embarcadero®

Developer Camp

【B4】Delphi/C++テクニカルセッション

「DataSnapユースケース研究」

多層技術の概要と最適化、実践テクニック

エンバカデロ・テクノロジーズ
エヴァンジェリスト　高橋智宏

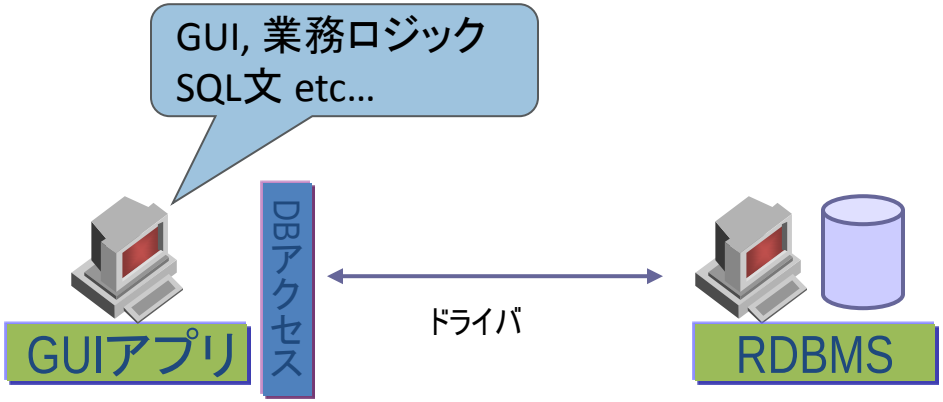
アジェンダ

- DataSnapの基礎
  - プロトコル
  - サーバーメソッド
- 現実的な構成例
  - DMZ + ロードバランス
  - ライフサイクル
- DB接続のレイヤ
  - 親クラス
  - FireDAC
- デモ
  - マルチデバイス対応
- Tips
  - HTTPSを利用するには?
  - バイナリデータの送受信
  - サーバーメソッドの非同期呼び出しは?

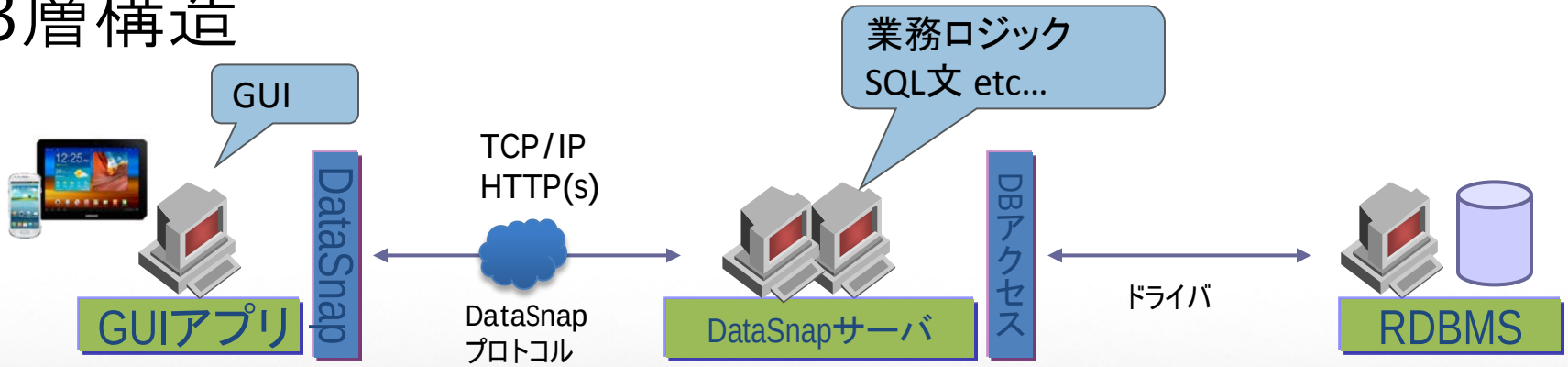
DataSnapの基礎

DataSnapとは?

