  // mixed concatenation

  Insert ([8, 9], di, 4);
  Delete (di, 2, 1); // remove the third (0-based)
```

```
for i in di do
  Memo1.Lines.Add (i.ToString);
```

RTL の機能強化

XE7 のランタイムライブラリレベルでは、Bluetooth サポートと新しい並列プログラミングライブラリのほかには、TXMLDocument に新しい XML 処理エンジン OmniXML が統合されており、モバイルプラットフォームでのパフォーマンス向上が図られています。一方、Windows では MS XML エンジンの使用が引き続きデフォルトになっています。

データベースアクセスと FIREDAC の機能強化

FireDAC は、クライアント／サーバーアーキテクチャにおける多層サーバーでも、デスクトッププラットフォームやモバイルプラットフォーム上のローカルデータベースへのアクセスにも使用できる汎用データアクセス技術です。XE7 では、FireDAC は開発環境にさらに統合され、ユーザーエクスペリエンスの不可欠な部分となっています。

FireDAC データベースコンポーネントおよびドライバは RAD Studio、Delphi、C++Builder に完全に統合されています。FireDAC を用いれば、Delphi および C++Builder から InterBase、SQLite、MySQL、SQL Server、Oracle、PostgreSQL、DB2、SQL Anywhere、Advantage DB、Firebird、Access、Informix などの、多様なローカル／組み込みデータベース、モバイルデータベース、エンタープライズデータベースに高速かつネイティブに直接アクセスできます。

提供される機能は、次のように、2 つのカテゴリに分かれます。

- FireDAC のローカル／組み込み接続機能により、主要なローカルデータベースとの接続が可能（RAD Studio、Delphi、C++Builder Professional で利用可能）
- FireDAC のローカル／組み込み接続およびリモート エンタープライズ接続機能により、あらゆるデータベースとの接続が可能（RAD Studio、Delphi、C++Builder Enterprise、Ultimate、Architect の各エディションで利用可能。Professional 版の場合、FireDAC Client/Server Add-On Pack を別途購入して利用可能）

最後に、BDE などの、現在では使用されていないデータベースアクセス技術から非常に簡単に移行できることは、FireDAC の特筆すべき機能です。これは、モデルが似ていることと、製品に移行スクリプトが付属しているためです。

XE7 でも、データベースと FireDAC に関する機能強化が数多く行われており、詳細に解説する価値があるでしょう。ここでは、以下に要約しておくこととします。

- 組み込み可能な無料のデータベースである IBLite（InterBase の機能限定版）が Windows、Mac、Android、iOS で使用できるようになりました（従来は、モバイルプラットフォームのみ無料で提供）。有料版と比較して、データベースサイズの制限と暗号化を行えないという機能制限があります。データベース ファイルについては、InterBase ToGo およびフル機能を提供する InterBase Server とバイナリ互換です。

  InterBase の詳細については、http://www.embarcadero.com/jp/products/interbase を参照してください。
- FireDAC では、BLOB フィールドのストリーミングと Microsoft SQL Server ファイルストリームをサポートするようになりました。（一部のデータベースエンジンで使用可能な）真のストリーミングのサポートにより、サーバーからのデータの送信中にクライアントアプリケーションでそのデータを使用し始めることができます。
- デスクトップおよびモバイル専用の新しい IBLite ドライバが FireDAC に追加され、データベースの無料の組み込み版がさらに使いやすくなりました。
- 開発者向けに、FireDAC の接続パラメータが「オブジェクトインスペクタ」にレコードとして表示されるようになり、その設定を設計時にカスタマイズしやすくなりました。
- 「データエクスプローラ」の機能強化がいくつか行われています。たとえば、主キーおよびフィールドのほか、外部キーおよびフィールド、シーケンスやジェネレータも表示されるようになりました。

FireDAC の詳細については、http://www.embarcadero.com/jp/products/rad-studio/firedac を参照してください。

VCL の機能強化

VCL に特化した新機能は多くはありませんが、Bluetooth サポートや並列プログラミングライブラリなどのプログラミング言語や RTL における機能強化はすべて、Windows プラットフォームを使っている VCL 開発者も利用できます。

XE6 での機能強化を踏襲し、Windows タスクバーボタンを管理するための Taskbar コンポーネントを使用できるほかに、XE7 では、これと同じタスクバーボタンにカスタムメニュー項目を追加するための JumpList コンポーネントも追加されています。これらは、開発者が既存の Windows アプリケーションをモダンにするうえで役に立つ多くの機能（64 ビットコンパイラから VCL スタイル設定のサポートや Modern UI に至るまでのさまざまな機能）のうちの一部にすぎません。

RAD STUDIO XE7 追加情報

RAD Studio XE7 は、Delphi XE7、C++Builder XE7、HTML5 Builder、InterBase を含む開発ツール スイート製品です。

- RAD Studio XE7 製品カタログ

  http://www.embarcadero.com/images/jp/dm/datasheet/radstudio-xe7-datasheet-jp.pdf
- Delphi XE7 製品カタログ

  http://www.embarcadero.com/images/jp/dm/datasheet/delphi-xe7-datasheet-jp.pdf
