データベースアプリケーション構築技法

Delphi、C++Builderによるメンテナンス性を考慮した開発

第三章：旧バージョンから BDS 2006 への移行

Borland®

移行に関する注意事項

- 移行の必要性
- BDS2006 での 旧Delphi / C++Builder から変更点
  - BDE の現状
- データアクセスフレームワークを変更するか否か？
  - データアクセスロジックをどのように扱うか
  - データアクセスロジックをどのように変更するのか？
- メンテナンス性の高いシステム、低いシステム
- 移行に関する一般的な手順

移行の必要性

- 例）Oracle 8i をターゲットに Delphi 6 で作成されていたシステム
  - Delphi 6 + BDE + SQl-Link for ORACLE + Oracle 8i
- ハードウェア、ソフトウェアのライフサイクルやサポート期間の終了による変更などがあった場合、どのように対応するか
  - 外的要因により、現状のシステム（アプリケーション）の変更を行なわなければならない
    - 稼動するプラットフォーム（Windows）のバージョン、サポート期間
    - ターゲットとなる DBMS のバージョン、サポート期間
    - ハードウェアの変更
  - これらの要因に対応するには、どのようにすれば良いのか？
    - 新しい、BDS を利用してシステムを構築し直す必要がある？
- システム再構築のために開発環境を最新の物に移行する場合
  - 開発環境の違いを知る必要がある

Borland®

移行に関する注意点

- Delphi / C++Builder /および VCL ライブラリは「可能な限り」高い下位互換性を保持するようにバージョンアップを繰り返している
  - ただし完全では無く、例えば VCL ライブラリではバージョン間で互換が失われる個所がある
    - 新しい機能の追加などに対応する為
    - ライブラリ構成の見直しの為の修正
    - 3rd Party 製コンポーネントの提供終了
    - 推奨されないコンポーネントのパッケージからの削除
  - 以前のプロジェクトを全く作業無しに完全な形で移行は難しい場合がある
    - 特に開発環境のバージョンが開いた場合などに顕著となる
      - Delphi 5 で作成されたソースを BDS 2006 へ移行する場合など

基本的に多くのコードは流用可能だが、移行先のバージョンが移行元のバージョンから、何がどのくらい変更されているかを把握する事は重要となる

Borland®

BDS2006 での 旧Delphi / C++Builder から変更点①

- BDE の取り扱いの変更があった（後述）
  - BDE の DBMS サポートの正式終了
- Delphi が採用しているクラスライブラリ VCL の互換性
  - VCL のバージョン間の互換性は極力、それを維持している
    - やむを得ない事情などによってはその限りではない
    - 例）データベース関連のクラス定義などを、dbExpress 導入前後で見直し
      - BDE / DBExpress など複数のデータアクセス技術に対応するため、TDataSet クラスの位置付けや定義が変更されている
      - BDE 用の抽象クラス ->汎用データセット抽象クラス
  - 変更によってインターフェイス間の互換の問題が出た場合
    - 新しいクラスの仕様に合わせコードを適切に変更する
- 3rd Party 製品での互換性の問題
  - 新しいバージョンに対応していない可能性
  - VCL の互換性と同等の問題
    - 例としてOffice 製品ではバージョンが変更で、以前のバージョンではプロパティであったものが、以後のバージョンでは同じ名前のプロシージャに変わっていたなどの実例がある

Borland®

BDE の現状

BDEの開発およびメンテナンスは、Delphi 7 付属の BDE 5.2 にて終了。最新の、BDS 2006 において、BDEはローカルデータベースファイルへアクセスするエンジンとして残されているが、SQL-Link は廃止されており、付属しておらず BDE からの DBMS への接続はサポートしていません。

- Oracle 8i / InterBase 6 等、旧バージョンの DBMS での対応で終了。今後の提供予定なし
- BDS 2006 に付属の BDE でも ODBC 経由での DBMS への接続は技術的に可能だが、その動作はサポート対象外
- 関連資料:
  - The Future of the BDE and SQL Links
    http://bdn.borland.com/article/0,1410,28688,00.html

Borland®

BDS2006 での 旧Delphi / C++Builder から変更点②

- VCLのライブラリ構成の変更
  - ユニット名の変更
  - クラスの定義が別ユニットへ移動
- この問題に遭遇した場合の対処法

症状	「未定義の識別子」コンパイルエラー
理由	クラスの定義がない
対処法	マイグレーション作業で新しいユニットをUSES節に反映する

症状	「型の不一致」コンパイルエラー
理由	偶然にも同名のクラスやレコード型などが複数のユニットに宣言されていると、従来適用されていた型がスコープから外れ、他方の型が適用されてしまう(この場合、識別子は見つかっているため「未定義」とは報告されない)
対処法	以前のバージョンのフレームワークを解析し、適切な名前空間が反映されるように、コードを修正する