- 2層のC/S



- 3層構造



検討事項

- 特にセキュリティ面と運用面
  - RDBMSの「ユーザー名」と「パスワード」の管理
  - ネットワーク上、直接RDBMSに接続出来てしまう
  - dbExpress/dbGo/FireDACなどのDBコンポーネント
    - コンポーネントのアップデート → *全端末への再配布*
  - RDBMSのネイティブクライアントライブラリ
    - サポートされているプラットフォームは??
    - インストーラ
    - モジュールのアップデート → *全端末への再配布*
  - ビジネスロジックやSQL文がクライアント側に存在
    - アプリケーションのアップデート → *全端末への再配布*
  - ファイアウォール

  etc...

DataSnapプロトコル

- TCP/IP
  - デフォルト211番ポート
  - SSL, IPv6 は未サポート
- HTTP
  - デフォルト80番ポート
  - HTTPSもサポート
- RPC形式
  - サーバーメソッド
    - **要は、RDBMSのストアドプロシージャと同じ扱い!!**
    - パラメータ & 戻り値
      - in, var, out, return
    - 組み込みの非同期呼び出しは無い
  - インスタンスのライフサイクル
    - Server, Session, Invocation
  - Callback, Filter, Event
- データフォーマット
  - JSON(JavaScript Object Notation)

クライアントからのリクエストのサンプル

```
{"method":"connect","params":"drivername":"DATASNAP",
"port":"211","communicationprotocol":"tcp/ip","hostna
me":"127.0.0.1","DriverUnit":"DbxDatasnap",....]}
{"method":"execute","params":[{"fields":[-
1,false,"Dbx.MetaData","GetDatabase"]}]}
{"method":"reader_close","params":[1,0]}
{"method":"disconnect","params":[0]}
```

サーバーからのレスポンスのサンプル

```
{"result":[0,"DataSnap",2]}
{"result":[{"rows":[0]},{"data":[1,`]},{"table":[{"fi
elds":[0,false,1,0,true]},{"columns":[13,["QuoteChar"
,26,0,0,0,16,0,0],["ProcedureQuoteChar",26,0,0,0,16,0
,0],["MaxCommands",6,0,0,0,0,0,0],["SupportsTransacti
ons",4,0,0,0,0,0,0],["SupportsNestedTransactions",4,0
,0,0,0,0,0],["SupportsRowSetSize",4,0,0,0,0,0,0]...
...
...
```

サーバーメソッドとは?

- リモートプロシージャコール(RPC)
  - http://ja.wikipedia.org/wiki/RPC
- サーバーメソッドの実装 & エクスポート
  - 引数の型
  - 戻り値
  - 例外は?
  - クライアント用プロキシは?

```
unit MyClass;

interface
uses Classes;
type
  {$METHODINFO ON}
  TMyClass = class(TComponent)
    function Sum(const A, B: Double): Double;
  end;
  {$METHODINFO OFF}

implementation
{ TMyClass }
function TMyClass.Sum(const A, B: Double): Double;
begin
  Result := A + B;
end;

end.
```

サーバーメソッドで扱える型

- 基本型
  - AnsiString
  - Boolean
  - Byte
  - Currency
  - TDateTime
  - TDBXDate
  - TDBXTime
  - Double
  - Int64
  - Integer
  - LongInt
  - OleVariant
  - Single
  - ShortInt
  - SmallInt
  - String(UnicodeString)
  - WideString
  - TStream

- DBXValue型
  - TDBXAnsiStringValue
  - TDBXAnsiCharsValue
  - TDBXBcdValue
  - TDBXBooleanValue
  - TDBXDateValue
  - TDBXDoubleValue
  - TDBXUInt8Value
  - TDBXInt8Value
  - TDBXInt16Value
  - TDBXInt32Value
  - TDBXInt64Value
  - TDBXJSonValue
  - TDBXReaderValue
  - TDBXSingleValue
  - TDBXStringValue
  - TDBXTimeStampValue
  - TDBXTimeValue
  - TDBXWideCharsValue
  - TDBXWideStringValue
  - TDBXStreamValue

