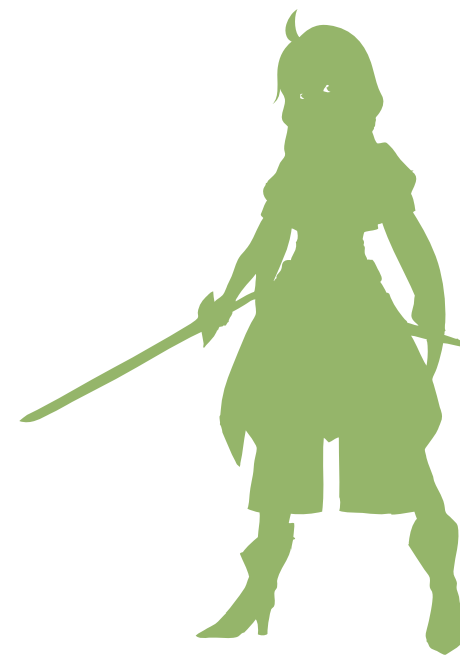


# モバイルからクラウドサービスまで 活用が広がる！ Delphiではじめる多層アプリ開発

第35回 エンバカデロ・デベロッパーキャンプ

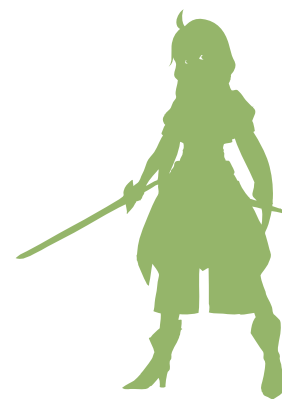
株式会社ミガロ.  
RAD事業部 技術支援課 吉原 泰介



**e**mbarcadero®  
DEVELOPER CAMP

# はじめに

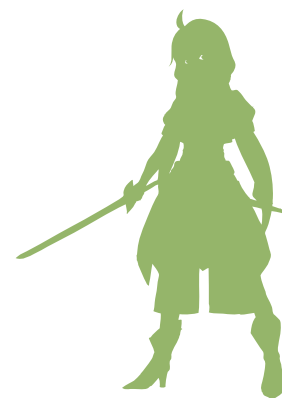
- 中間サーバを使った多層アプリケーション開発はされていますか？  
仕組みが面倒だと思われがちですが、実装するメリットも沢山あります。  
もちろんモバイルアプリケーションだけではなく、  
デスクトップアプリケーション（もちろんVCLも！）にも有効な手法です。



# アジェンダ

- 多層アプリケーションとは？
- 中間サーバのアプリケーションを開発する手法
- DataSnap Server を使った実装手順
- RAD Serverを使った実装手順
- 中間サーバの活用範囲を拡張
  - Enterprise Connectorsを利用したクラウドサービス等の連携
  - Sencha等のWebアプリケーションへの連携

# 多層アプリケーションとは？



# 多層アプリケーションとは？

- 多層アプリケーションとは、複数層で構成されたアプリケーションのこと。例えばClient/Serverアプリケーションは、クライアント層とデータ層で処理される2層構成、Webアプリケーションやモバイルアプリケーションでデータベースにアクセスする場合は、中間層のサーバを経由する3層構成となっている。



# 多層アプリケーションとは？

## ■ よく使われるアプリケーションの構成

### 【2層アプリケーションの構成】

Client/Serverアプリケーション



クライアント



データベースサーバ

### 【3層アプリケーションの構成】

Webアプリケーション



クライアント (ブラウザ)



Webサーバ



データベースサーバ

モバイルアプリケーション



クライアント



中間サーバ



データベースサーバ

# 多層アプリケーションとは？

## ■ 中間サーバを活用したアプリケーション構成

【中間サーバを活用したアプリケーションの構成例】

Client/Serverアプリケーション



Webアプリケーション



クライアント (ブラウザ)

モバイルアプリケーション



クライアント



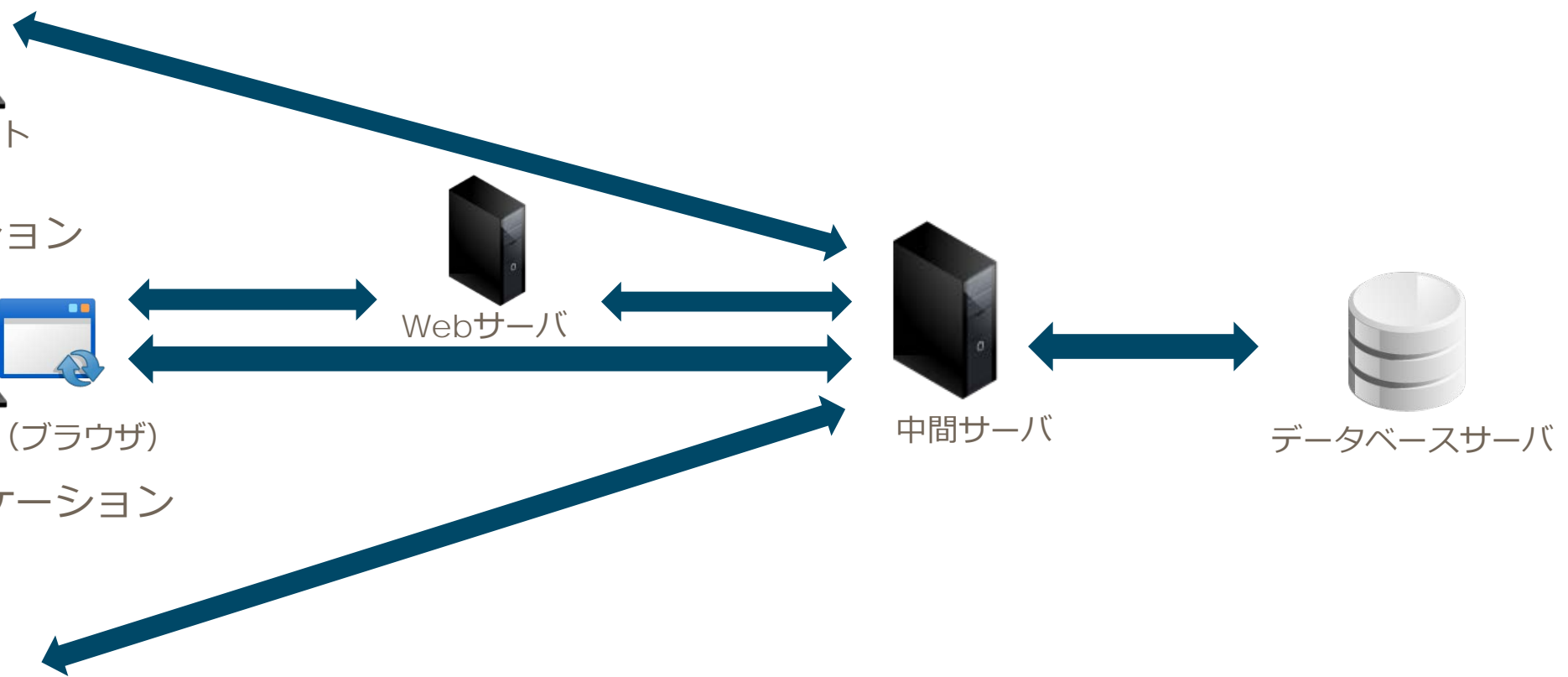
Webサーバ



中間サーバ



データベースサーバ



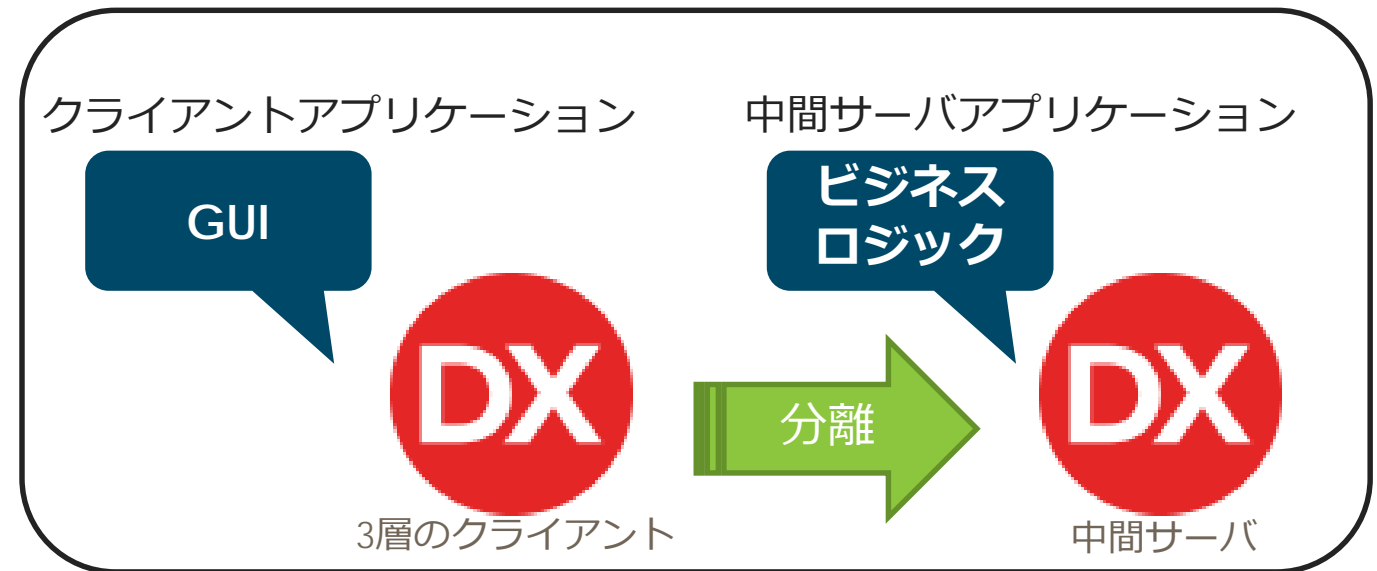
# 中間サーバを経由する構成のメリット

## ■ メリット①

### ビジネスロジックの分離

画面などのGUI部分とデータ側の業務処理を含むビジネスロジック部分を分離して開発・メンテナンスすることができる。

これによりアプリケーションの実装内容は目的別のシンプルな内容になる。





# 中間サーバを経由する構成のメリット

## ■ メリット②

### クライアントアプリケーションのダウンサイズ・配布の軽減

ユーザー環境へ配布するクライアントアプリケーションはビジネスロジックを含まないため、実行ファイルのサイズを小さく抑えることができ、変更や再配布の頻度も減らすことができる。

(ビジネスロジック側の変更でクライアントアプリケーションは変わらない)  
また使用するDB関連の導入や設定等もクライアント側では不要になる。

DBクライアント  
導入してる？

バージョンは？  
環境設定は？



2層のクライアント

DB関連は中間サーバ側で  
導入・設定するので不要



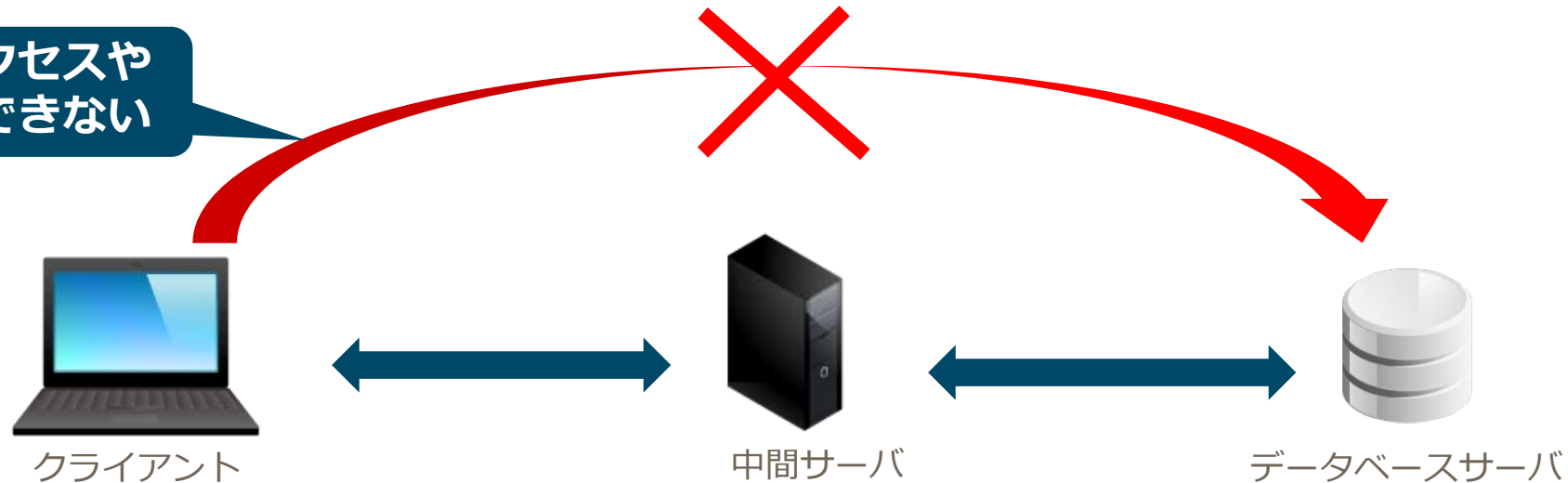
3層のクライアント

# 中間サーバを経由する構成のメリット

## ■ メリット③ セキュリティ

クライアント環境からDBに直接アクセスできない構成で環境を構築できる。特に外部からも利用する環境を考える場合には、中間サーバの環境でセキュリティ上の配慮・対策がしやすい。またデータ通信もHTTPS等による暗号化データでやりとりも可能。

直接アクセスや  
操作ができない



# 中間サーバを経由する構成のメリット

## ■ メリット④

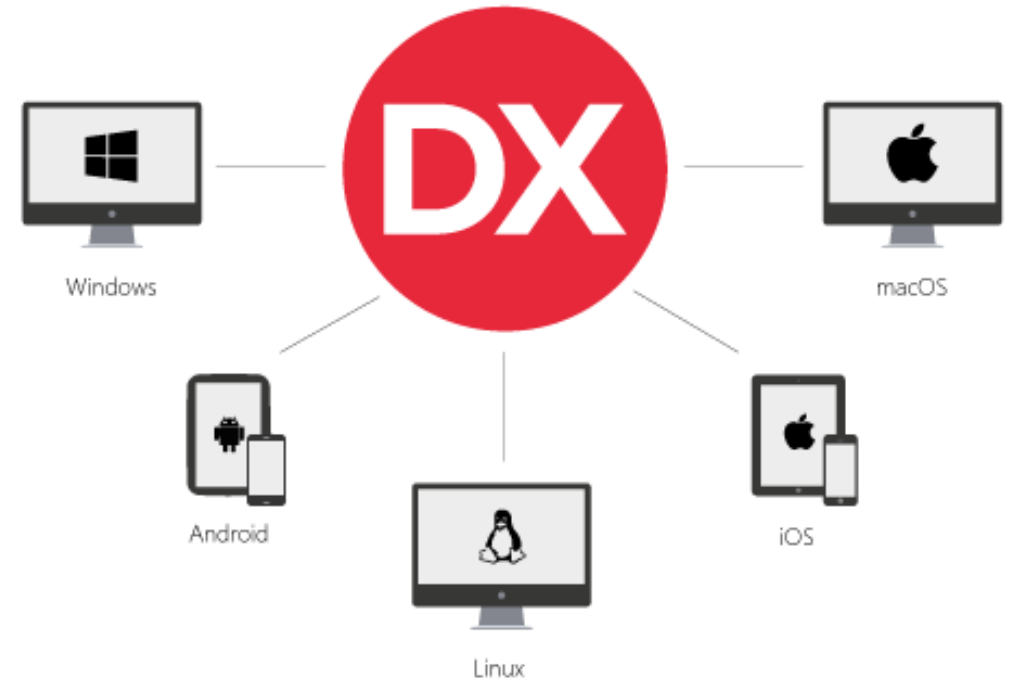
### マルチデバイス対応・展開

ビジネスロジックを中間サーバ側のアプリケーションで持つことで、クライアントアプリケーションがデバイス毎に異なっても共通で対応できる。

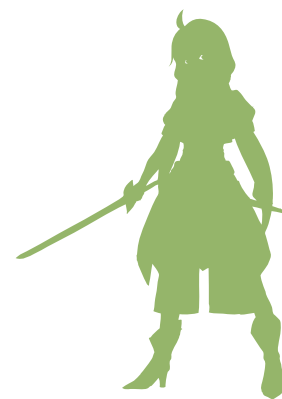
(同じビジネスロジックをデバイス毎に作成不要)

Delphiでマルチデバイス開発したアプリケーションであればワンソースでの管理もでき、中間サーバ側との親和性も高い。

Delphiで開発したマルチデバイスアプリは中間サーバアプリと親和性が高い



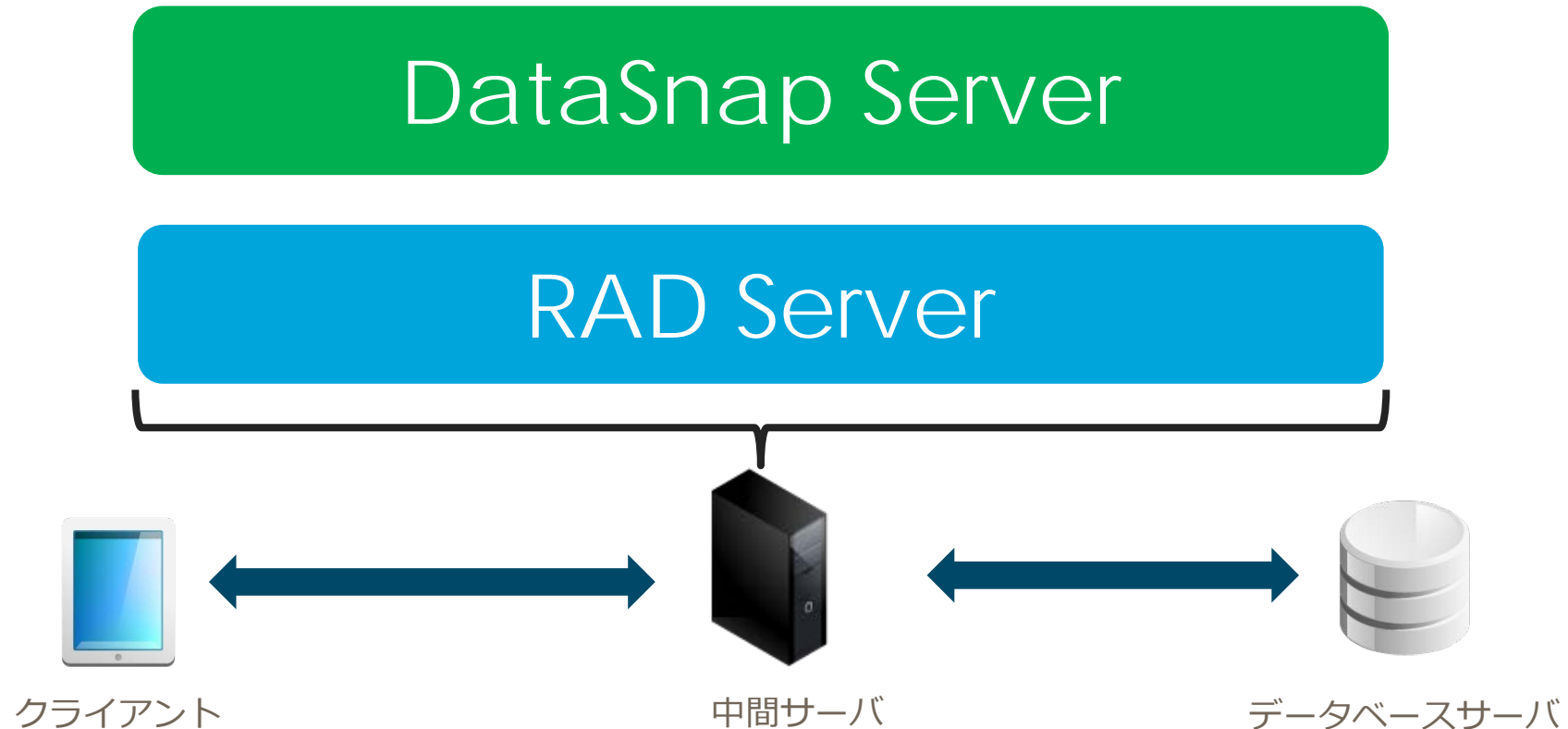
# 中間サーバのアプリケーションを開発する手法



**e**mbarcadero®  
DEVELOPER CAMP

# 中間サーバのアプリケーションを開発する手法

- Delphiを使って中間サーバ側のアプリケーションを開発する手法はいくつかありますが、今回は主な2手法をご紹介します。



# DataSnap Serverとは？

## ■ 特徴

- ・ 多層アプリケーションの開発を可能にするSDK
- ・ サーバ機能はプログラムで開発する必要がある
- ・ 開発での実装となるため、プログラムの自由度が高い
- ・ TCP/IP 、 HTTP (S)、 REST、 JSON、 COMなどの標準技術をサポート