Borland®

BDS2006 での 旧Delphi / C++Builder から変更点③

- クラスライブラリのフォント / 文字コード関連の変更
  - VCLアプリケーションのデフォルト・フォントの変更
    - Delphi 2005以降の Font プロパティを持つVCLクラスにおける変更
      - デフォルトのフォントが MS Pゴシック 9ptから Tahoma 8ptに変更
      - デフォルトの Charsetが SHIFTJIS_CHARSETから DEFAULT_CHARSETに変更
      - 旧プロジェクトに新規フォームを追加する場合など注意が必要
- この問題に遭遇した場合の対処法

症状	設定を変更せずにWindows XPでコンパイルすると、Windows 98環境で実行したときに文字化け
理由	Font / Charset のデフォルト値が変更され、初期値のままではWindows9x では表示できなくなった為
対処法	CharsetをSHIFTJIS_CHARSETに変更し、再コンパイルする

Borland®

BDS2006 での 旧Delphi / C++Builder から変更点④

- ヘルプシステムの取り扱い方式の変更
  - デフォルトのヘルプファイルの取り扱いがWinHELP からHTMLHelp に変更（Delphi 2005以降）
  - Delphi 2005以降でWinHELPを利用するには
    - WinHelpViewer ユニットを USES 節に指定する
- レポートツール「QuickReport」の使用
  - C++Builder6 / Delphi7まで提供されていた QuSoft 社の QuickReport は、このバージョンでは提供を受けていない
  - これを利用した実装がある場合は、次の2つから選択する
    - 別の機能に置き換える
      - 例：RaveReports 等
    - 別途QuSoft 社から QuickReport を購入する
      - http://www.qusoft.com/

※ 国内での購入方法については、弊社問い合わせ窓口まで、お問い合わせください。

Borland®

BDS2006 での 旧Delphi / C++Builder から変更点⑤

- STL の変更(C++Builder)
  - 付属の STL が STLport から Dinkumware へ変更されています
- MakeFile 作成ツールが付属していない (C++Builder)
  - 別途、Borlandのダウンロードサイトより入手可能
    - BDSProj2Mak
      - http://www.borland.com/downloads/registered/download_bds.html
- NetMaster 系の ネットワーク関連コンポーネントの廃止
  - オープンソースで提供される Indy ライブラリのみが付属
    - http://www.indyproject.org/

Borland®

古いコードを変更する際のテクニック

- 古いバージョンの Delphi でコンパイルできる可能性を残す
  - IFDEF の利用
    - バージョンの互換性を単一コードで確保できない場合などに利用する
    - コンパイラの条件定義、IFDEF を利用しコードを分割する
  - 条件指令の指定方法
    - {$DEFINE …} を利用する
    - プロジェクトオプションの[ディレクトリ/条件]の条件指令に指定する
    - 埋め込みバージョン定義を利用する
      - Delphi 7
        - VER150
      - BDS 2006 の Delphi Win32
        - VER180

ただし埋め込みバージョン定義を用いる場合、コンパイラのマイナーバージョンアップに伴い、定義も変更される恐れがあるので注意！

```
{$IFDEF VER150}
    //Delphi 7 固有のコード
{$ELSEIF VER180}
    //Delphi 2006 固有のコード
{$ELSE}
    //Delphi 7 でも Delphi 2006 でも無い場合
{$ENDIF}
```

Borland®

データアクセスフレームワークを変更するか否か？

- 旧Delphi / C++Builder + BDE + DBMSで構成されていたシステム
  - BDS では BDE + SQL-Link によるDBMSへのアクセスをサポートしない
  - BDE + SQL –Link から他の接続形態を利用するように変更しなければならない
- データアクセスロジックをどのように扱うか

現行のシステムに手を入れたくない

- BDE + ODBC による接続設定変更のみでの対応
- 正規にサポートされない接続形式
  - 開発者の責任において行なう
  - 変更作業の大半は「テスト・テスト・テスト」に費やす
    - テストケースの作成

現行のシステムを修正する

- ADO Express / DBExpress 等を利用
  - 正規にサポートされる範囲での組み合わせで利用可能
  - システムに対し、ある程度の規模の修正が必要

Borland®

現行のシステムを修正する場合

- データアクセス部分のコードを変更する
  - データアクセスコンポーネントを違う物に置き換える
    - BDE -> ADO Express / DBExpress / IBExpress or 3rd Party 製品等
  - 置き換えたコンポーネント用にコードを書き換える
    - 各接続形式に対応したデータアクセスコンポーネントは、細かい部分での動作に違いがある
      - 接続設定の扱い
      - トランザクションの扱い
      - データの取得された結果やその扱い
      - データベース、ミドルウェアエラーのハンドリング方法や、その扱い
    - 必要に応じ、コードを書き換える必要がある
      - メンテナンス性の高いシステム、低いシステム、変更要件、現行システムの仕様、様々なファクターで必要な工数は変わる
        - 特にメンテナンス性の低いシステムでは工数は飛躍的に増える傾向にある