- テーブル型
  - TDataSet
  - TParams
  - TDBXReader

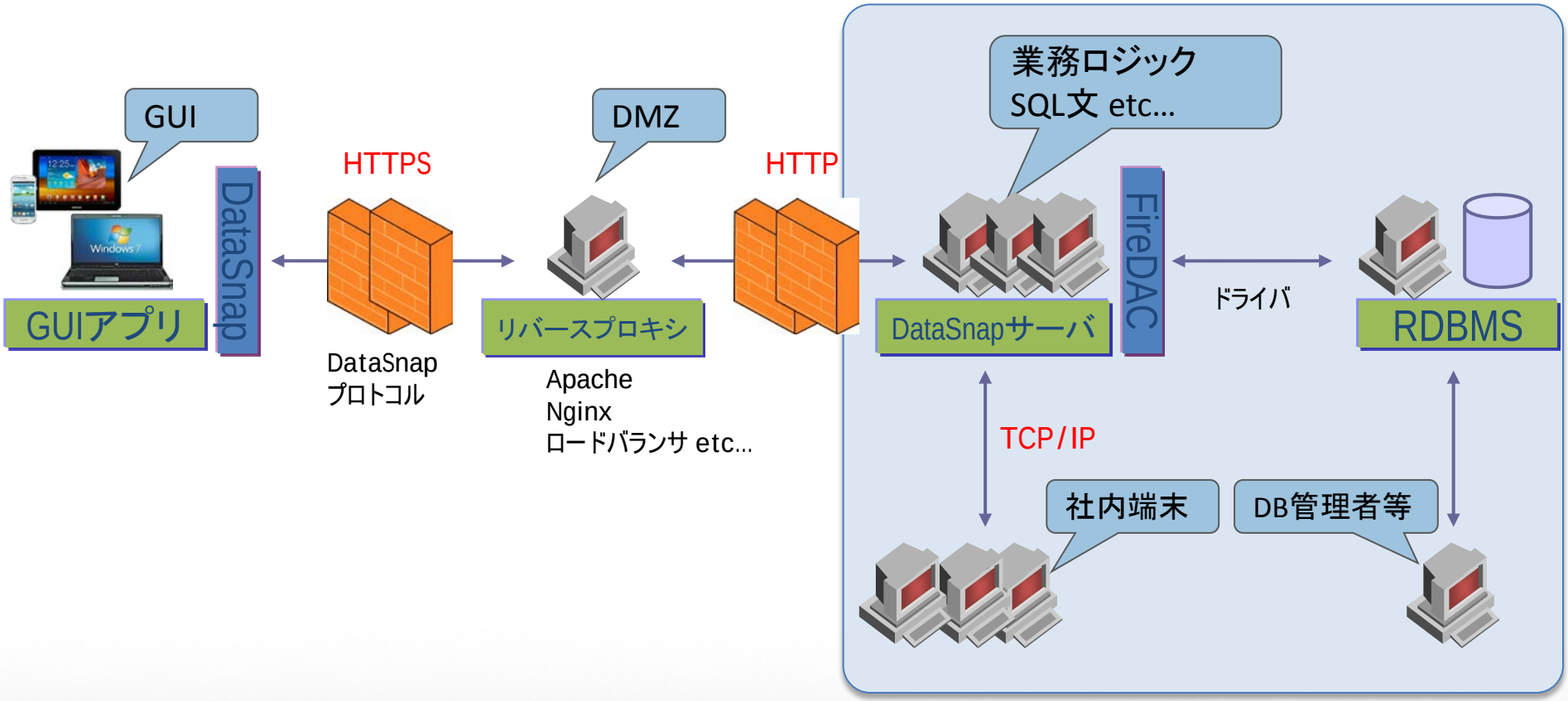
- JSON型
  - TJSONArray
  - TJSONNumber
  - TJSONObject
  - TJSONString
  - TJSONValue

基本型	NULLサポート無し。 varや戻り値もOK
DBXValue型	NULLサポートあり。 varや戻り値はNG
テーブル型	varや戻り値もOK
配列型 (array of …)	サポートされません。 TJSONArrayを使う