# RAD Serverとは？

## ■ 特徴

- ・ 多層アプリケーションのためのREST APIを構築／公開するサーバー
- ・ サーバで必要となる高度な機能がいくつも提供されている
- ・ ユーザー管理機能、認証機能、分析機能などの標準機能を豊富に搭載
- ・ HTTP (S)、 REST、 JSONなどの標準技術をサポート

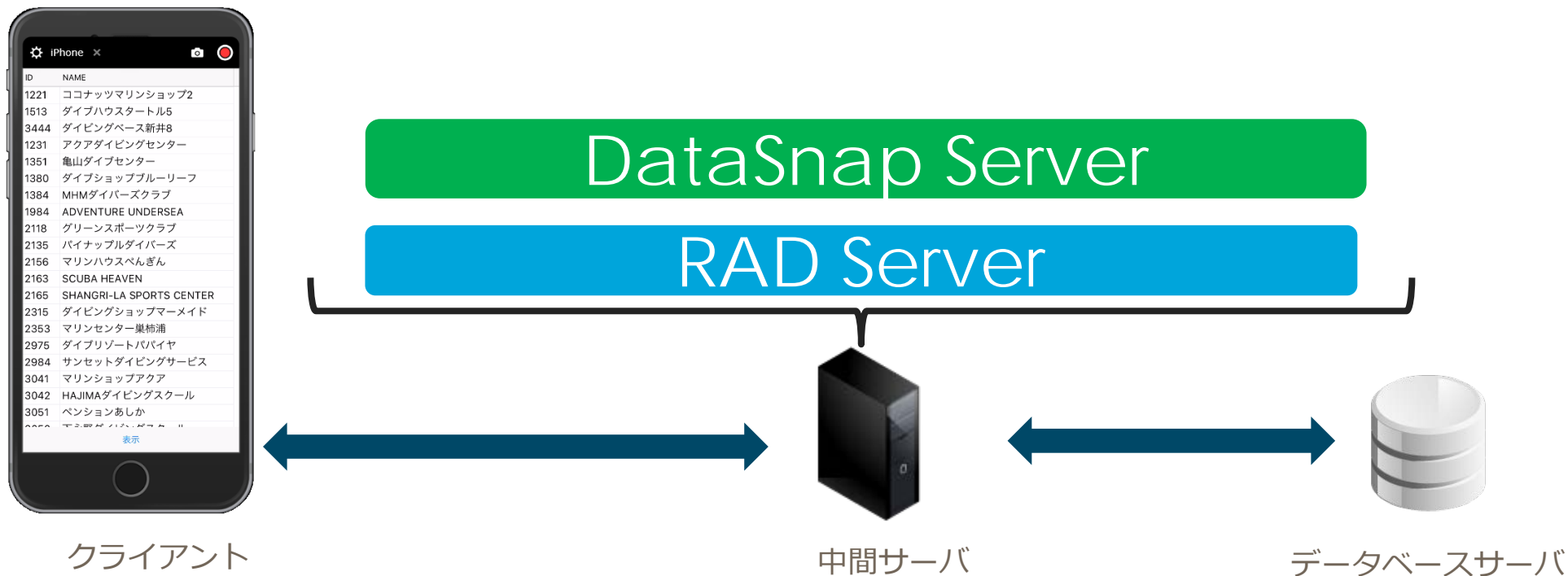
# DataSnap Server と RAD Serverの主な違い

	DataSnap Server	RAD Server
機能開発	全て開発で実装が必要	必要な部分のみ開発
標準通信	TCPIP/HTTP(S)	HTTP(S)
DBエンジン	FireDAC、dbExpress	FireDAC
モバイル対応機能	開発が必要	Push通知、デバイス認証等が標準機能
管理ツール	開発が必要	標準で付属（分析も可能）
ライセンス	開発ライセンスに含まれる (Enterprise以上)	<b><u>開発ライセンスに1サイトライセンス付属 (Tokyo Enterprise以降)</u></b>

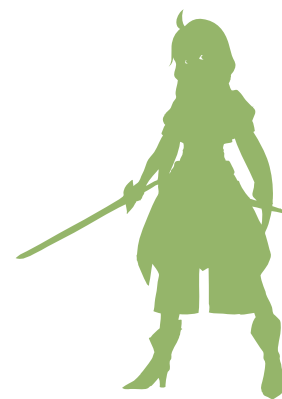


# DataSnap Server と RAD Serverの実装手順

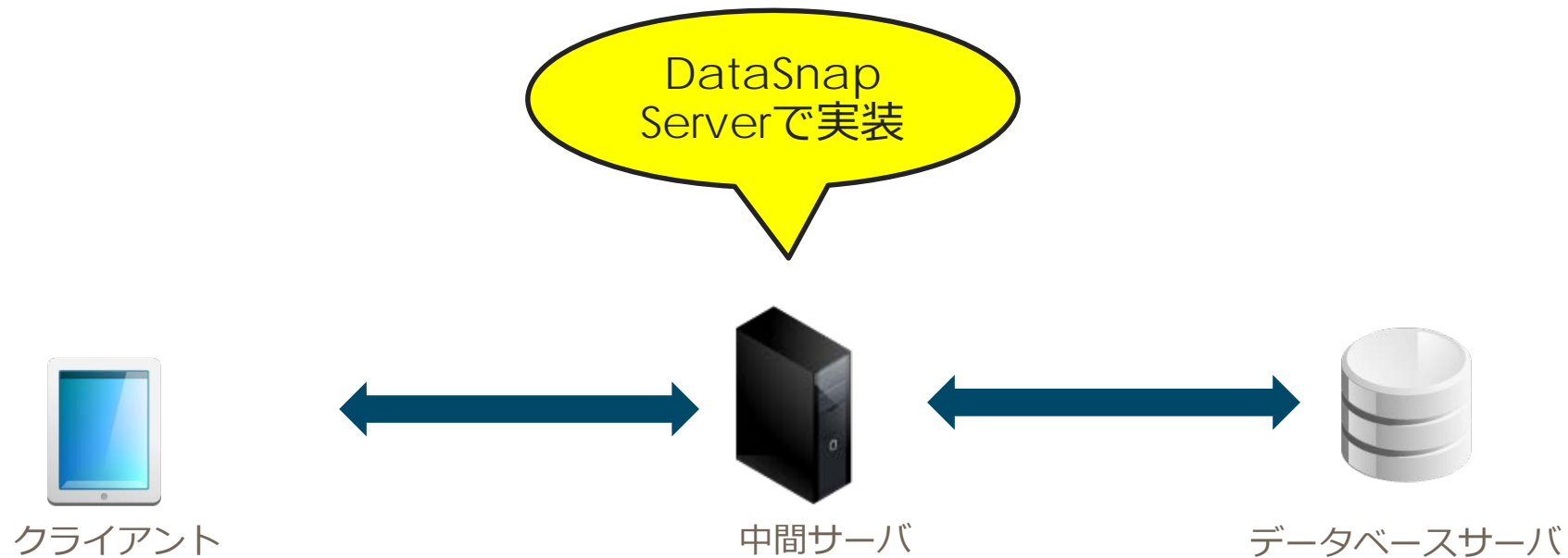
- DataSnap ServerとRAD Serverでは中間サーバで担う役割は似ているが、実装内容や手順は異なる。  
単純なデータアクセスを題材に実装手順の違いをご説明。



# DataSnap Server を使った実装手順

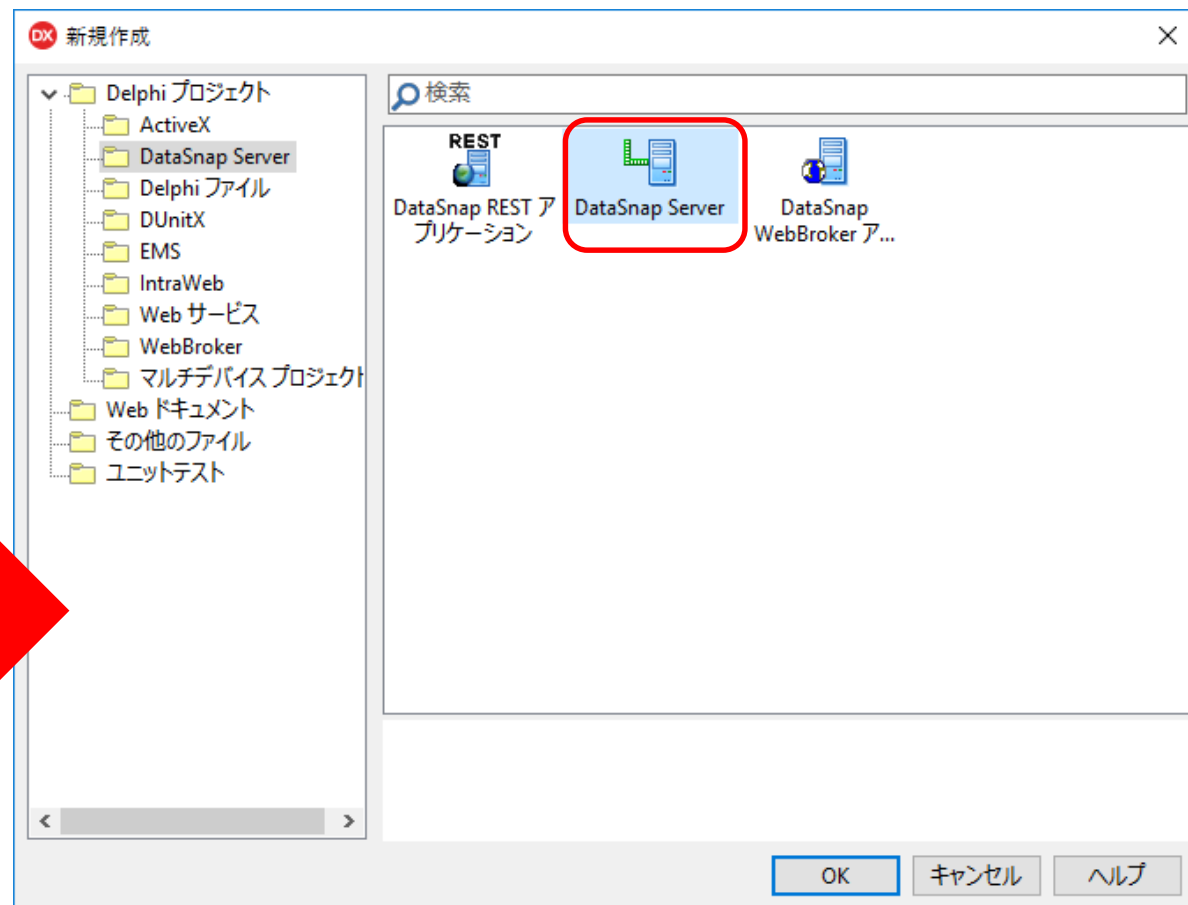
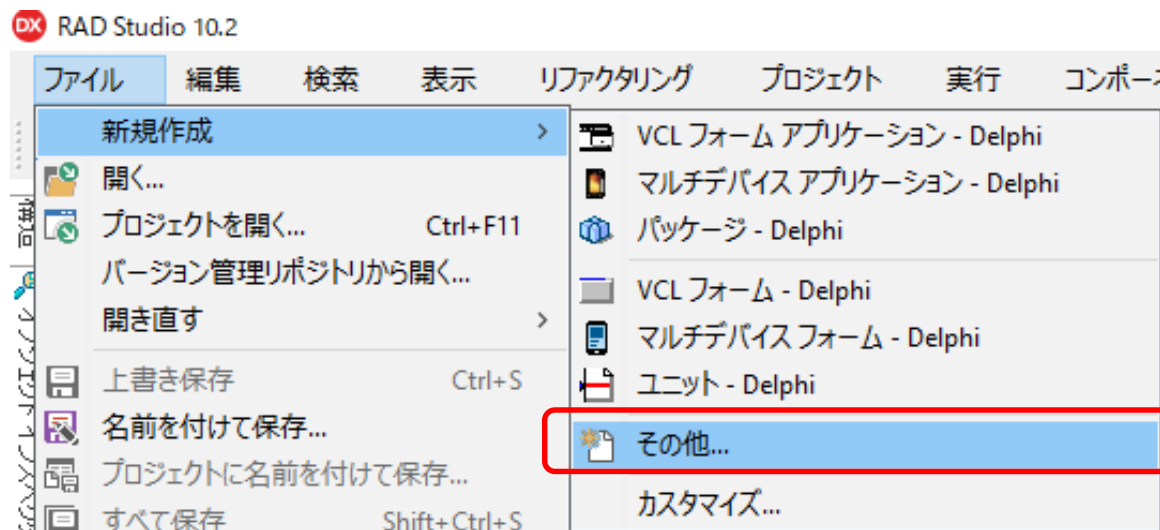


# DataSnap Serverを使った実装手順（ウィザード）



# DataSnap Serverを使った実装手順（ウィザード）

- 実装手順1  
プロジェクトの作成



# DataSnap Serverを使った実装手順（ウィザード）

## ■ 実装手順2 ウィザードの指定

The image displays three sequential screenshots of the DataSnap Server wizard, illustrating the steps to create a new server. Red arrows indicate the flow from left to right. The 'Next' button in each screenshot is highlighted with a red box.

**スクリーンショット 1: プラットフォームの指定**  
作成するアプリケーションのプラットフォームを選択します  
アプリケーションのプラットフォームの種類を指定します  
 Windows  
 Linux

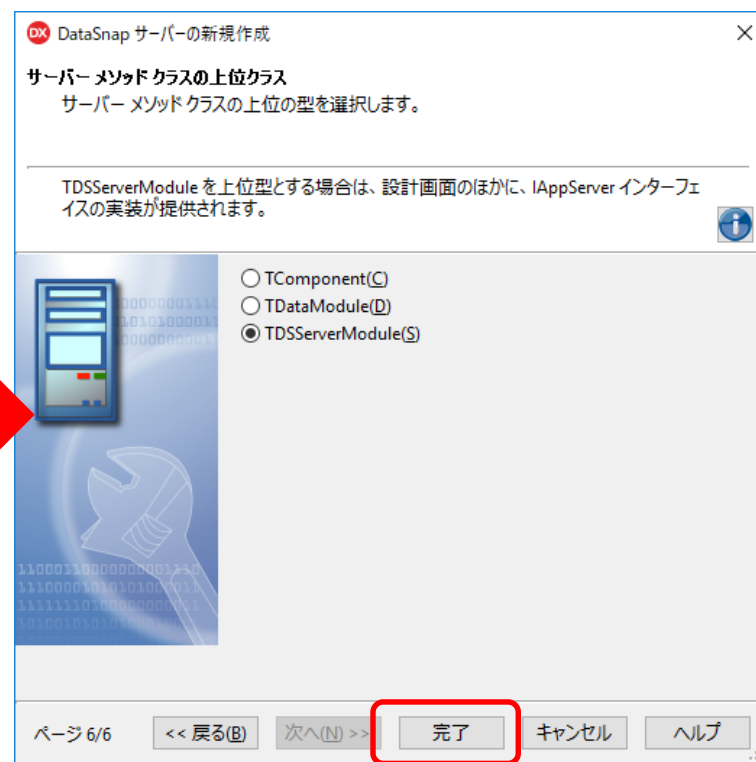
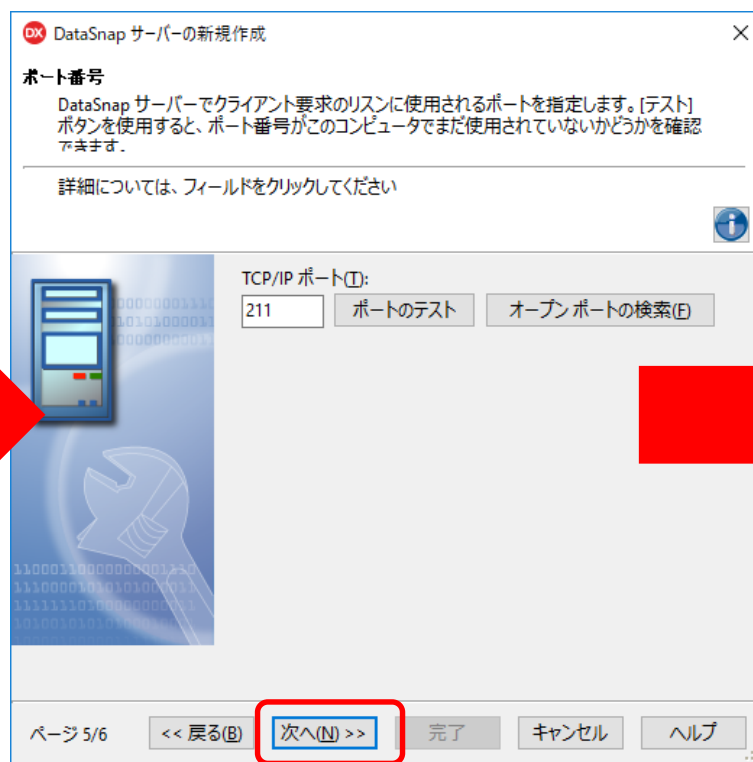
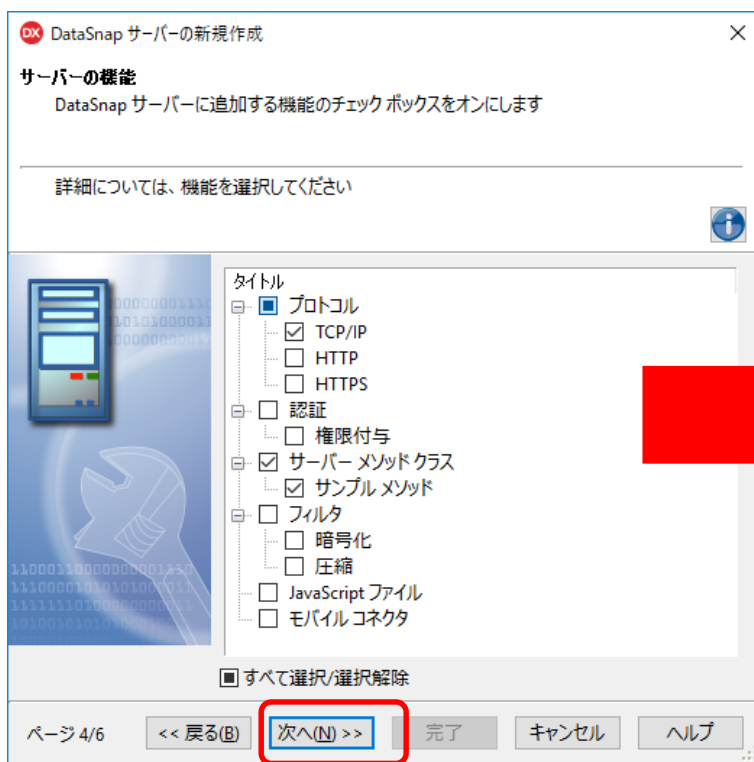
**スクリーンショット 2: プロジェクトの種類を指定**  
このアプリケーションの種類を指定します  
フォーム アプリケーションにはフォームが表示されます  
 フォーム アプリケーション(F)  
 コンソール アプリケーション(C)  
 サービス アプリケーション(S)