Borland®

メンテナンス性の高いシステム、低いシステム

- 適切なクラス設計をしているメンテナンス性の高いシステムの移行は比較的容易になる傾向がある
  - データ操作を正しく抽象化、カプセル化しているか否か？

悪い例
- TDabase.Open(); をフォームのイベントで呼んでいる
- 検索をフォーム上の TTable クラスの Filter プロパティで処理している

良い例
- 自分で作成した DataModule に DBConnect 等の関数を作成し、TDabase.Open(); を その中から呼んでいる
- 自分で作成した DataModule に検索用の関数を用意してある

- メンテナンス性の高い/低いで移行の作業内容は大きく変わ

メンテナンス性の高いシステム
- 変更箇所の把握が行ないやすい
- 変更に対するほかの個所への影響が把握しやすい
- 関連する場所への影響を最小限に留められる

メンテナンス性の低いシステム
- 画面上のイベントなどに散乱したデータ操作コードを探し出し一つづつ直す
- 直し忘れや不適切な修正方法、修正によるほかのコードとの競合など問題がある

Borland®

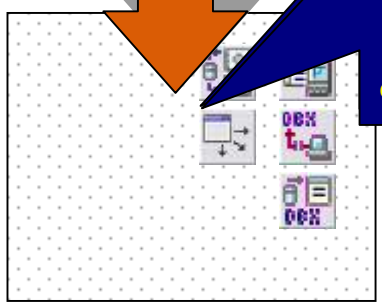
メンテナンス性の高いシステム、低いシステム

- 例えば、データベース、ミドルウェアエラーのハンドリングの変更方法を考えた場合
  - BDE でのデータベースエラーは「EDBEngine クラス」の例外となる
    - 他のコンポーネントに置き換えた場合、別のクラスの例外となる
    - 例外ハンドリングで「try … Except on EDBEngine do…」を利用できない
    - 例外のエラーコードやメッセージも変わる
      - 例えば、「try … Except on E:EDBEngine do begin」等で e.message と取得するメッセージは同じではない
      - どうすれば良いのか？
        - 単に例外クラスのみ置き換えるだけでは終わらない
        - どのようなエラーが発生した場合にどのようにするかを確認し、適切に直す必要がある

データベース、ミドルウェアエラーのハンドリングの変更

- このような問題を回避するために、データ操作ロジックと同じように例外を抽象化することが有効
  - VCL フレームワークで提供されているエラーをアプリケーション固有の例外クラスを作成し抽象化しておく
  - データアクセス操作と同様、その結果としての例外もデータモジュール側で処理する

例えば単純なデータベース接続に関する実装の場合



```
TMyErrorClass = class(Exception);
.
.
.
procedure TMyDataModule.Connect;
begin
  DataBase.Params := getConnectionName();
  Try
 DataBase.Open();
  except on EDBEngineError do
    raise TMyErrorClass.Create('正常に接続できませんでした。');
  end;
end;
```

Form 側は Connect 関数を呼び出し、失敗は TMyErrorClass でハンドリングしているので変更の影響が無い

Borland®

移行に関する一般的な手順①

- 何を移行するのかを明確化
  - 漏れがないように、また必要のないものまで移行しないように
- 移行の影響調査
  - テストアプリケーションの作成
  - バックアップ後、テスト環境で事前コンバートのテストを行なう

- 移行に必要なソースコードを準備する
  - パッケージ・ソースコード、外部コンポーネントのソース
  - プロジェクトソースコード（旧バージョンでビルド可能な物）

- 移行前に、旧バージョンのDelphi / C++Builder で作業
  - 既存の問題点の解消
  - 移行に際し、問題となりそうな個所の切り出しと修正
  - 少しでもメンテナンス性の高いシステムとして構成を変更するのも有効

開発環境移行前に、既存プロジェクトの問題を
解決しておけば、移行後に発生した問題が、
従来からの問題か、移行による問題なのか
切り分けが容易になる

Borland®

移行に関する一般的な手順②

- パッケージ / コンポーネントのインストール
  - 必要な場合、修正を行いインストールする
- プロジェクトファイルのコンバート
  - 旧バージョンのDelphi / C++Builder のプロジェクトを BDS 2006 で開く
    - Win32 プロジェクトへコンバートする
  - 必要な場合、修正を行ない、ビルドする
- 修正作業
  - データベースアプリケーションの場合、コンポーネントや関連コードの変更など
- テストを繰り返す
  - 必要な場合、ユニットテストなどを用いる

Borland®

この章のまとめ

- 旧 Delphi / C++Builder から BDS 2006 への移行はプランニングが重要
  - 特にデータベースアプリケーションでは、そのフレームワークの違いによる移行方法の検討が非常に重要となる
    - BDE を利用しているシステム
      - BDE はサポートが終了しており 最新の DBMS をサポートしていない
  - もちろん、多くのコンポーネントの機能に対して互換性は確保してある
  - プロトタイプによるテスト等も有効な手段となる
- どのような開発にも当てはまるが、メンテナンス性の高いシステムほど移行は楽になる

移行に関する一般的な内容は解説したが、やはり移行はアプリケーション毎に千差万別であり、充分なプランニングの上で行なって欲しい

Borland®