現実的な構成例

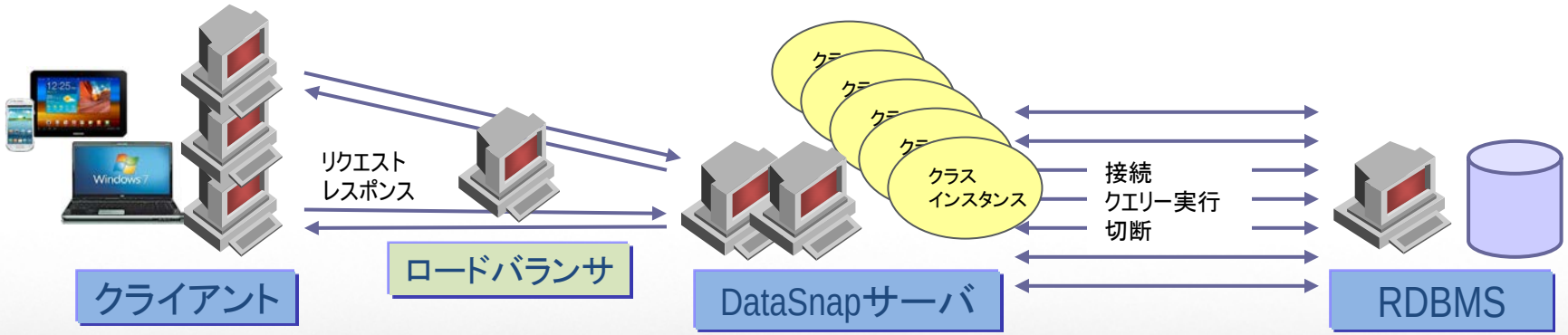
システム構成例

- 複数のDataSnapサーバーをロードバランス



ロードバランスすると…

- サーバーメソッド用クラスのライフサイクルとの兼ね合い
  - Session (デフォルト)
    - クライアントアプリケーションからの接続とインスタンスが結びつく
    - ステートフル
      - ステート(セッション情報)のリプリケーションは??
  - Invocation
    - サーバーメソッドの呼び出しごとにインスタンスの生成&破棄を繰り返す
    - ステートレス



DB接続のレイヤ

サーバーメソッドクラスの構造

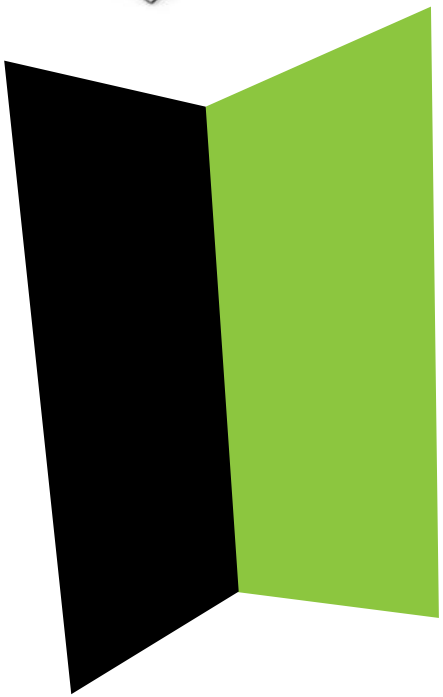
- 親クラス
  - TComponent, TDataModule, TDSServerModule
- データアクセス
  - IBExpress
  - dbExpress
  - dbGo(ADO)
  - FireDAC

  etc…

- 親クラスをTComponentで、データモジュールをプーリング?
- コネクションプーリング
  - 独自実装
  - FireDAC

DBアクセスモジュールの設計

- FireDACで使用する主なコンポーネント
  - TFDConnection
  - TFDTable / TFDQuery など
  - TFDPhysIBDriverLink など
  - TFDGUIxWaitCursor (ProviderプロパティはConsole)
- データモジュール
  - データベースアクセスをカプセル化
  - サーバーメソッドの処理とデータベース処理を分離