**スクリーンショット 3: 作成するアプリケーションの種類を選択**  
作成するアプリケーションの種類を選択します  
VCL アプリケーションを作成します  
 VCL アプリケーション  
 FireMonkey アプリケーション

テスト時はフォームアプリケーションが手軽だが、運用時はサービスアプリケーションが現実的（同じソースでプロジェクトだけ分けておくと便利）

# DataSnap Serverを使った実装手順（ウィザード）

## ■ 実装手順3 ウィザードの指定

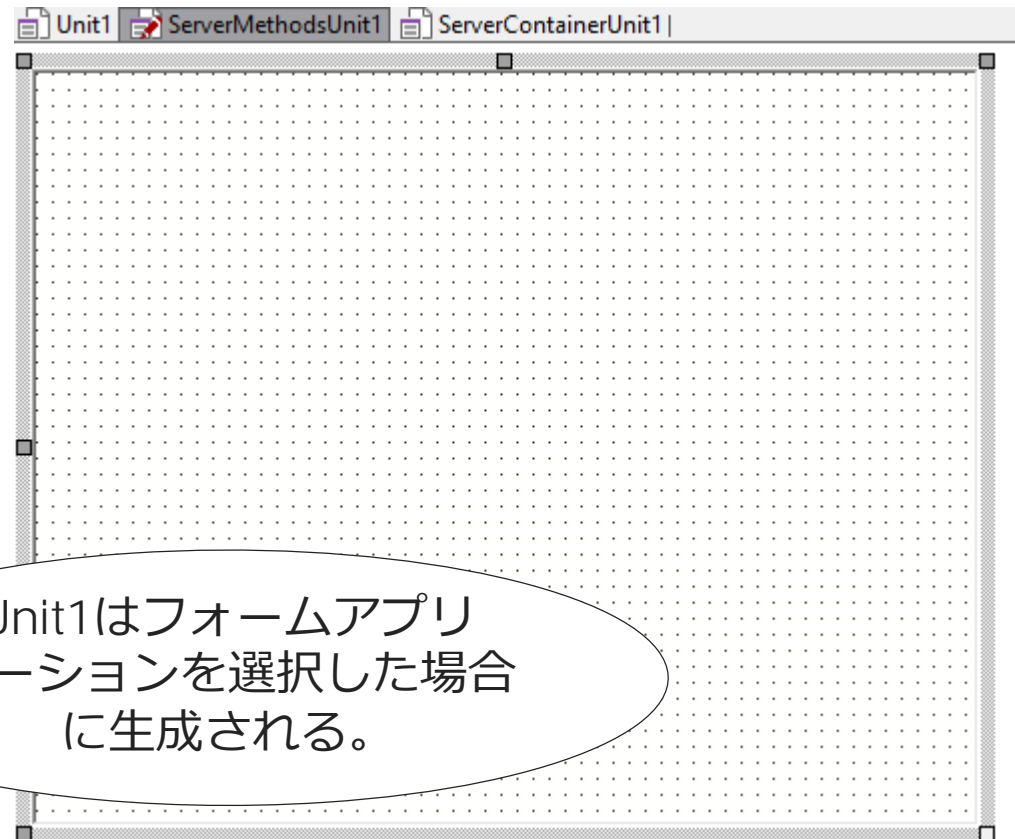
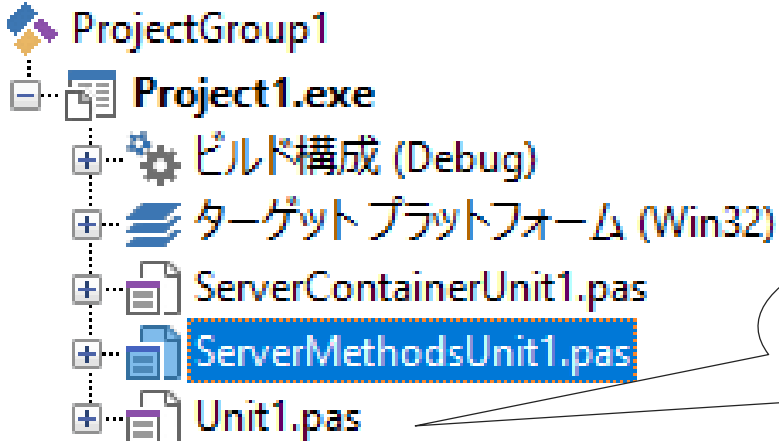


# DataSnap Serverを使った実装手順（ウィザード）

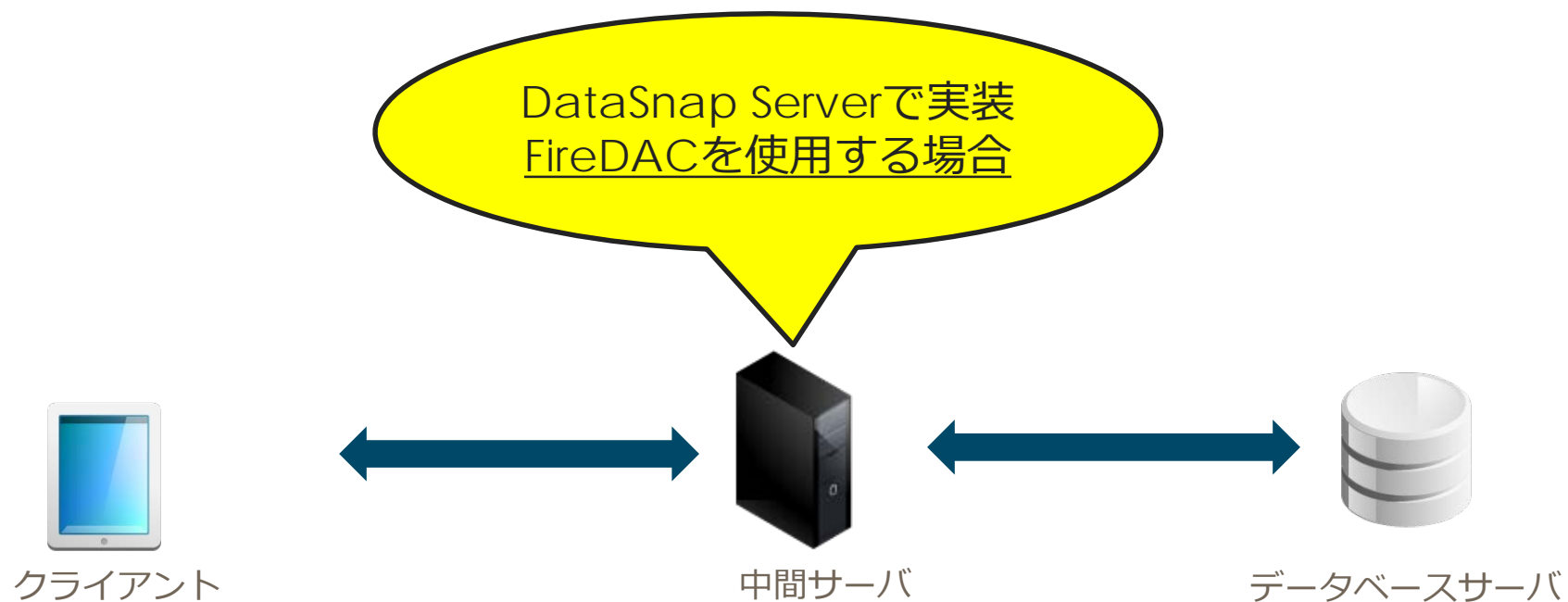
- 実装手順4  
自動生成されるユニット



ファイル



# DataSnap Serverを使った実装手順（FireDAC）



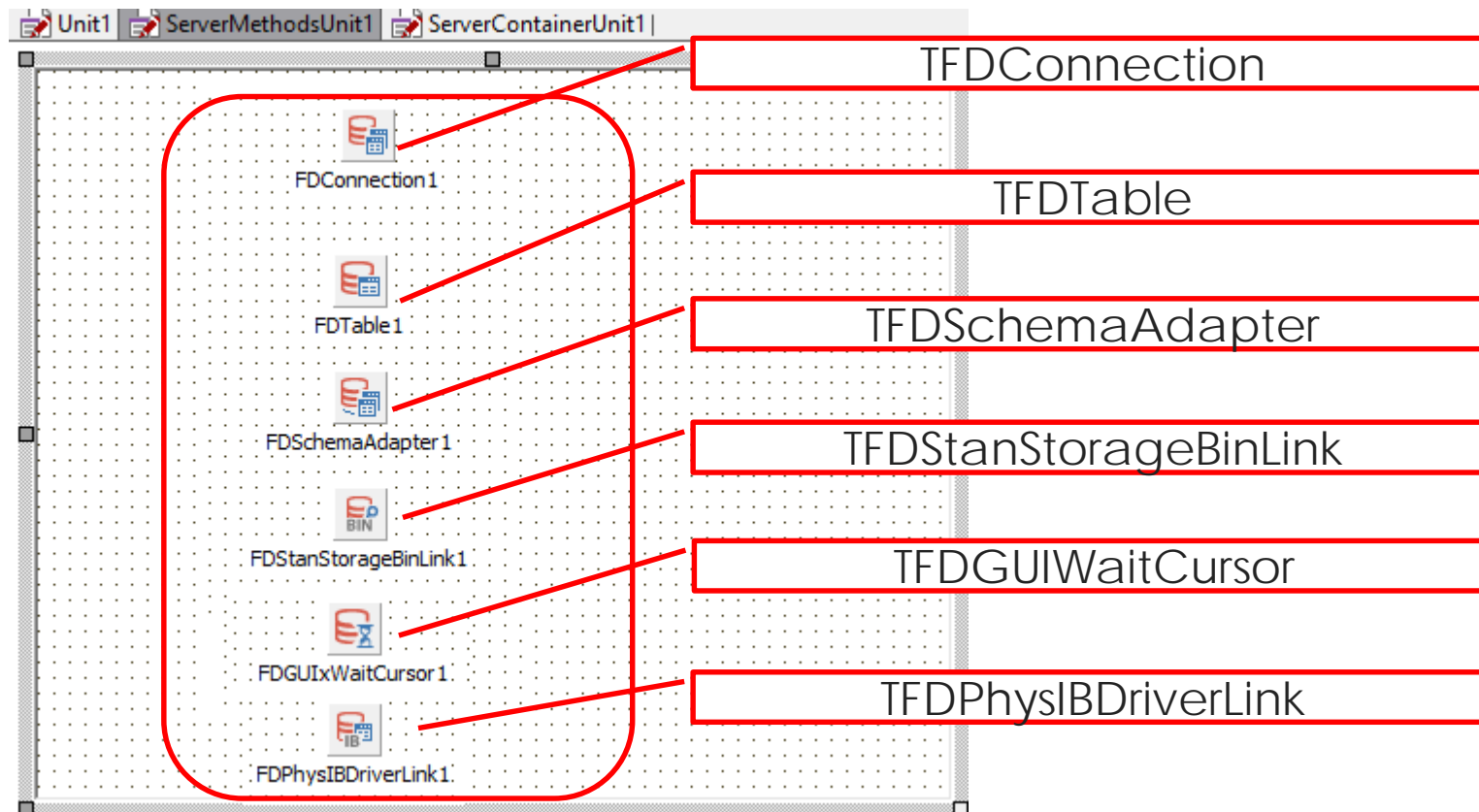


# DataSnap Serverを使った実装手順（FireDAC）

## ■ 実装手順5

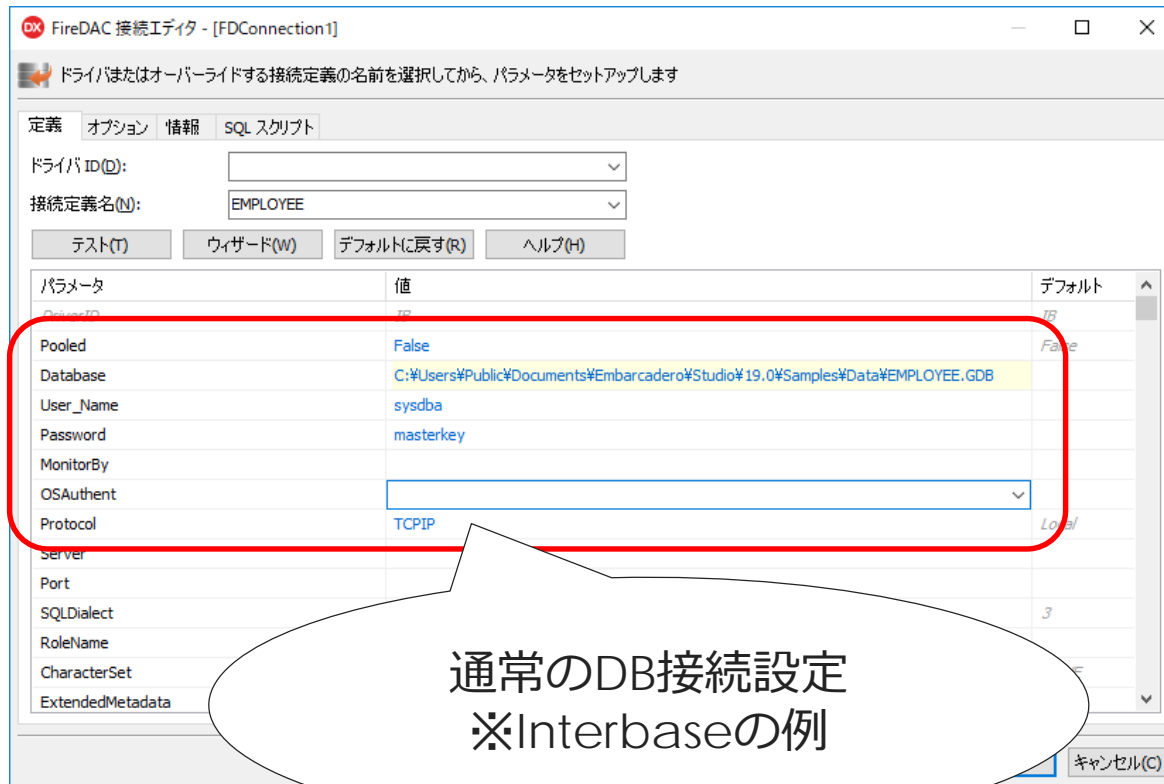
サーバ機能のコンポーネント配置

ServerMethodsユニットに機能を実装

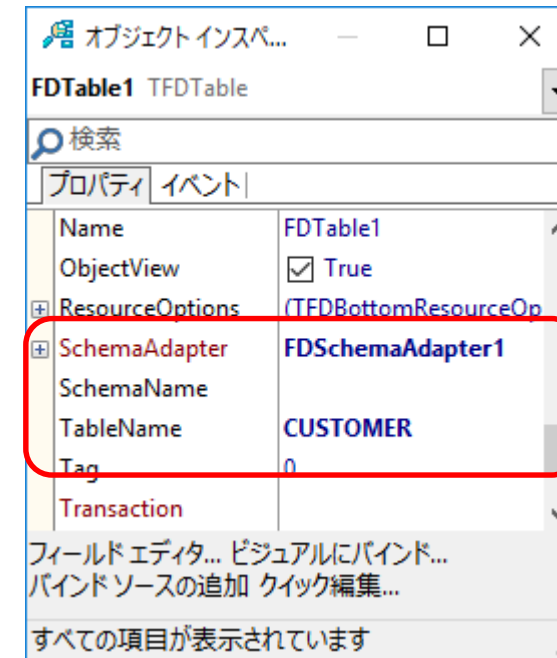


# DataSnap Serverを使った実装手順（FireDAC）

- 実装手順6  
コンポーネントの設定  
TFDConnection



## TFDTable



# DataSnap Serverを使った実装手順（FireDAC）

## ■ 実装手順7

データ取得の機能を実装

usesにIPPeerClientを追加しておく。

```
function TServerMethods1.GetTable: TStream;
begin
  Result := TMemoryStream.Create;
  try
    FDTTable1.Close;
    FDTTable1.Open;
    //TFDSchemaAdapterを経由してStream形式で結果をセットする
    FDSchemaAdapter1.SaveToStream(Result, TFDStorageFormat.sfBinary);
    Result.Position := 0;
  except
    raise;
  end;
end;
```

