# Delphi から利用する Firebird

Firebird 日本ユーザー会　林　務

2004/6/19
@ タイムインターメディア社会議室

# 本セミナーの目的

- Firebird の利点を最も有効に利用できる開発環境は Delphi です。
- 理由その 1
  ミドルウェアの充実
- 理由その 2
  データベース対応機能の充実
- 理由その 3
  国内・海外での豊富な実績の蓄積

# ミドルウェア

- BDE
  SQL-Link 、 ODBC
- dbExpress
  InterBase ドライバ、 interXpress
- IBX(InterBase Express)
- IBObjects
- FIBPlus
- ZeosDBO
- その他・・・、 ODBCCall 、 Mercury 等

# dbExpress

- ボーランドが今後の標準とするミドルウェア
- InterBase/Firebird の他、 MySQL にも対応
  サードパーティ製では PostgreSQL も
- 単方向のみのため、使い方に注意が必要

# InterXpress for Firebird

http://www.upscene.com/

- UPSCENE 社が開発している、 dbExpress 用のドライバ
- Desktop 版と Server 版、 Server 版 ( コネクションプーリング有 )
  ※ ConnectionPooling 版は取りやめ？

# IBX (InterBase Express)

- FreeIBComponents 派生のネイティブコンポーネント
- ボーランドが InterBase と共に配布してきたもの
- InterBase と Firebird の交換性が低くなると使えなくなってくる
- Backup/Restore 等にも対応
- 実質上、 Firebird 1.0.3 まで

# IBObjects

http://www.ibobjects.com/

- Jason Wharton 氏が開発しているミドルウェア
- もともと TDataSet 派生ではないため、速度的に問題がある？
- デザインが派手なのも好き嫌いの分かれるところ
- 多機能だが、管理系は含まず
  （ Lorenzo 氏作の IBOAdmin が存在する ）

# FIBPlus(FastInterbasePlus)

http://www.devrace.com/en/fibplus/

- DevRace 社が開発しているミドルウェア
- IBX と同様に、 freeIBComponents 派生
- 配列型のフィールドも扱える
- かっちりとした印象
- Backup/Restore 等にも対応

# Zeos Library

http://www.zeoslib.net/

- オープンソースのミドルウェア
- Firebird/InterBase のほか、 PostgreSQL・ MySQL・ Oracle・ Sybase・ MS SQL と各種 RDBMS に接続可能
- 基本機能はしっかりしているが、必要最低限なものに限られる

# Firebird ネイティブ対応

- Firebird にネイティブ対応しているミドルウェアは以下の 3 つ

| | IBX | IBObjects | FIBPlus |
|---|---|---|---|
| 安定感 | ○ | △ | ○ |
| 速度 | ◎ | △ | ◎ |
| 機能 | ○ | ◎ | ○ |
| 使い勝手 | ○ | ○ | ○ |
| ジェネレータ | ○ | ○ | ○ |
| イベントアラータ | ○ | ○ | ○ |
| 配列型 | × | × | ○ |
| 二相コミット | ○ | ○ | ○ |

# テスト環境

- Firebird 1.5.4290 on Windows
- Delphi6 Professional

- IBX version 6.08
- IBO version 4.3.A.a
- FIB version 5.3

- テストテーブル  test.sql

# ジェネレータ (IBX)

- ## IBX --- GeneratorField

  プロパティエディタから設定できるので簡単

  TgeneratorField
  <property>
  ApplyEvent, Field, Generator, IncrementBy
  <method>
  Apply

  <ApplyEvent>
  gamOnNewRecord OnNewRecord イベントの直前に、項目値を生成します。( ※通常使用 )
  gamOnPost BeforePost イベントの直前に、項目値を生成します。
  gamOnServer サーバ側のトリガで Generator 値をセットする場合に使用します。

  gamOnServer は、結局の所何もしないという意味でしかありません。

  ※内部的には、必要に応じて select gen_id( ジェネレータ名 , インクリメント値 ) from rdb$database; を実行している。

# ジェネレータ (IBO)

## ◆IBO --- GeneratorLinks property

<TIBOQuery>
　GeneratorLinks

タイミングは BeforeInsert となります。

書式： [< tablename >.]< columnname >=< generatorname >

GeneratorLinks ではインクリメントが 1 で固定なので、変更したい場合は以下の方法をとります。

use the GeneratorValue function or Gen_ID function of the internal dataset instead and call it in the IBOQuery's BeforeI nsert event. i.e.

```
IBOQuery1.ParamByName('IDField').AsInteger := GeneratorValue(MyGenerator,100);
or
IBOQuery1.ParamByName('IDField').AsInteger := Gen_ID(MyGenerator,100);
```