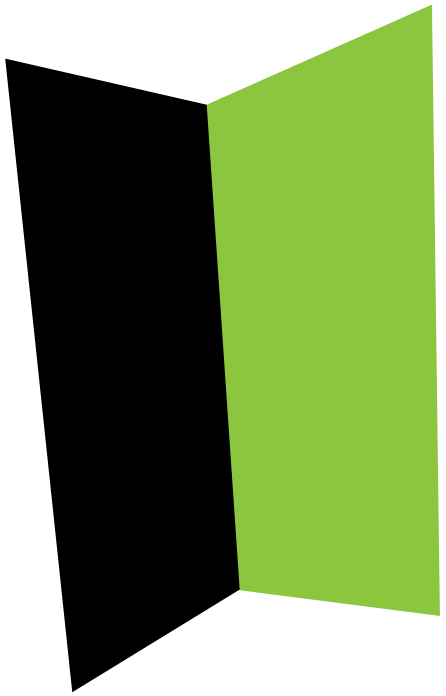


デモ

- 複数デバイス対応 -

サーバー & クライアントの例

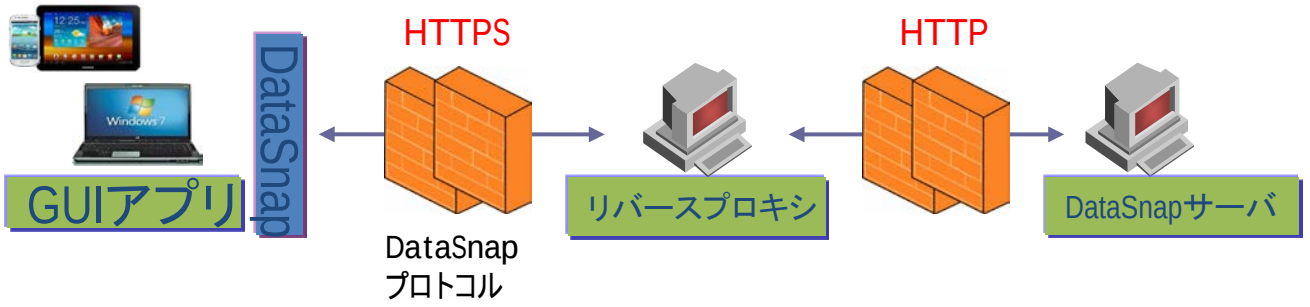
- サーバー
  - Delphi 64bit スタンドアロン
  - TCP/IP + HTTP
  - TComponent(親) + データモジュール
  - FireDAC
- クライアント
  - Delphi VCL
  - Delphi iOS
  - Delphi Android
  - C++Builder OS X



Tips

HTTPSを利用するには?

- リバースプロキシでHTTPSを処理し、HTTPのDataSnapにリダイレクトする
  - http://blogs.embarcadero.com/teamj/2013/10/29/4096/



- DataSnap単体でもOpenSSL(Indyで利用)を使用したHTTPS接続を受付可能
  - http://blogs.embarcadero.com/teamj/2013/10/30/4103/
- Windows ServerのIISを利用する
  - ISAPI(DLL)モードでDataSnapサーバーを実装
  - IIS自体でHTTPSを処理する

- クライアントは、プロトコルにhttpsを指定するだけでOK

バイナリデータの送受信

- TStreamを使う
  - 高速
  - もちろん、TJSONAarrayにByte配列を入れる方法もOKですが…

Server側

```
function TTestServer.Download(…): TStream;
var
  memStream: TBytesStream;
begin
  Result := nil;
  FileName := …;
  if FileExists(FileName) then
  begin
    memStream := TBytesStream.Create;
    memStream.LoadFromFile(FileName);
    Result := memStream;
  end;
end;
```

Client側

```
procedure TForm1.BtnClick(Sender: TObject);
var
  stream: TStream;
  b: Byte;
  mem: TBytesStream;
begin
  stream := TestSV.Download('xyz.jpg');
  mem := TBytesStream.Create;
  while stream.Read(b, 1) = 1 do
    mem.Write(b, 1);
  …
  mem.Free;
end;
```

サーバーメソッドの非同期呼び出しは?

- VCL(Windows)向けなら OmniThreadLibrary が便利
  - http://code.google.com/p/omnithreadlibrary/
  - 最新版は 3.03a で、Delphi 2007～XE5 (32bit/64bit)をサポート

```
uses OtlTaskControl, OtlParallel;

procedure TfrmDemoParallelAsync.btnAsyncClick(Sender: TObject);
begin
  btnAsync.Enabled := False;
  Parallel.Async(
    procedure
    begin
      // バックグラウンドスレッドで実行
      Sleep(500);
      MessageBeep($FFFFFFFF);
    end,
    Parallel.TaskConfig.OnTerminated(
      procedure (const task: IOmniTaskControl)
      begin
        // メイン(GUI)スレッドで実行
        btnAsync.Enabled := True;
      end
    )
  );
end;
```

- VCL以外(FireMonkey, iOS, Android)は
  - TThreadの派生クラスとSynchronizeメソッドなどを組み合わせる