# DataSnap Serverを使った実装手順（FireDAC）

DataSnap Serverで実装  
FireDACを使用する場合



クライアント



中間サーバ

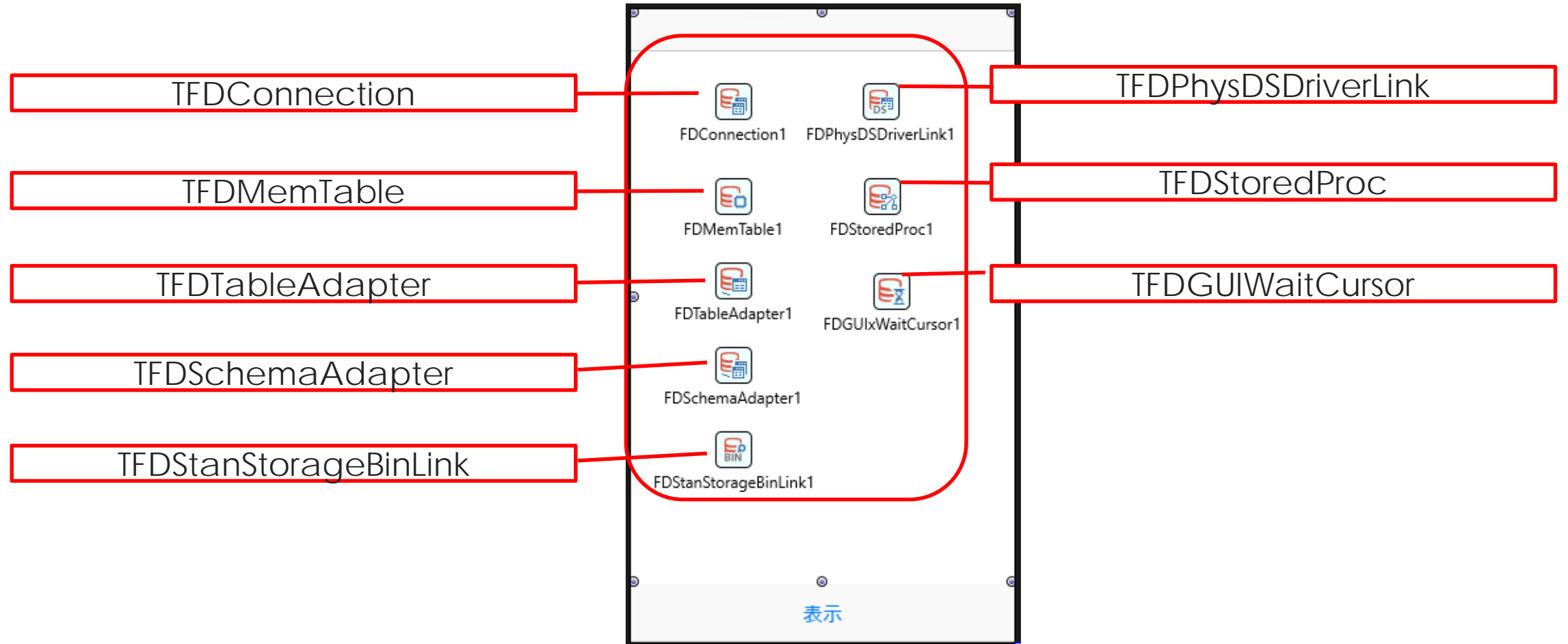


データベースサーバ

# DataSnap Serverを使った実装手順（FireDAC）

## ■ 実装手順8

クライアントアプリのコンポーネント配置



# DataSnap Serverを使った実装手順（FireDAC）

## ■ 実装手順9 コンポーネントの設定

### TFDConnection

ドライバはDS (DataSnap) を指定

サーバIPやポートを設定  
※localhostでは開発端末上でしか接続できません。

パラメータ	値	デフォルト
DriverID	DS	DS
Pooled	False	False
Database		
User_Name		
Password		
MonitorBy		
DBXAdvanced		
Protocol		
IPImplementationID		
CommunicationIPVersion		
Server	999.999.999.999	<LOCAL>
Port	211	211
BufferKbSize	32	32
Filters		

### TFDMemTable

Adapter: FDTTableAdapter1

### TFDTableAdapter

サーバ側のデータセット

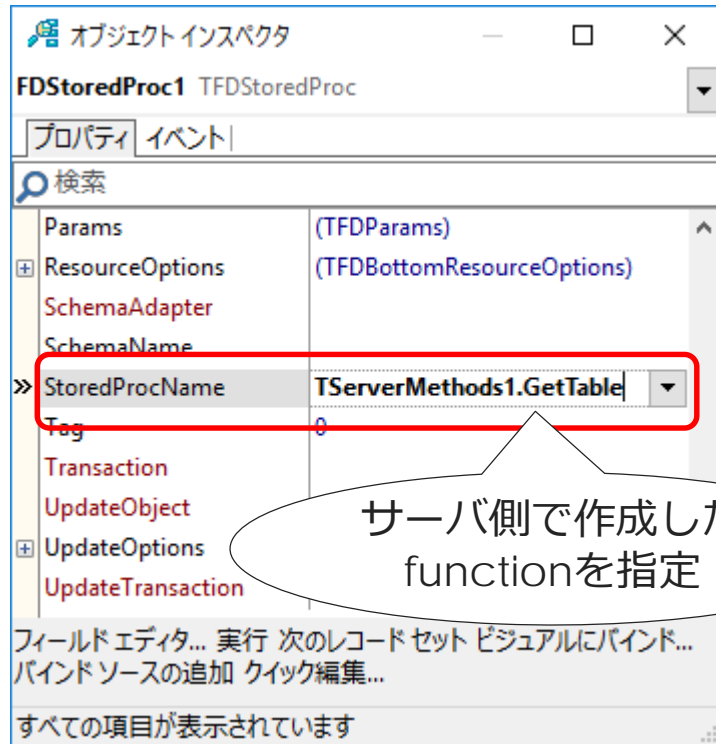
DatTableName: FDTTable1

SchemaAdapter: FDSchemaAdapter1

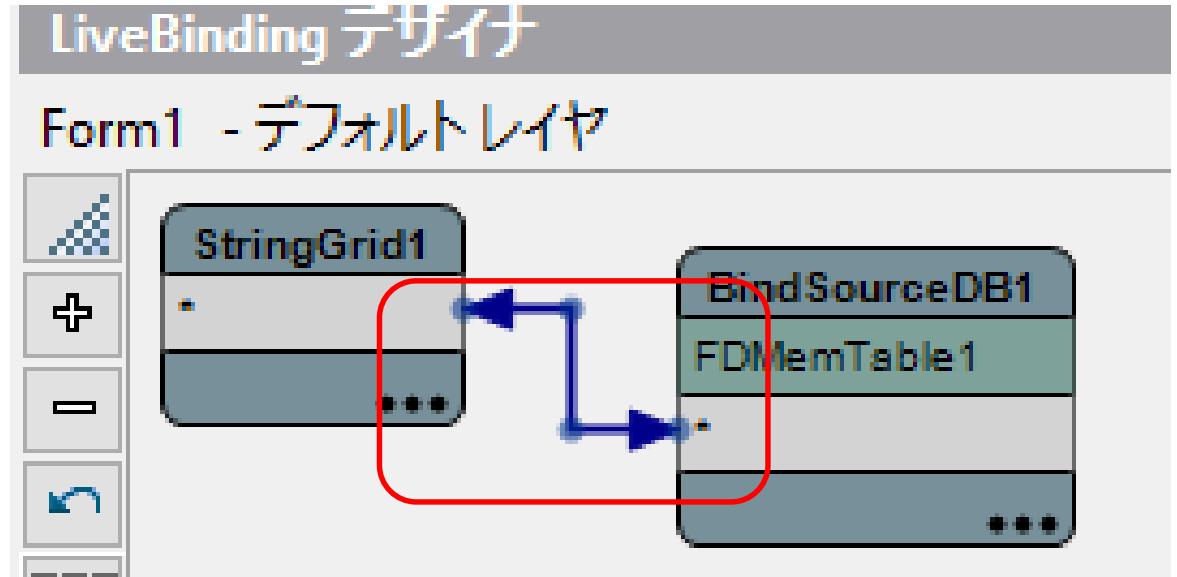
# DataSnap Serverを使った実装手順（FireDAC）

## ■ 実装手順10 コンポーネントの設定

TFDStoredProc



ライブバインディング設定



# DataSnap Serverを使った実装手順（FireDAC）

## ■ 実装手順11

データ取得の機能を実装

```
procedure TForm1.Button1Click(Sender: TObject);
var
  LStringStream: TStringStream;
begin
  //TFDStoredProcでサーバで作成したfunctionをCallしてストリームを受け取る
  FDStoredProc1.ExecProc;
  LStringStream := TStringStream.Create(FDStoredProc1.Params[0].asBlob);
  try
    if LStringStream <> nil then
      begin
        LStringStream.Position := 0;
        FDSchemaAdapter1.LoadFromStream(LStringStream, TFDStorageFormat.sfBinary);
      end;
    finally
      LStringStream.Free;
    end;
  end;
end;
```

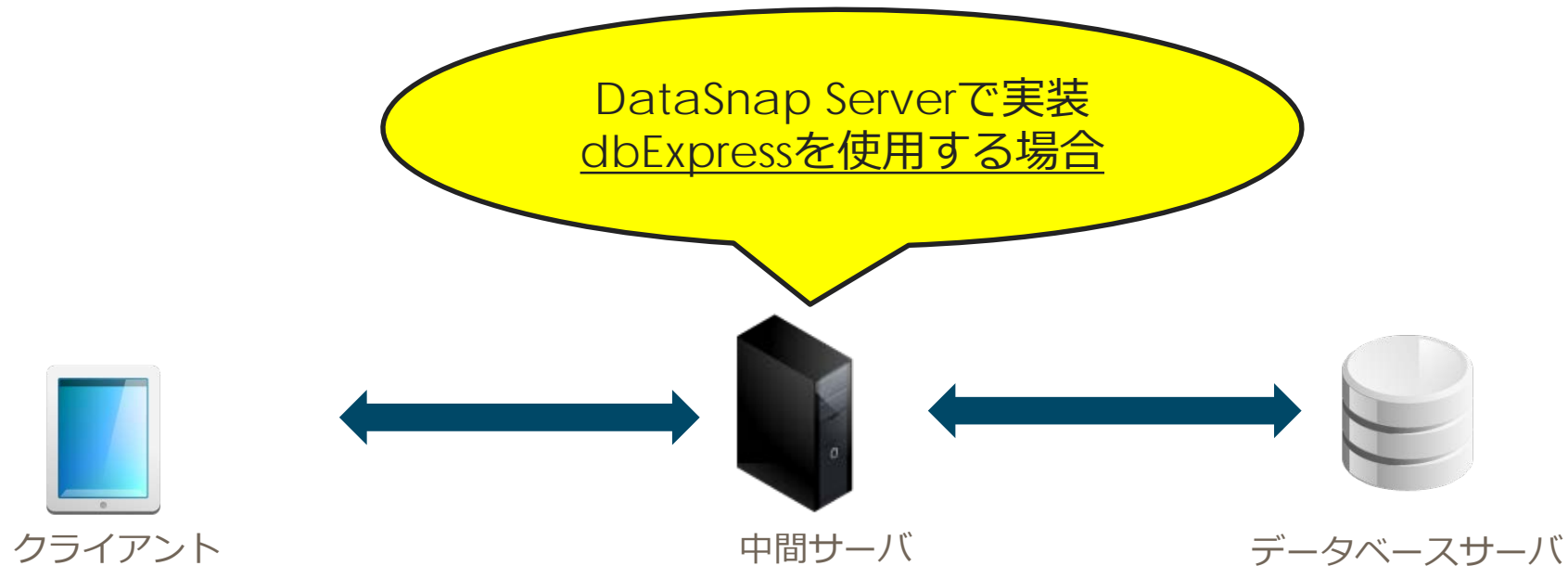


# DataSnap Serverを使った実装手順（FireDAC）

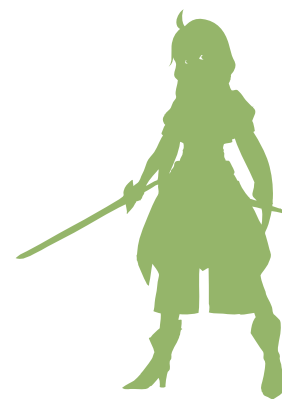
## ■ 実行



# DataSnap ServerでdbExpressを使う実装方法は、 P.77以降の補足資料を参考ください。

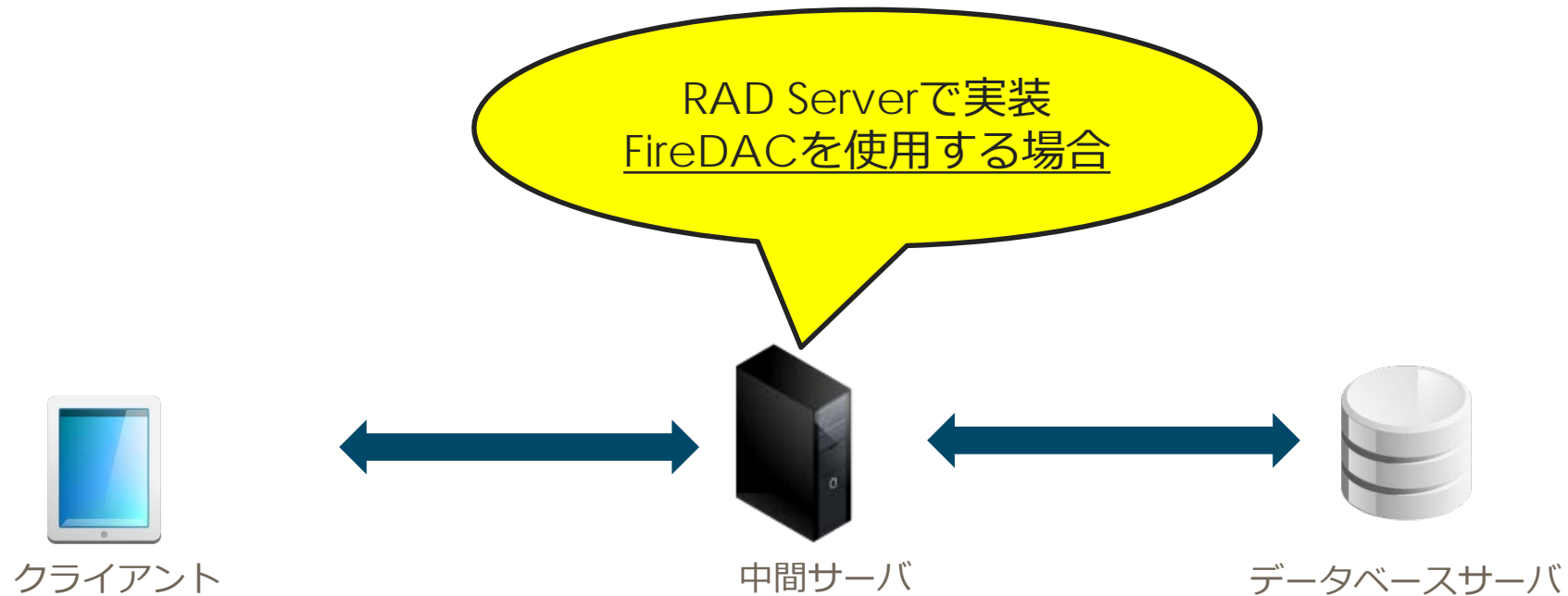


# RAD Serverを使った実装手順



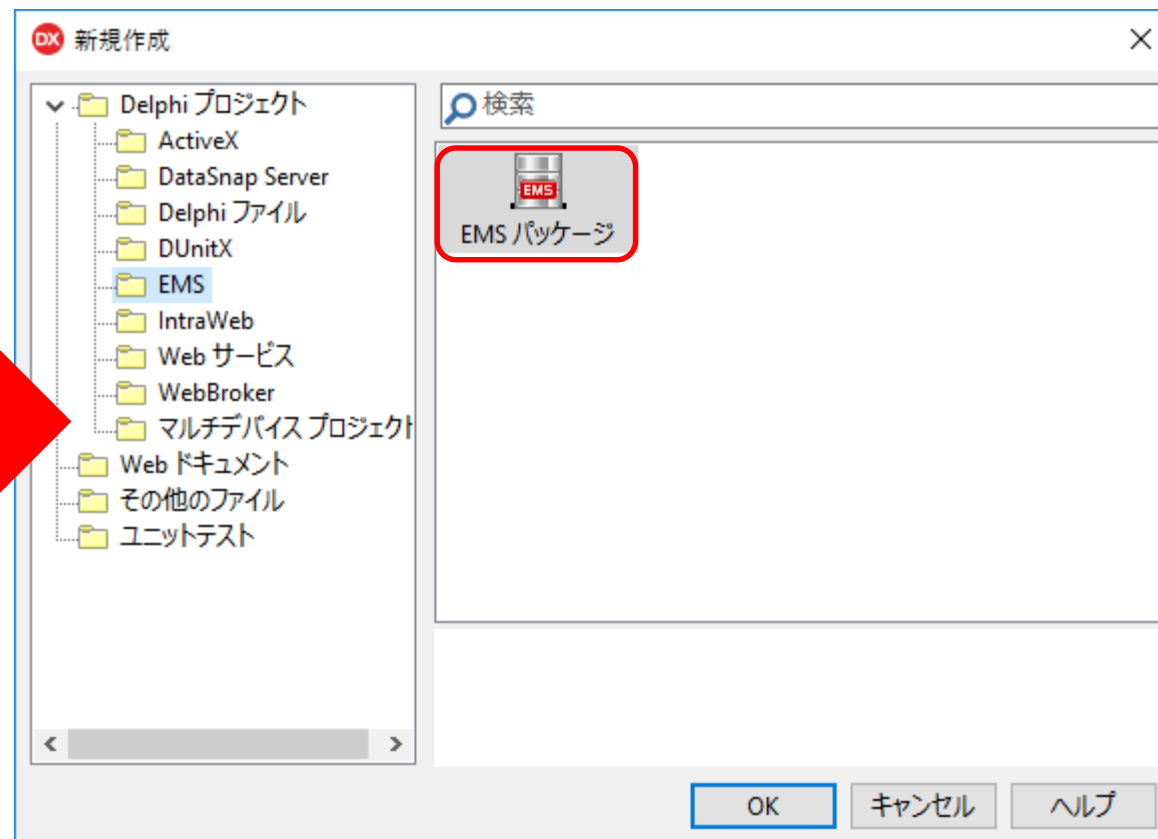
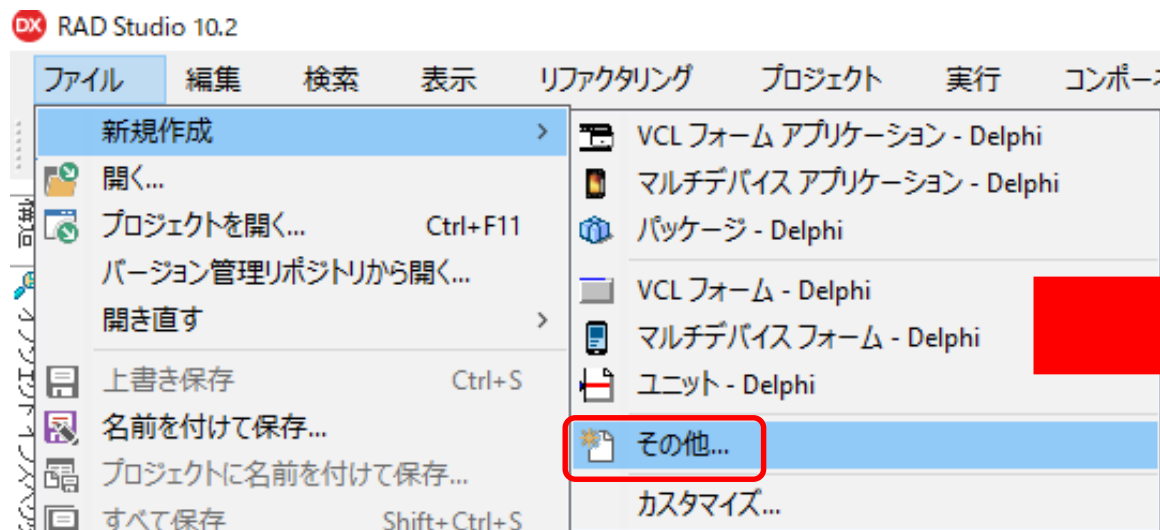
**e**mbarcadero®  
DEVELOPER CAMP