TIBODatabase/TIB_Database 等 では、　　GEN_ID()
TIBOQuery では、　　GeneratorValue()
TIB_Query では、　　両方ある。

※なんか、統一感がないですね。

# ジェネレータ (FIB)

- FIB --- AutoUpdateOptions

<pFIBDataset>
  AutoUpdateOptions.UpdateTableName = テーブル名
  AutoUpdateOptions.KeyFields = フィールド名
  AutoUpdateOptions.GeneratorName = ジェネレータ名
  AutoUpdateOptions.GeneratorSteps = インクリメント値
  AutoUpdateOptions.WhenGetGenID = [wgBeforePost, wgOnNewRecord, wgNever]

タイミングについては、 IBX と基本的に同じです。

<pFIBDataset>
  IncGenerator

IncGenerator メソッドを呼び出すと、 AutoUpdateOptions で設定されている KeyFields の値が、 Generator 値を使って再設定されます。

※内部的には、必要に応じて select gen_id( ジェネレータ名 , インクリメント値 ) from rdb$database; を実行している。

# イベント (IBX)

- IBX --- TIBEvents

プロパティ

| | |
|---|---|
| AutoRegister:Boolean | イベントの自動登録、コネクションの接続時に自動登録する |
| Events:TStrings | イベントを文字列で登録 |
| Registered:Boolean | True に設定することで、イベントを登録 |

メソッド

| | |
|---|---|
| RegisterEvents | イベントをサーバに登録する |
| UnRegisterEvents | イベントの登録を解除する |

※登録できるイベントの数に制限はありません。

※内部的には、 API で利用する EPB(EventParameterBuffer) に登録できるイベントの数が各 15 個なので、 15 個毎にスレッドを生成してイベントを待機しています。

# イベント (IBO)

- IBO --- TIB_Events

プロパティ

| | |
|---|---|
| Events:TStrings | イベントを文字列で登録 ( 16 個まで ) |
| Interval:Boolean | TTimer ベースでチェックする間隔を指定 |
| Passive:Boolean | True に設定した場合、 CheckEvents でイベントをチェックする |
| Registered:Boolean | True に設定することで、イベントを登録 |

メソッド

| | |
|---|---|
| RegisterEvents | イベントをサーバに登録する |
| UnRegisterEvents | イベントの登録を解除する |
| CheckEvents | Passive モードの時、イベントの発生をチェックする |

※ TIBOSession の IdleTimer イベントを利用してイベントの発生をチェックしているとのことです。多少反応が鈍い感じもしますが、マルチスレッドでの使用に向いているとのこと。

※「 Events に登録出来るイベントの数は 16 個までで、これは InterBase の制限です」という記載が Help にあるが、 API で利用する EPB(EventParameterBuffer) には 15 個までのイベントを登録出来るので、ヘルプの記述は間違いではないかと思います。

# イベント (FIB)

- FIB --- TSIBfibEventAlerter

プロパティ

| | |
|---|---|
| AutoRegister:Boolean | イベントの自動登録、コネクションの接続時に自動登録する |
| Events:TStrings | イベントを文字列で登録 |

メソッド

| | |
|---|---|
| RegisterEvents | イベントをサーバに登録する |
| UnRegisterEvents | イベントの登録を解除する |

※登録できるイベントの数に制限はありません。

※内部的な構造は、 IBX と大体同じです。

# 配列型

## FIBPlus の TFIBArryaField 型

TByteField を継承して、 Firebird の Array 型フィールドを取り扱えるよう実装

プロパティ

| | |
|---|---|
| ArrayID:T_ISC_QUAD | 配列の ID を返す |
| ArraySize | |
| Dimension:T_ISC_ARRAY_BOUND | 配列の下限と上限を返す |
| DimensionCount:Integer | 配列の行数を返す |
| ElementType:TFieldTyep | 配列の要素型を返す |
| Value:Variant | Variant 配列を返す |

※以下はヘルプからの転載です。
You should call Post and Refresh methods after data modifying in array field.

```
with pFIBDataSet1 do
begin
  if not (State in [dsEdit,dsInsert]) then Edit;
  FieldByName('ArrayField').Value := VarArrayOf ([0,1,2,3,4]);
  Post;
  if not (poRefreshAfterPost in Options) then Refresh;
end;
```

※TFIBCustomDataSet.ArrayFieldValue でも、 Variant 配列を得ることが出来る

# 2 相コミット

- IBX --- TIBTransaction
  AddDatabases メソッド
- IBO --- TIB_Transaction
  AddConnection メソッド
- FIB --- TFIBTransaction
  AddDatabase メソッド

※トランザクションが接続と分離しているのが BDE 等との違い