- C++Builder XE7 製品カタログ

  http://www.embarcadero.com/images/jp/dm/datasheet/cbuilder-xe7-datasheet-jp.pdf

RAD Studio XE7 の画面ショット集は http://www.embarcadero.com/jp/products/rad-studio/screen-shots で入手できます。

XE7 製品ラインナップ

RAD Studio XE7 には、Professional、Enterprise、Ultimate、Architect の 4 つのエディションがあります。

機能	Architect	Ultimate	Enterprise	Professional
マルチデバイス開発	Android、iOS、Windows、Mac のネイティブ開発（Delphi および C++Builder） Web アプリケーションとモバイル Web アプリ開発（HTML5 Builder）			
DB アプリケーションアーキテクチャ	クライアント/サーバー、多層、ローカル/組み込み			ローカル/組み込み
データベースおよびクラウドサポート	InterBase、SQL Server、Oracle、DB2、Sybase、MySQL、ODBC など、Amazon / Azure クラウド、DataSnap 多層アクセス、FireDAC データアクセスコンポーネント			InterBase、MySQL、SQLite、Amazon / Azure クラウド
IDE ツール	リファクタリング、ユニットテスト、UML モデリングのフル機能を備えた先進的な IDE			リファクタリング、ユニットテスト、UML による可視化機能を備えた先進的な IDE
高度なデータベースモデリングツールおよび SQL ツール	ER/Studio Developer Edition	DB PowerStudio Developer Edition	含まれていない	
旧バージョンへのアクセス	旧バージョン（Delphi/C++Builder 2007 ～ XE6、Delphi 7、C++Builder 6、RadPHP XE2、RadPHP XE）のライセンス取得とダウンロードが可能。			
含まれている IDE パーソナリティ	Delphi、C++Builder、HTML5 Builder			

RAD Studio XE7 Professional

Embarcadero® RAD Studio XE7 Professional は、Windows、Mac、iOS、Android 向けの真のネイティブアプリケーションを単一のコードベースで構築することができるビジュアル開発ツールスイートです。ローカル データベース接続機能を備えたハイパフォーマンスかつマルチデバイス対応のコンパイル済みネイティブアプリケーションを開発でき、アプリケーションユーザーにハイレベルなユーザーエクスペリエンスを提供できます。

FireDAC Client/Server Add-On Pack for RAD Studio XE7 Professional

FireDAC Client/Server Add-On Pack を利用すれば、RAD Studio XE7 Professional でクライアント／サーバーデータベースへの接続が可能になり、その他のエンタープライズデータベースもサポートできるようになります。強力な汎用アーキテクチャであるため、FireDAC を使用すると、アプリケーションから Oracle、SQL Server、InterBase、DB2、Firebird、SQLite、MySQL、PostgreSQL、SQL Anywhere、Advantage DB、Access、Informix などに直接アクセスすることができます。

RAD Studio XE7 Enterprise

Embarcadero® RAD Studio XE7 Enterprise は、クライアント／サーバーデータアクセスや多層開発機能を搭載したエディションです。ISV や企業が、Windows、Mac、iOS、Android 向けのマルチデバイスアプリケーションを作成するために必要な機能が多数搭載されています。RAD Studio Enterprise には、Professional エディションのすべての機能に加えて、エンタープライズデータ接続、モバイルアプリケーション開発、DataSnap や EMS による多層アプリケーション開発の機能が含まれています。

RAD Studio XE7 Ultimate

Embarcadero® RAD Studio XE7 Ultimate は、エンタープライズデータベースシステムを扱うアプリケーション開発プロジェクトで役に立つ機能を搭載した上位エディションです。RAD Studio Ultimate には、Enterprise エディションのすべての機能に加えて、DB PowerStudio® Developer エディションの SQL プロファイリングツールと SQL チューニングツールが含まれています。

RAD Studio XE7 Architect

Embarcadero® XE7 Architect は、エンタープライズデータベースシステムの設計や利用で役に立つ機能を搭載した上位エディションです RAD Studio Architect には、Enterprise エディションのすべての機能に加えて、ER/Studio® Developer Edition の強力なデータベースモデリングおよび設計機能が含まれています。

エンバカデロ・テクノロジーズは、1993 年にデータベースツールベンダーとして設立され、2008 年にボーランドの開発ツール部門「CodeGear」との合併によって、アプリケーション開発者とデータベース技術者が多様な環境でソフトウェアアプリケーションを設計、構築、実行するためのツールを提供する最大規模の独立系ツールベンダーとなりました。米国企業の総収入ランキング「フォーチュン 100」のうち 90 以上の企業と、世界で 300 万以上のコミュニティが、エンバカデロの Delphi®、C++Builder®といった CodeGear™製品や ER/Studio®、DBArtisan®、RapidSQL®をはじめとする DatabaseGear™製品を採用し、生産性の向上と革新的なソフトウェア開発を実現しています。エンバカデロ・テクノロジーズは、サンフランシスコに本社を置き、世界各国に支社を展開しています。詳細は、www.embarcadero.com/jp をご覧ください。