# RAD Serverを使った実装手順（FireDAC）



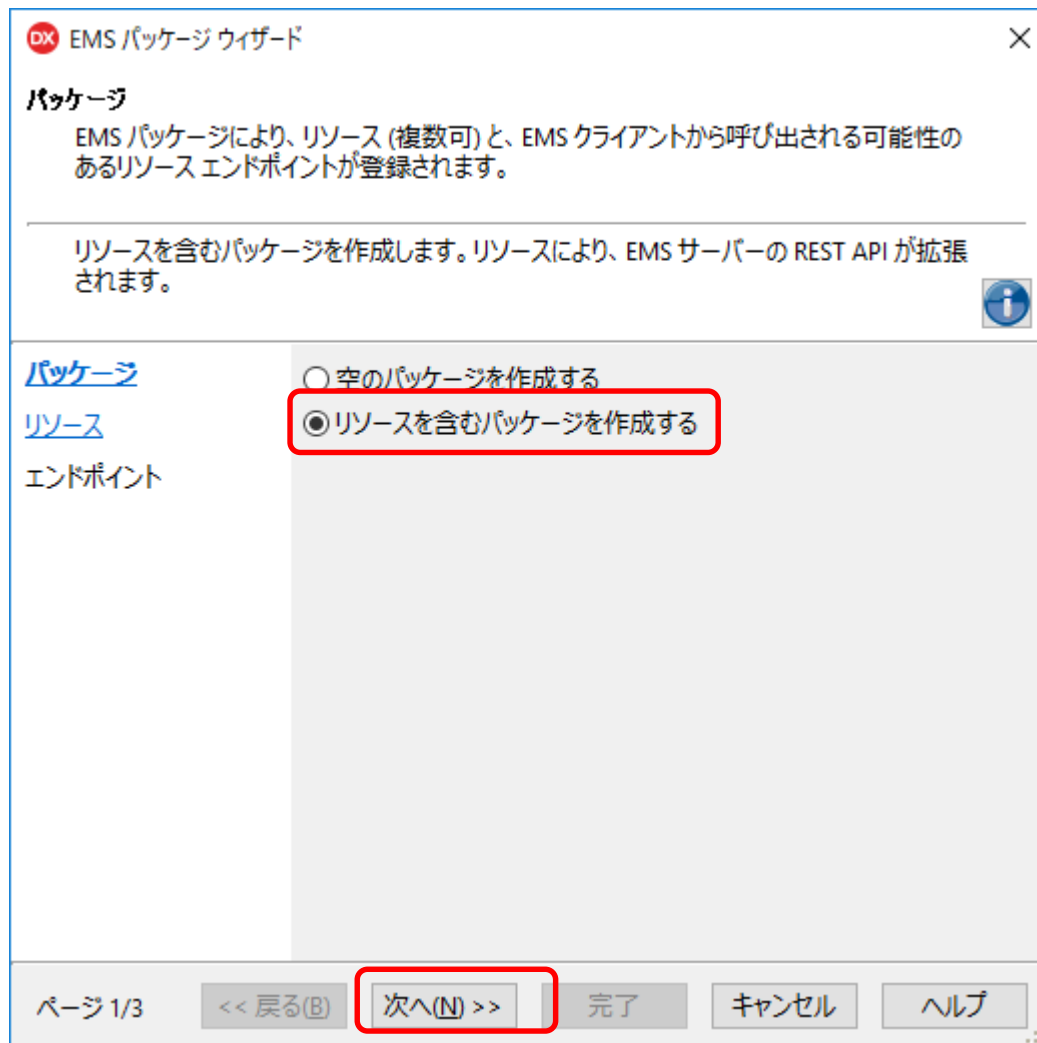
# RAD Serverを使った実装手順（FireDAC）

## ■ 実装手順1 プロジェクトの作成



# RAD Serverを使った実装手順（FireDAC）

## ■ 実装手順2 ウィザードの指定



# RAD Serverを使った実装手順（FireDAC）

## ■ 実装手順3 ウィザードの指定

DX EMS パッケージ ウィザード

リソース  
リソースを追加すると、EMS サーバーの REST API を拡張できます。

リソース データ モジュールを作成します。リソース データ モジュールは、デザインとコンポーネントを使ってリソースを実装するためのものです。

パッケージ  
リソース  
エンドポイント

リソース名(R): CUST

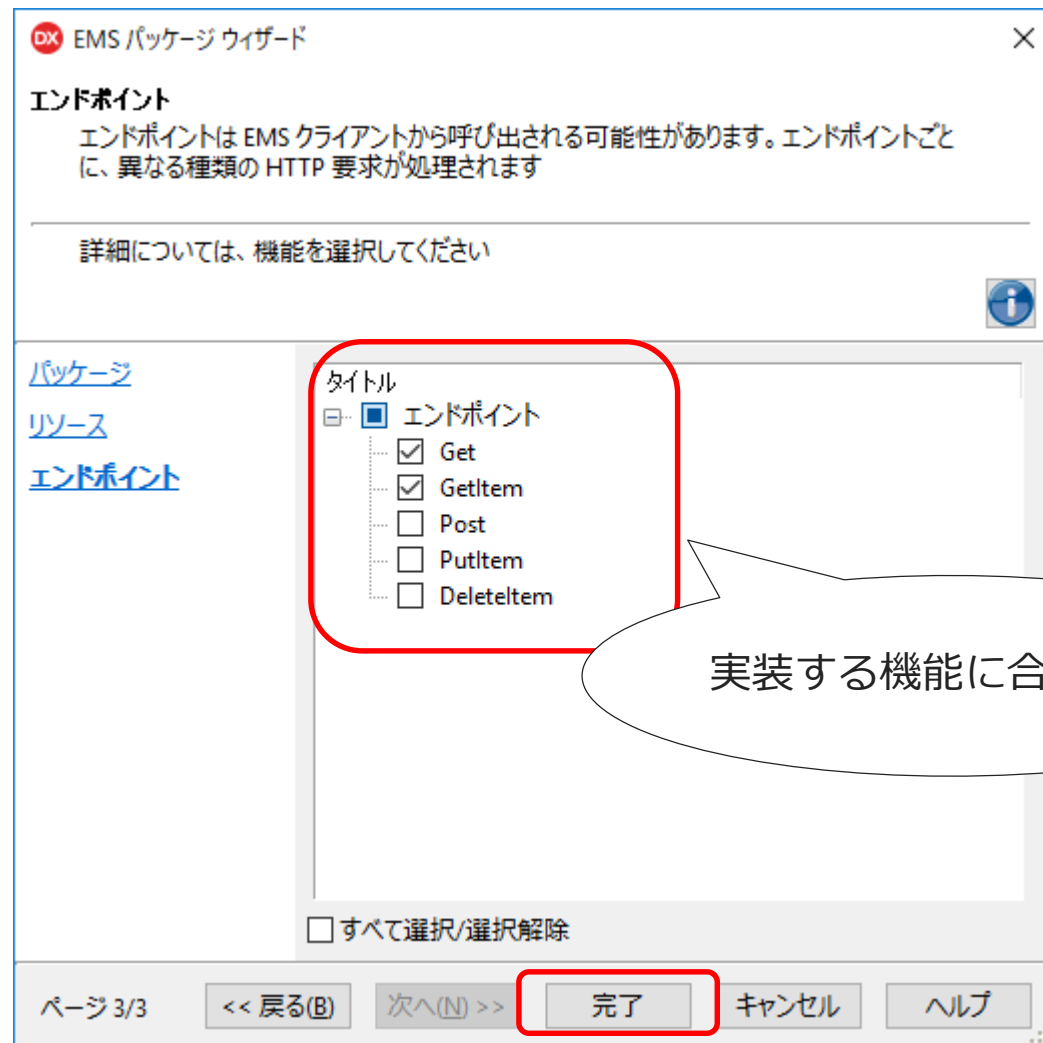
ファイルの種類(T):  データ モジュール  
 ユニット

ページ 2/3 << 戻る(B) 次へ(N) >> 完了 キャンセル ヘルプ

Restサービスとして呼び出す際は、このリソース名を使用

# RAD Serverを使った実装手順（FireDAC）

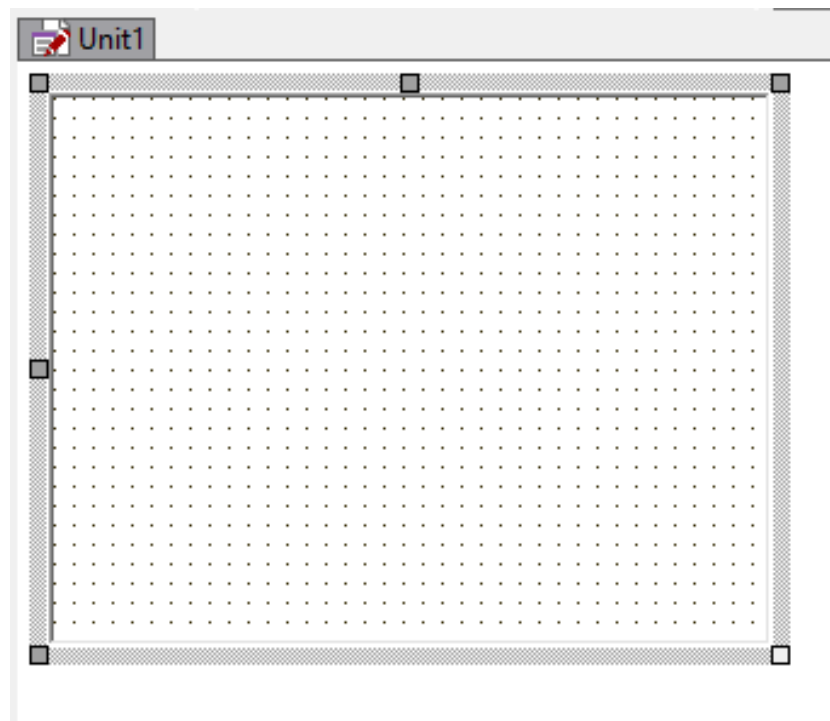
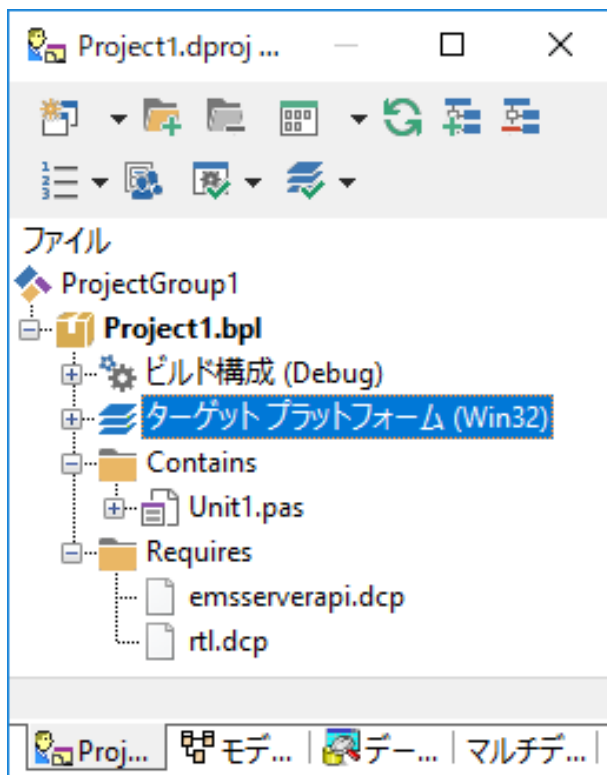
## ■ 実装手順4 ウィザードの指定





# RAD Serverを使った実装手順（FireDAC）

- 実装手順5  
自動生成されるユニット

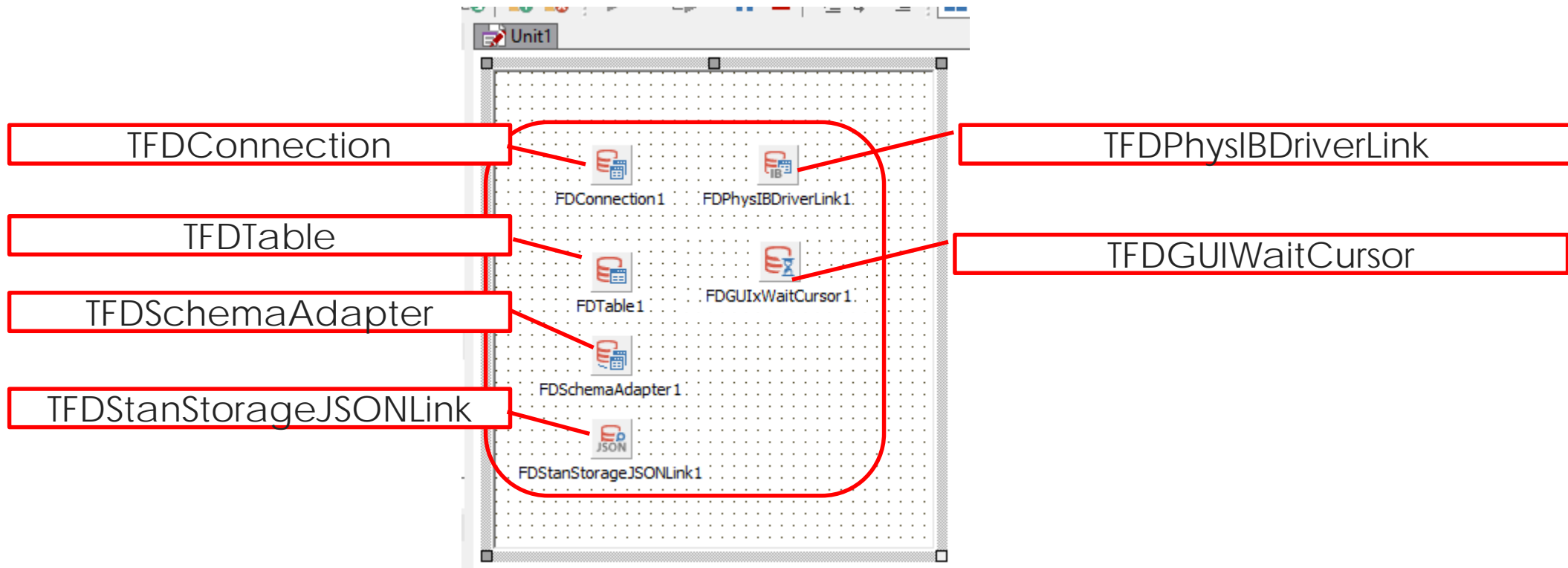


# RAD Serverを使った実装手順（FireDAC）

## ■ 実装手順6

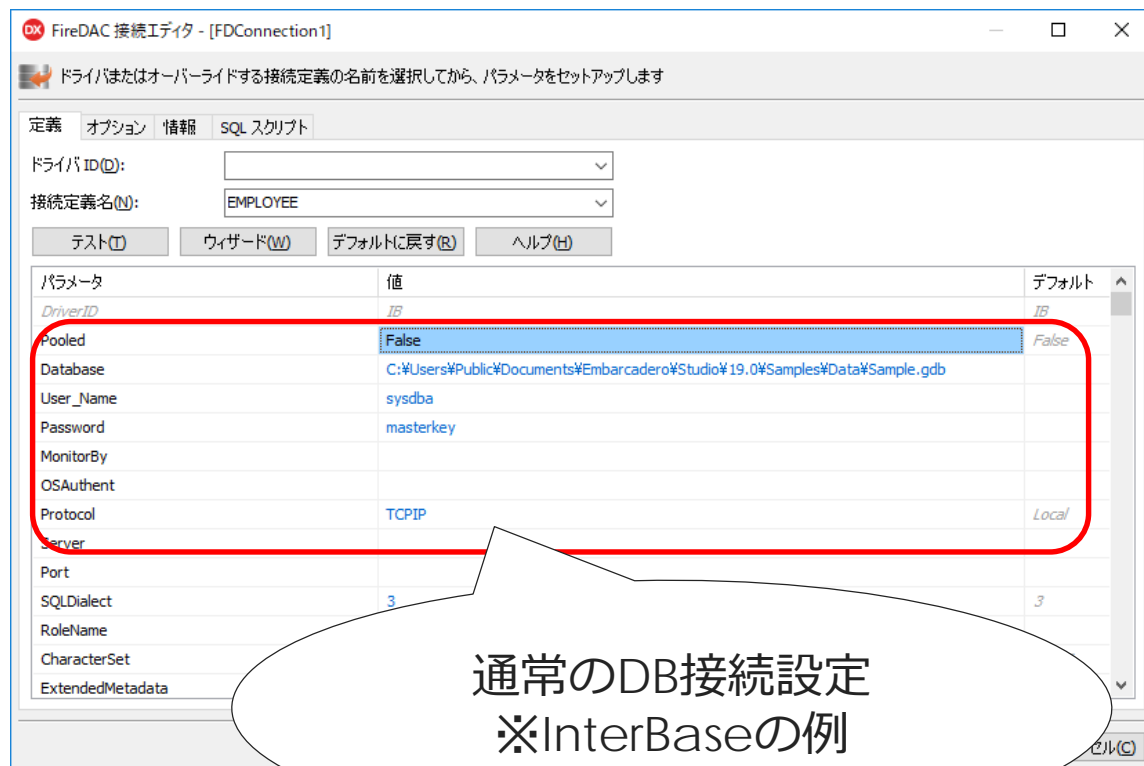
サーバ機能のコンポーネント配置

リソースとして機能を実装

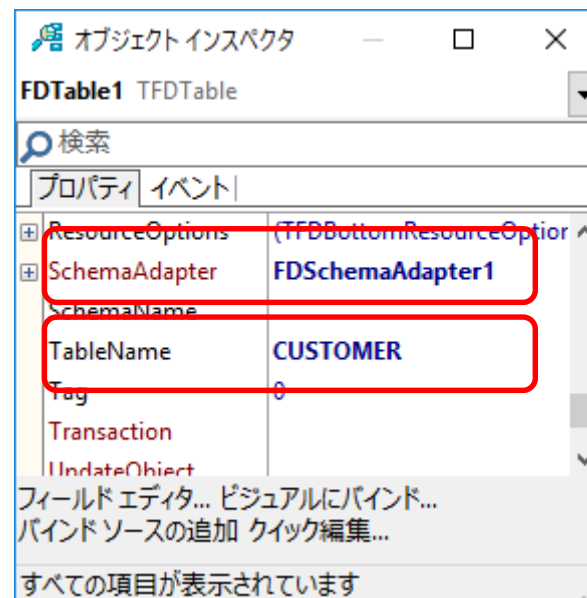


# RAD Serverを使った実装手順 ( FireDAC )

- 実装手順7  
コンポーネントの設定  
TFDConnection



## TFDTable



# RAD Serverを使った実装手順（FireDAC）

## ■ 実装手順8

データ取得の機能を実装

```
procedure TCUSTResource1.Get(const AContext: TEndpointContext; const ARequest:
TEndpointRequest; const AResponse: TEndpointResponse);
var
  oStr: TMemoryStream;
begin
  oStr := TMemoryStream.Create;

  // クエリの実行結果をスキーマアダプタからメモリストリーム経由で返す
  FDTTable1.Open;
  FDSchemaAdapter1.SaveToStream(oStr, TFDStorageFormat.sfJSON);
  AResponse.Body.SetStream(oStr, 'application/json', True);
end;
```

Restでパラメータを受け取ることも可能  
ARequest.Params.Values['XXXX']  
※QueryのSQLでバインド変数として  
ParamBy値で利用すると便利

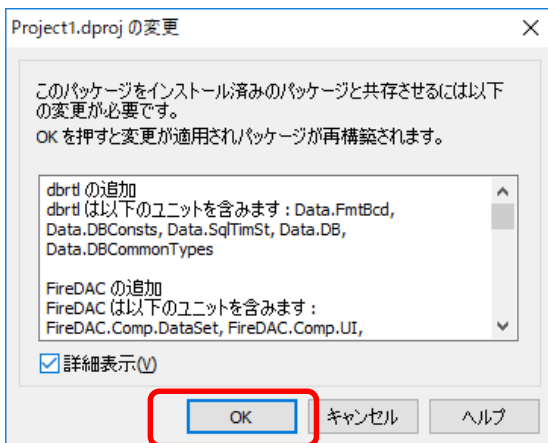
# RAD Serverを使った実装手順（FireDAC）

## ■ 実装手順9

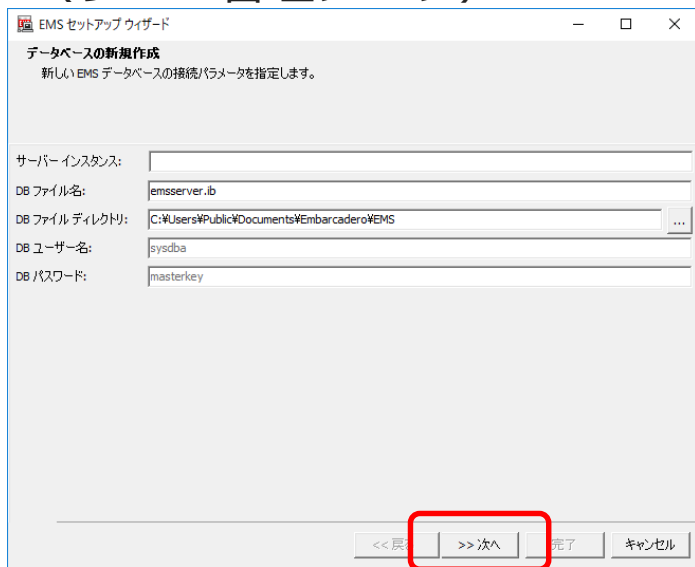
コンパイル・実行

初回コンパイルや実行時はinterBaseの設定も含め、ダイアログがいくつか出るので応答が必要。

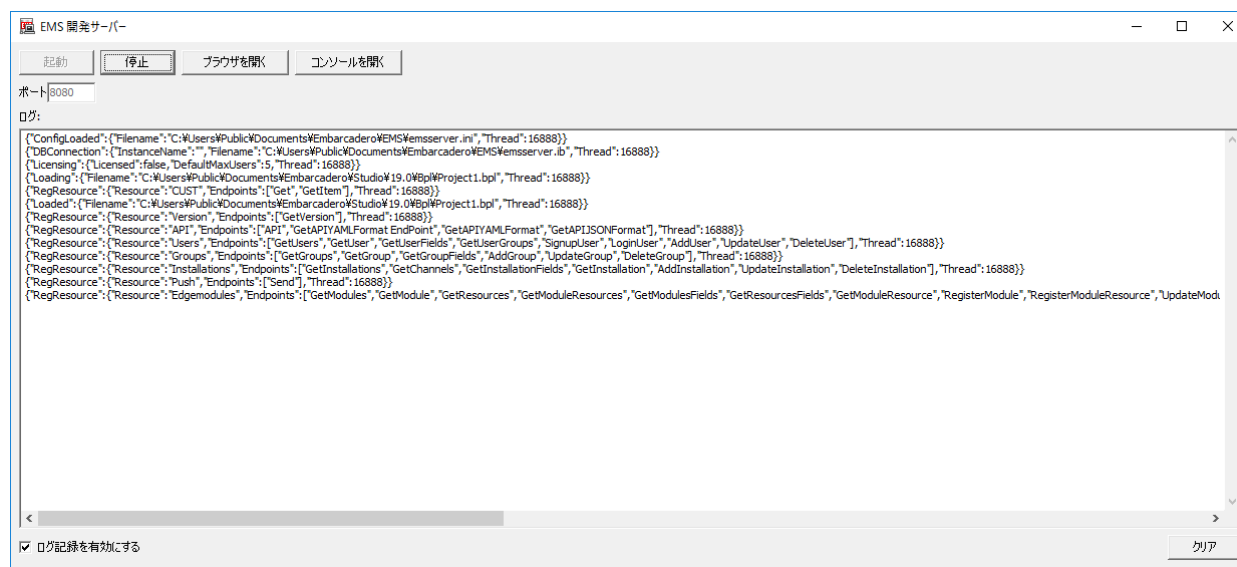
### パッケージの参照



### InterBaseの設定 (サーバ管理データ)



### 起動画面



# RAD Serverを使った実装手順（FireDAC）

RAD Serverで実装  
FireDACを使用する場合



クライアント



中間サーバ

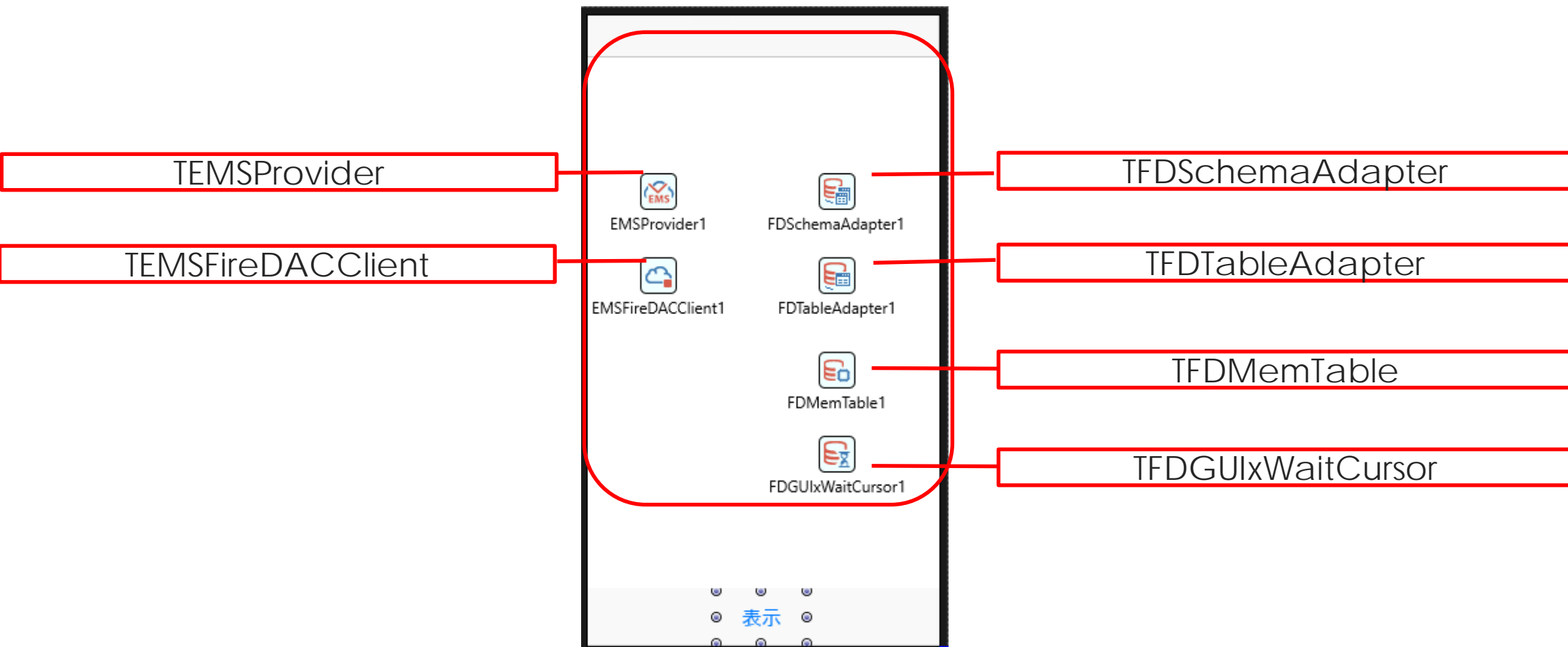


データベースサーバ

# RAD Serverを使った実装手順（FireDAC）

## ■ 実装手順10

クライアントアプリケーションのコンポーネント配置



# RAD Serverを使った実装手順（FireDAC）

- 実装手順11  
コンポーネントの設定

TEMSPProvider

オブジェクトインスペクタ

EMSPProvider1 TEMSPProvider

検索

プロパティ イベント

ProxyPort	0
ProxyServer	
TenantSecret	
URLBasePath	
URLHost	999.999.999.999
URLPort	8080
URLProtocol	http

接続テスト クイック編集...

すべての項目が表示されています

サーバIPやポートを設定。

TEMSPFireDACClient

オブジェクトインスペクタ

EMSPFireDACClient1 TEMSPFireDACClient

検索

プロパティ イベント

Auth	
LiveBinding デザイン	LiveBinding デザイン
Name	EMSPFireDACClient1
Provider	EMSPProvider1
Resource	CUST
SchemaAdapter	FDSchemaAdapter1
Tag	

クイック編集...

すべての項目が表示されています

接続したサーバで使用するリソースなどを指定。

TFDTableAdapter

オブジェクトインスペクタ

FDTableAdapter1 TFDTableAdapter

検索

プロパティ イベント

ColumnMappings	TFDDAptColumnMappings
DatTableName	FDTable1
DeleteCommand	
FetchRowCommand	
InsertCommand	
LiveBinding デザイン	LiveBinding デザイン
LockCommand	
MetaInfoMergeMode	mmReset
Name	FDTableAdapter1
SchemaAdapter	FDSchemaAdapter1

クイック編集...

すべての項目が表示されています

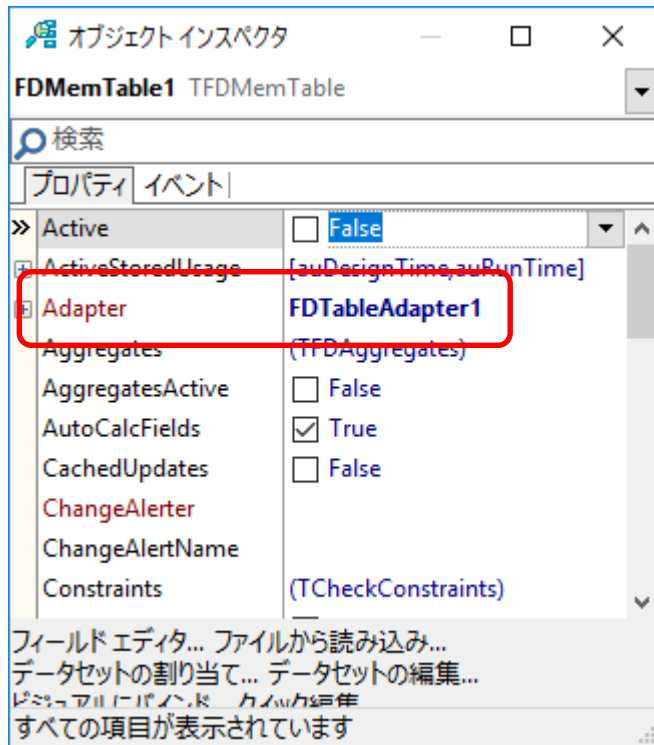
サーバ側のデータセット



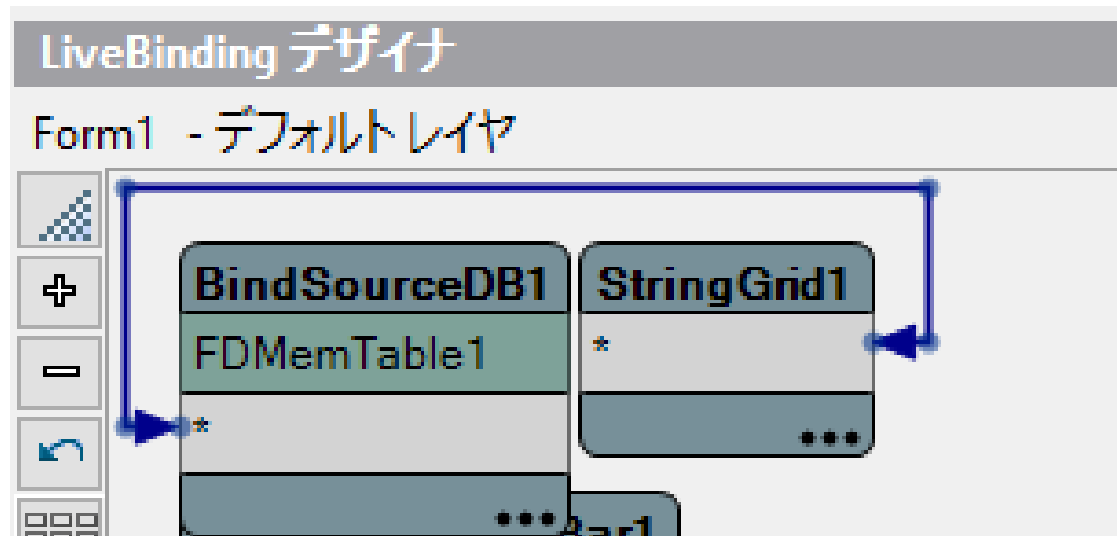
# RAD Serverを使った実装手順（FireDAC）

- 実装手順12  
コンポーネントの設定

TFDMemTable



ライブバインディング設定



# RAD Serverを使った実装手順（FireDAC）

- 実装手順13  
データ取得の機能を実装

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
    //設定しているリソースのGetDataを呼び出す  
    EMSFireDACClient1.GetData;  
end;
```

# RAD Serverを使った実装手順（FireDAC）

## ■ 実行



# RAD Serverのコンソール

## ■ コンソールのログイン

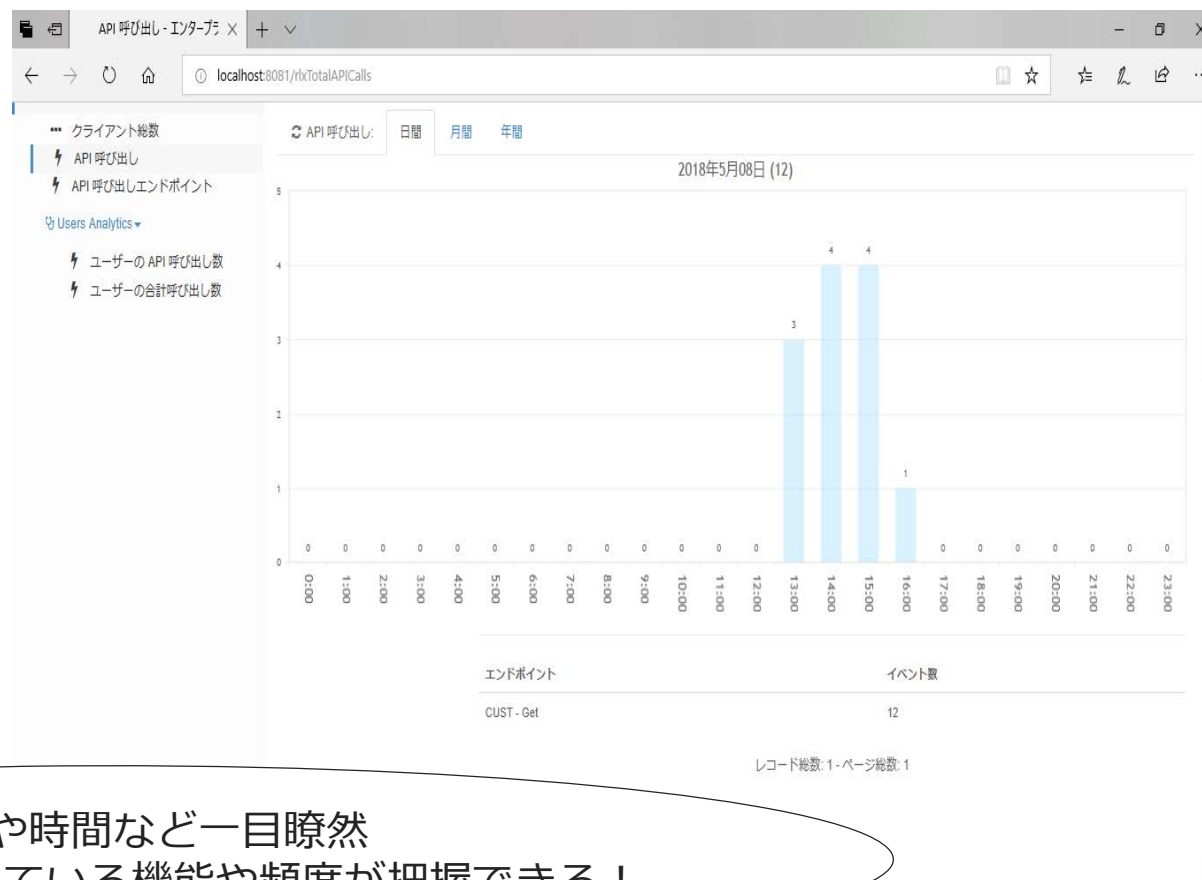
ログインアカウント情報はiniファイルに設定されています。

C:\Users\Public\Documents\Embarcadero\EMS\emsserver.ini



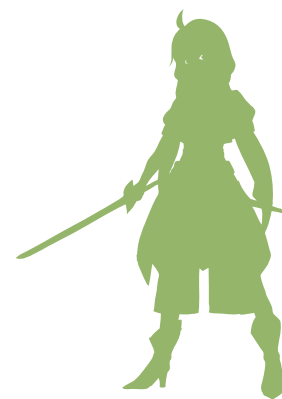
# RAD Serverのコンソール

- コンソールで使える分析メニュー（アナリティクス）



APIが利用された回数や時間など一目瞭然  
ユーザーが実際に使っている機能や頻度が把握できる！

# 中間サーバの活用範囲を拡張



**e**mbarcadero®  
DEVELOPER CAMP

# Enterprise Connectorsを利用したクラウドサービス等の連携

## ■ Enterprise Connectorsとは？

DelphiからSalesforceやAWS、SAP、ERP、Office 365、Googleドキュメント、ビッグデータDB、決済、ソーシャルサービスなど、80種類を超えるクラウドデータやエンタープライズサービスへのアクセス機能を追加できるパッケージソリューション



## 従来のDBアプリ開発スキルで簡単に実装が可能

Delphiに標準搭載されたDBエンジン「FireDAC」に対応しており、各サービスを利用するためのAPIは、コンポーネントによって提供される。

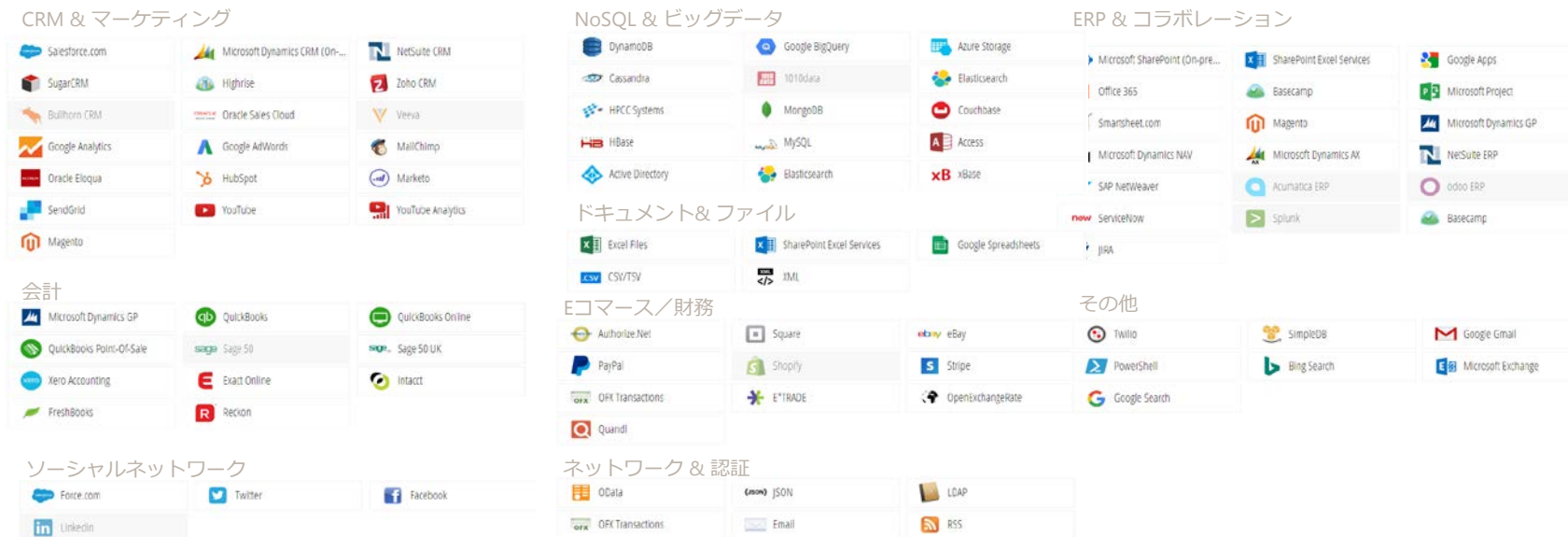
開発者はこれまでのデータベースアプリケーション開発スキルでクラウドデータ等を簡単に扱うことが可能。

# Enterprise Connectorsを利用したクラウドサービス等の連携

## ■ Enterprise Connectorsを中間サーバで利用するメリット

- ・ 社内DBだけではなく、クラウドサービスやパッケージのデータが連携できる
- ・ FireDACのコンポーネントで開発できるので、習得や調査が不要
- ・ クラウドサービスやパッケージのAPI変更にプログラム対応が不要
- ・ 中間サーバに実装すれば各クライアントでの導入や設定が不要

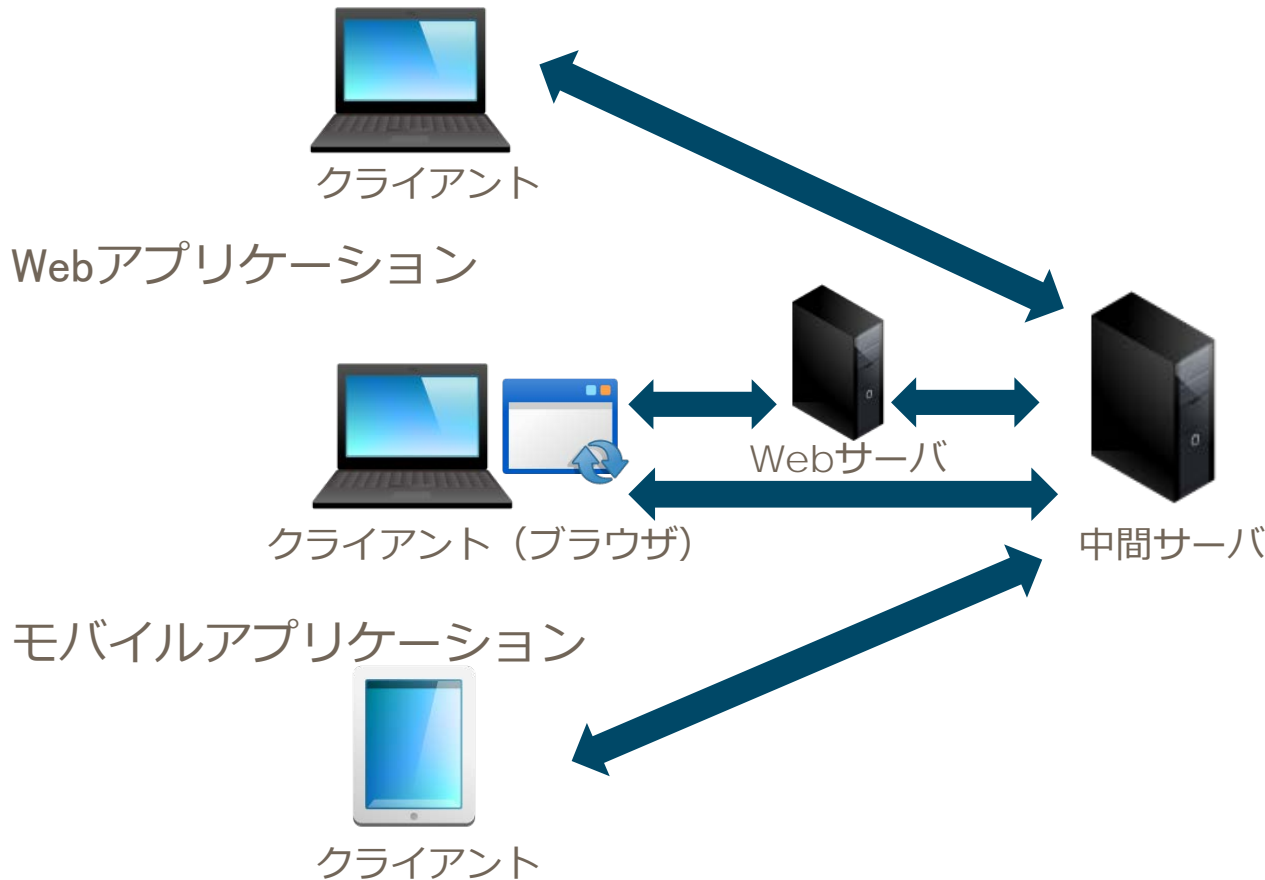
開発者の負担となる  
工数・コストが大幅軽減して  
アプリの拡張ができる





# Enterprise Connectorsを利用したクラウドサービス等の連携

【n層アプリケーション】  
Client/Serverアプリケーション



Salesforce & Force.com	Microsoft Dynamics CRM	Microsoft Dynamics GP
Microsoft Dynamics NAV	Microsoft Dynamics AX	Microsoft SharePoint
NetSuite CRM & ERP	QuickBooks Desktop	QuickBooks Online
QuickBooks Point of Sale	Sage US	Sage 50 UK
Xero Accounting	Exact Online	Intacct
FreshBooks Accounting	SAP NetWeaver	OData Services
JSON Services	Microsoft Excel	SharePoint Excel Services
Twitter	Facebook	LDAP Directory Services
Amazon DynamoDB	Amazon SimpleDB	Google Analytics
Google Apps	Google Apps	Google Sheets
Office 365	Gmail	OFX Financial Accounts
PowerShell	Email	RSS Feeds
MongoDB	Google BigQuery	Azure Table
Apache Cassandra	Couchbase Server	Apache HBase



# Enterprise Connectorsの組み込み例 (Salesforce)

- 組み込み手順1 (※製品購入版はCData社サイトよりインストーラをDLしてインストールします)  
[ツール|GetItパッケージマネージャ]から対象のECを検索して選択

The screenshot shows the 'GetIt パッケージマネージャ' (GetIt Package Manager) window. The search bar contains 'salesforce'. Three connector packages are listed: 'Beta - Salesforce Einstein...', 'Trial - Salesforce Components', and 'Trial - Salesforce Chatter...'. The 'Trial - Salesforce Components' package is highlighted with a red box. A red arrow points from this package to a '依存先コンポーネントのライセンス' (Dependency Component License) dialog box. The dialog box displays the license agreement for 'Trial - Salesforce Components'. The license agreement text is as follows:

**説明**  
Enables you to use standard drivers to connect to Salesforce objects. Just specify your credentials and access tables like Leads, Contacts, etc. For technical questions, contact support@cdata.com.

**ライセンス**  
SOFTWARE LICENSE AGREEMENT

THIS SOFTWARE LICENSE AGREEMENT IS A LEGAL AGREEMENT BETWEEN YOU, EITHER A SINGLE INDIVIDUAL, ENTITY OR GOVERNMENT ORGANIZATION AND EMBARCADERO TECHNOLOGIES, INC. AND ITS AFFILIATES FOR THE SOFTWARE YOU ARE LICENSING. CAREFULLY READ THIS AGREEMENT BEFORE YOU INSTALL OR USE THE SOFTWARE. BY INSTALLING OR USING THE SOFTWARE OR BY CLICKING ON "ACCEPT" YOU AGREE TO BE BOUND BY THE TERMS OF THIS AGREEMENT AND YOU REPRESENT THAT YOU HAVE THE AUTHORITY TO ENTER INTO THIS AGREEMENT. ALL SOFTWARE ORDERED THROUGH AN AUTHORIZED RESELLER OR DISTRIBUTOR IS GOVERNED BY THIS AGREEMENT. IF YOU DO NOT HAVE THE AUTHORITY TO ENTER INTO THIS AGREEMENT, OR IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, YOU MUST NOT INSTALL OR USE THE SOFTWARE. THIS AGREEMENT, SHALL GOVERN YOUR INSTALLATION AND USE OF THE PRODUCTS UPON THE EARLIER OF YOUR AGREEMENT TO PURCHASE A LICENSE FOR SUCH PRODUCTS OR YOUR INSTALLATION OR USE OF THE PRODUCTS.

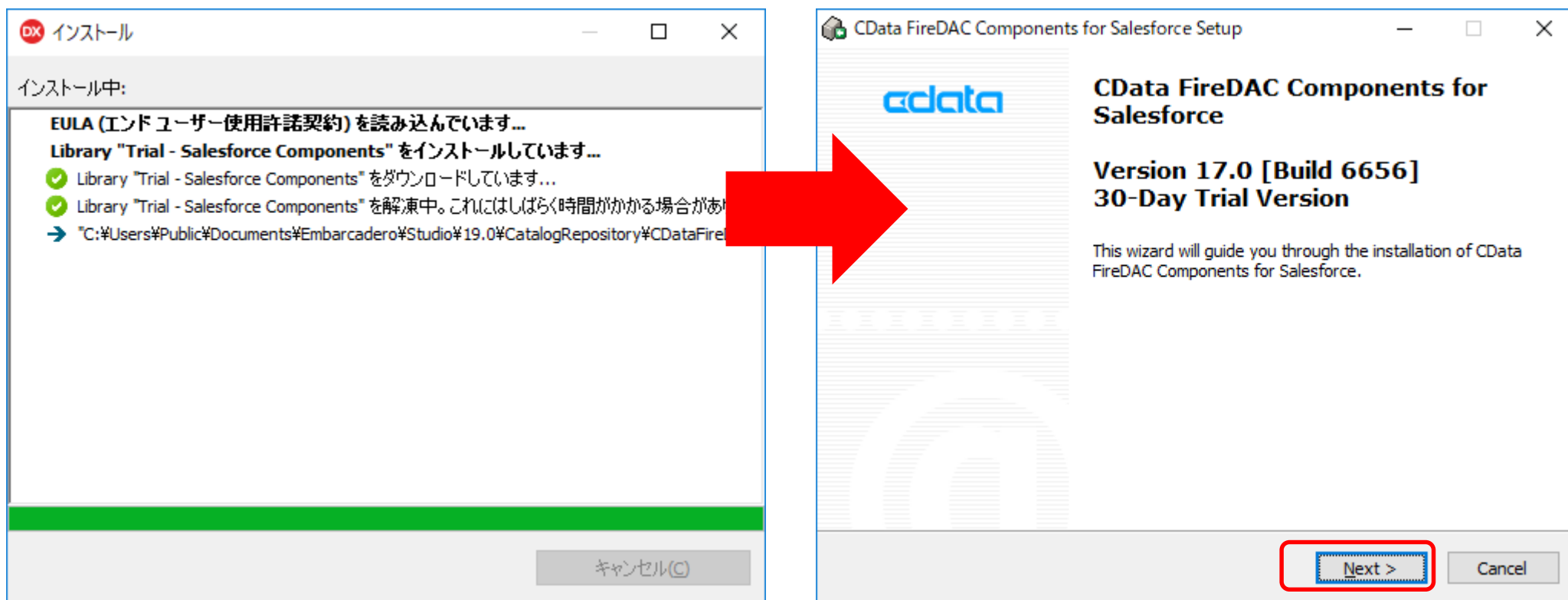
THIS SOFTWARE LICENSE AGREEMENT (this "Agreement"), dated as of the date of your purchase or receipt of a license to use the Software, is between Embarcadero Technologies, Inc., a Delaware corporation ("Licensor") and the customer set forth on

At the bottom of the dialog box, there are two buttons: 'すべて同意する(A)' (Accept All) and '同意しない(D)' (Do Not Accept). The 'すべて同意する(A)' button is highlighted with a red box.

パッケージはコネクタ毎 (サービス毎) にインストールが必要

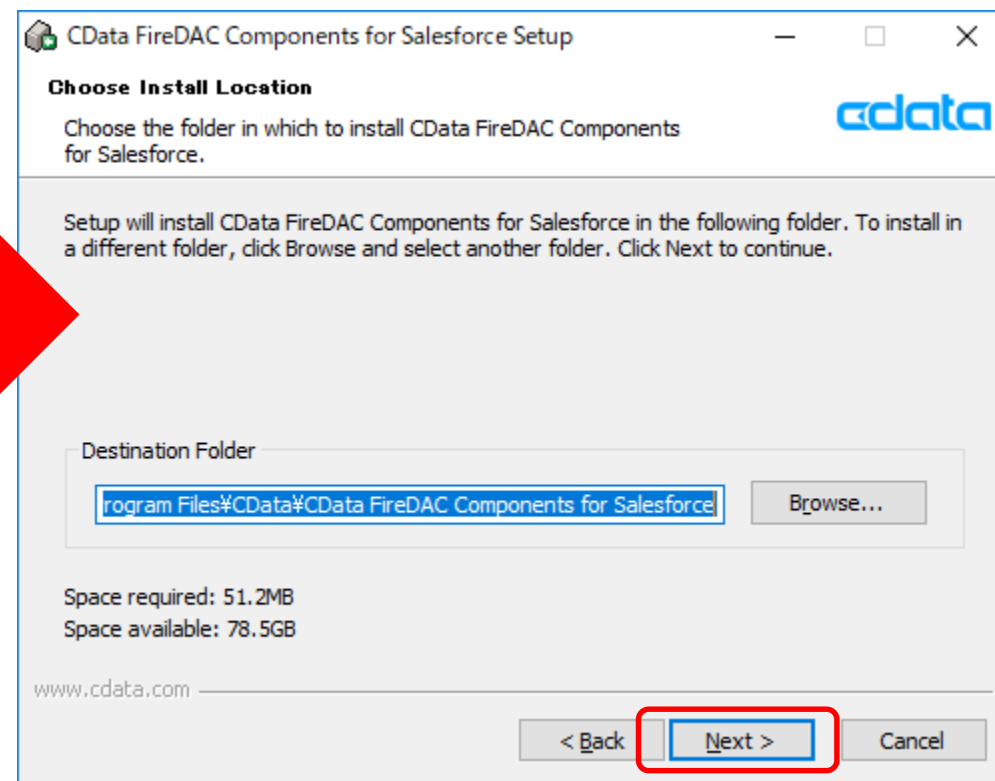
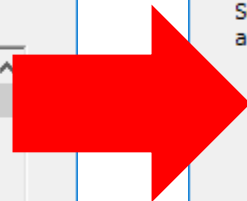
# Enterprise Connectorsの組み込み例 (Salesforce)

- 組み込み手順2  
自動ダウンロード・配置



# Enterprise Connectorsの組み込み例 (Salesforce)

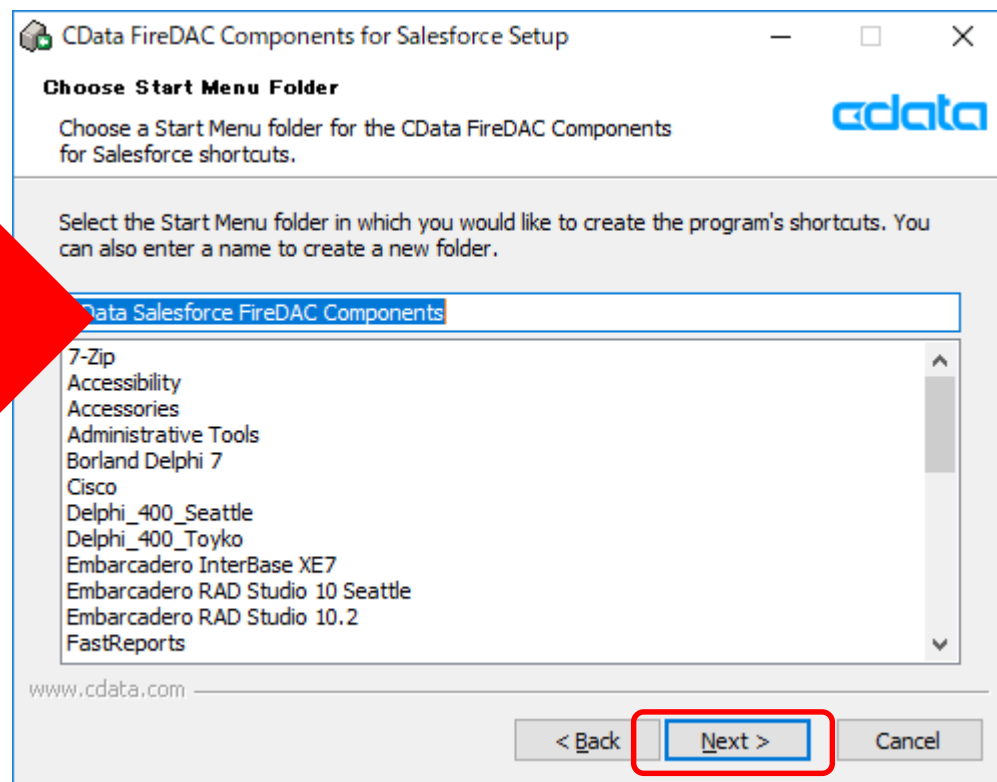
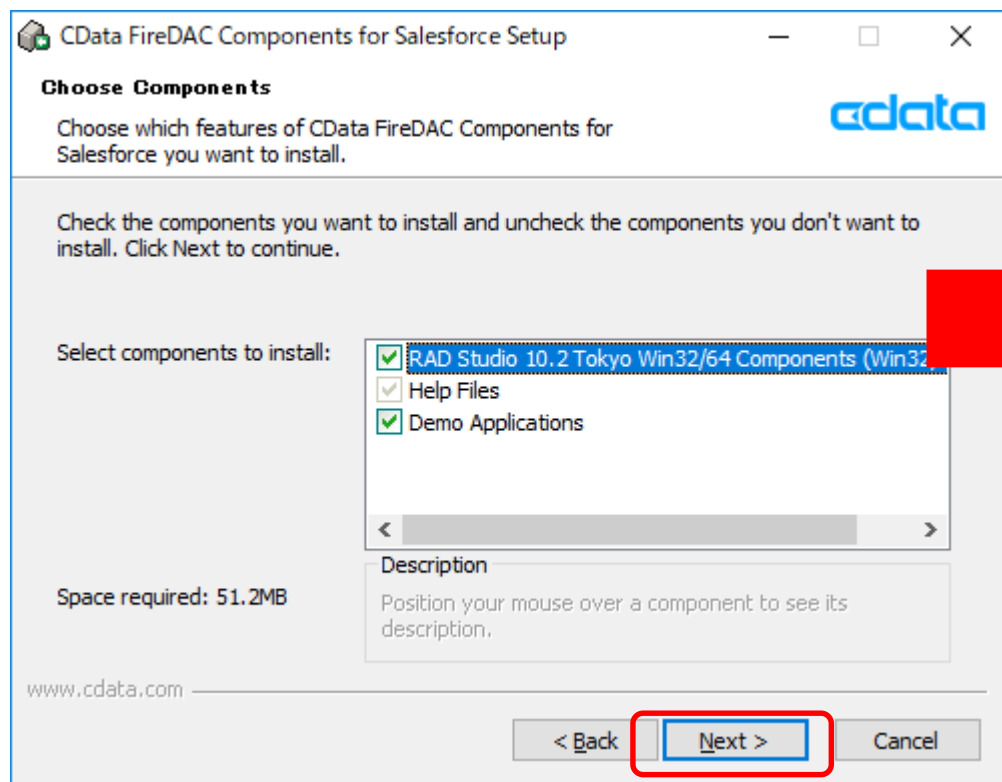
- 組み込み手順3  
インストール先を設定



# Enterprise Connectorsの組み込み例 (Salesforce)

## ■ 組み込み手順4

インストールオプションの選択



# Enterprise Connectorsの組み込み例 (Salesforce)

- 組み込み手順5  
導入ユーザーの登録

**Product Registration**  
Product registration is a requirement for support.

Name: Taisuke Yoshiwara  
Company:   
Title:   
Email: \*\*\*\*\*@\*\*\*\*.\*\*.\*\*\*

Phone Number:   
Address:   
City:  State:  Zip:   
Country:

< Back **Next >** Cancel

氏名とメールアドレスは必須

**Trial License Activation**

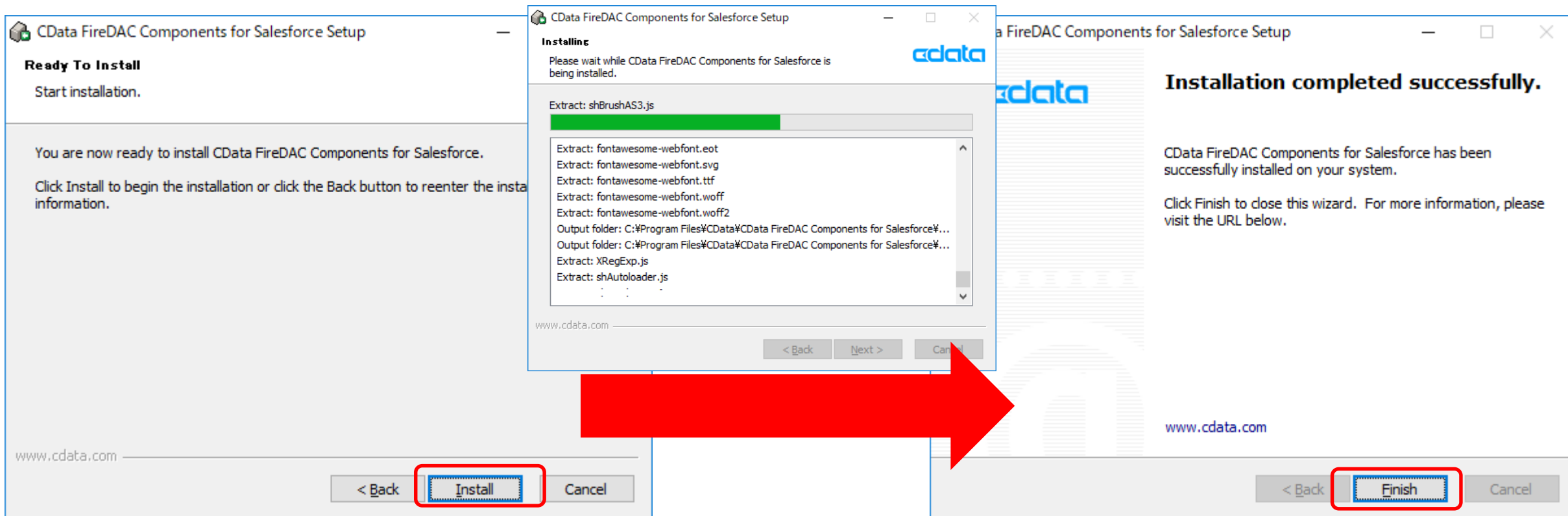
When you click 'Next', the setup will automatically activate a trial license.

www.cdata.com

< Back **Next >** Cancel

# Enterprise Connectorsの組み込み例 (Salesforce)

- 組み込み手順6  
インストールの実行



# Enterprise Connectorsの組み込み例 (Salesforce)

## ■ 組み込み手順7 ドライバの組み込み

対象サービスのFireDAC用  
ドライバとして導入され、使用できる

FDConnection1    FDPhysCDataSalesforceDriverLink1

定義   オプション   情報   SQL スクリプト

ドライバID(D):    CDataSalesforce  
接続定義名(N):    ADS  
                          ASA  
                          CDataSalesforce  
                          CO400  
                          DB2  
                          DS  
                          FB  
                          IB

テスト(T)    ウィ    CO400

パラメータ

DriverID

User

Password

通常のDBと同じように  
ドライバが選択できる



# Enterprise Connectorsの組み込み例 (Salesforce)

## ■ 組み込み手順8

通常のFireDACのDB操作と同じ

定義 オプション 情報 SQL スクリプト

ドライバID(D): CDataSalesforce

接続定義名(N):

テスト(T)

パラメータ

パラメータ	値
DriverID	CDataSalesforce
User	*****@*****.***
Password	*****
SecurityToken	

例) ログインアカウントは  
通常Salesforceへログインする  
アカウントを使用



ContractHistory  
ContractStatus  
CorsWhitelistEntry  
CronJobDetail  
CronTrigger  
CspTrustedSite  
CustomBrand  
CustomBrandAsset  
Customers\_Problem\_c  
CustomObject2\_c  
CustomObjectUserLicenseMetrics  
CustomPermission  
CustomPermissionDependency  
Dashboard  
DashboardComponent  
DashboardComponentFeed  
DashboardFeed  
DataAssessmentFieldMetric  
DataAssessmentMetric  
DataAssessmentValueMetric  
TableName: Salesforce.Account  
Tag: 0

SQLやテーブル名で、  
通常のDBと同じように  
アクセス・操作できる

# Enterprise Connectorsの組み込み例 (Salesforce)

## ■ 実行

実際のSalesforceデータ

ホーム Chatter ファイル 取引先 取引先責任者 ケース ソリューション レポート ダッシュボード

### 新規取引先一覧

レポート生成状況: 完了

オプション:

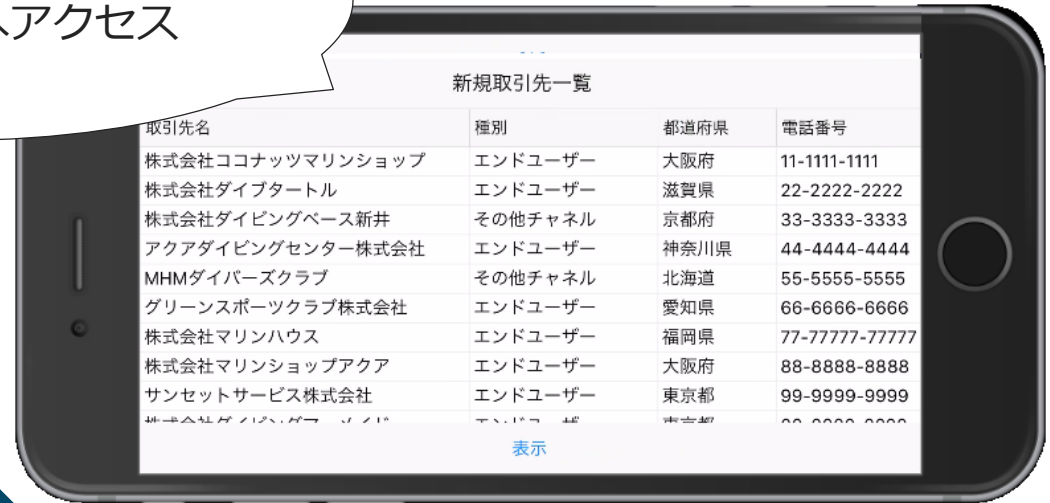
集計情報:  表示  期間条件:

レポート実行 詳細を非表示 カスタマイズ 保存 別名で保存 削除 印刷用に表示 詳細のエキスポート 登録

取引先所有者	取引先名	都道府県(請求先)	電話	種別	最終更新日
サポート	株式会社コナツツマリショップ	大阪府	11-1111-1111	エンドユーザー	2018/05/12
サポート	株式会社ダイブタイトル	滋賀県	22-2222-2222	エンドユーザー	2018/05/12
サポート	株式会社ダイビングベース新井	京都府	33-3333-3333	その他チャネル	2018/05/12
サポート	アクアダイビングセンター株式会社	神奈川県	44-4444-4444	エンドユーザー	2018/05/12
サポート	MHMダイバーズクラブ	北海道	55-5555-5555	その他チャネル	2018/05/12
サポート	グリーンスポーツクラブ株式会社	愛知県	66-6666-6666	エンドユーザー	2018/05/12
サポート	株式会社マリnhaus	福岡県	77-7777-7777	エンドユーザー	2018/05/12
サポート	株式会社マリショップアクア	大阪府	88-8888-8888	エンドユーザー	2018/05/12
サポート	サンセットサービス株式会社	東京都	99-9999-9999	エンドユーザー	2018/05/12
サポート	株式会社ダイビングマーマイド	東京都	00-0000-0000	エンドユーザー	2018/05/12

総計 (10件)

Delphiアプリでは自由にSalesforceへアクセス



社内DB

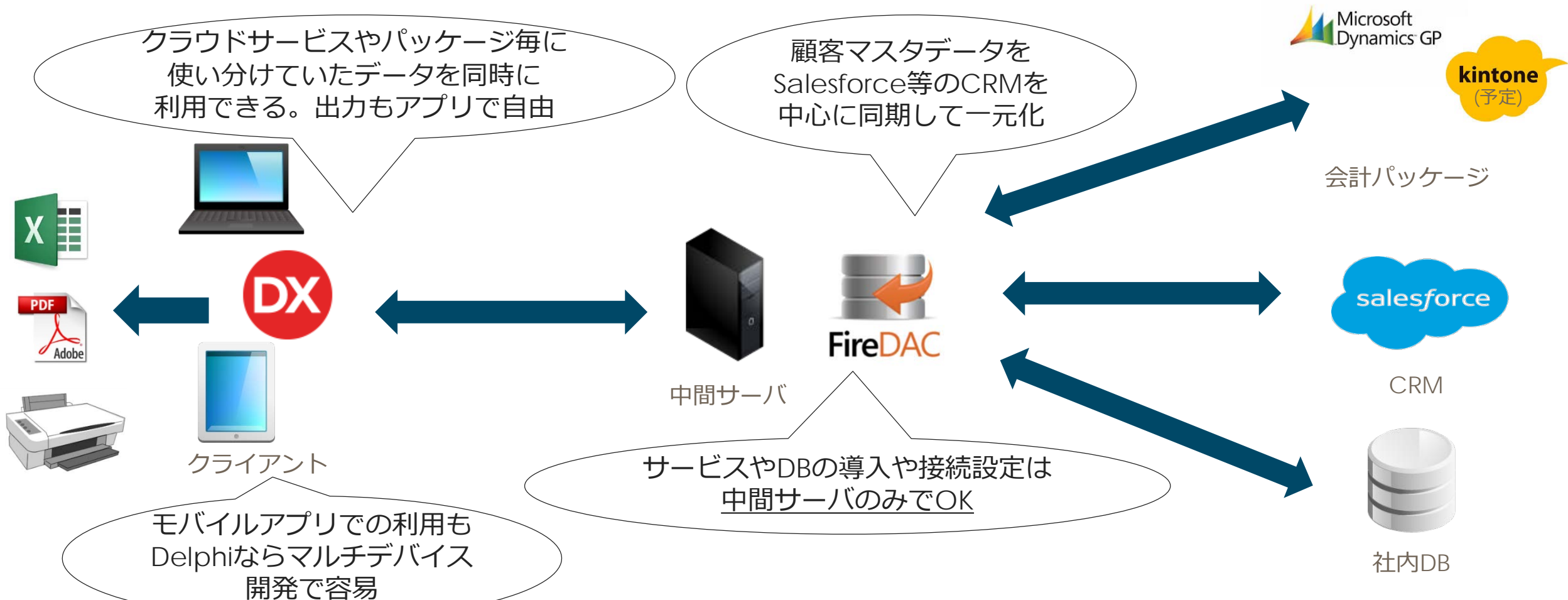
SELECT \* FROM CUSTOMER

CUSTNO	COMPANY	ADDR1	ADDR2	CITY	STA
1221	コナツツマリショップ	大島町4-976-321		練馬区	東京
1513	ダイブタイトル	東荻5-8-7		杉並区	東京
3444	ダイビングベース新井	新井2-14-3	新井2-16-13	石垣市	沖縄
1231	アクアダイビングセンター	明太区豊根541		北九州市	福岡
1351	亀山ダイブセンター	稲毛区亀山町632-1	稲毛区鶴亀2-4-22	千葉市	千葉
1380	ダイブショップブルーリーフ	鯉松町23-738	鎌3-15-23	港区	東京
1384	MHMダイバーズクラブ	塩崎町32		松戸市	千葉
1994	ADVENTURE UNDERSEA	PO BOX 64594		GUAM	
2118	グリーンスポーツクラブ	中海妻町622	東進辺町3-147	清水市	静岡
2135	バイナブルダイバーズ			石垣市	沖縄
2156	マリnhaus			青森市	青森
2163					
2169					
2174					
2319					

社内DBへの同期や連携にも使える

# Enterprise Connectorsを利用した拡張

## ■ クラウドサービスやパッケージとの連携・同期例



# Enterprise Connectors利用時の補足

## ■ 補足：通常のRDBMSと異なる点

- ビューやプロシージャなど、用意されているものしか基本使えない。  
（自由に作成はできない）
- トランザクションは基本使えない。
- ネットワークやサービス側の負荷が応答パフォーマンスに影響する。
- 必要なデータだけを絞り込む(select \* 的な検索は行わない)

Enterprise Connectors の select 実行時に where で使う検索条件は、接続先サービスの検索仕様で指定する

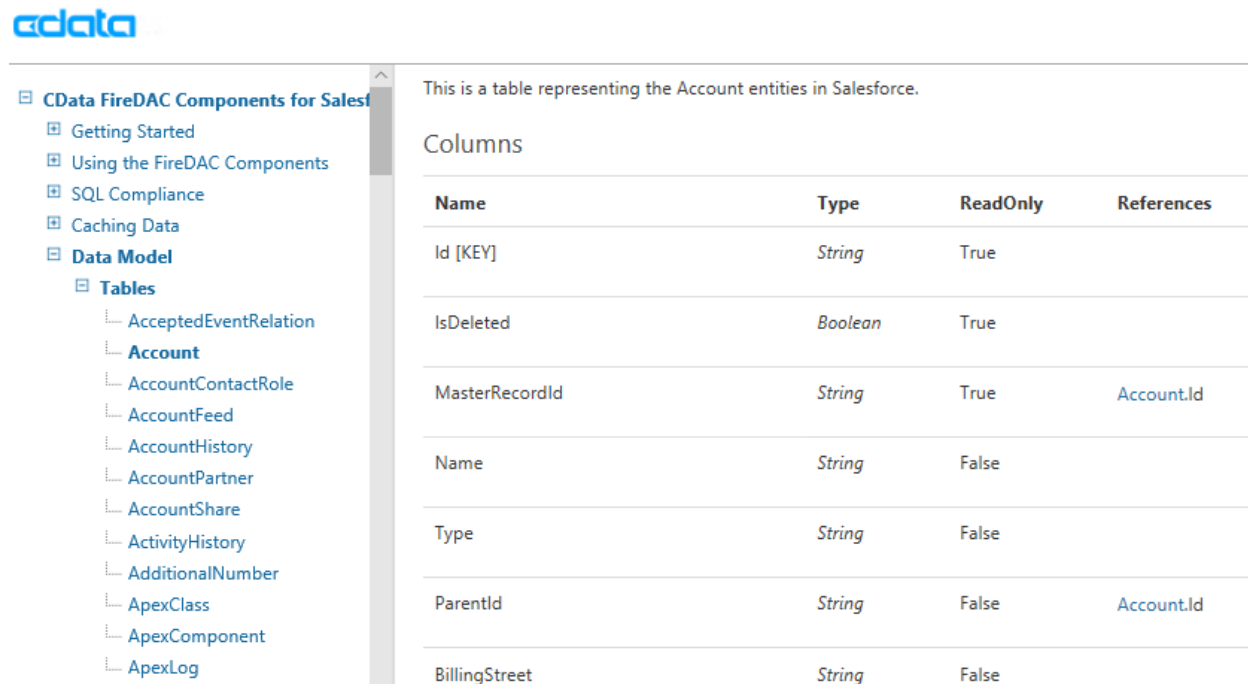
# Enterprise Connectors利用時の補足

## ■ 補足：テーブル等の定義

アクセスできるテーブル（の形で用意された）定義等はローカルのヘルプに詳しく掲載されている。

定義だけで分からない不明なものは実データを見て判断する方法も有効。

### ローカルのHelp



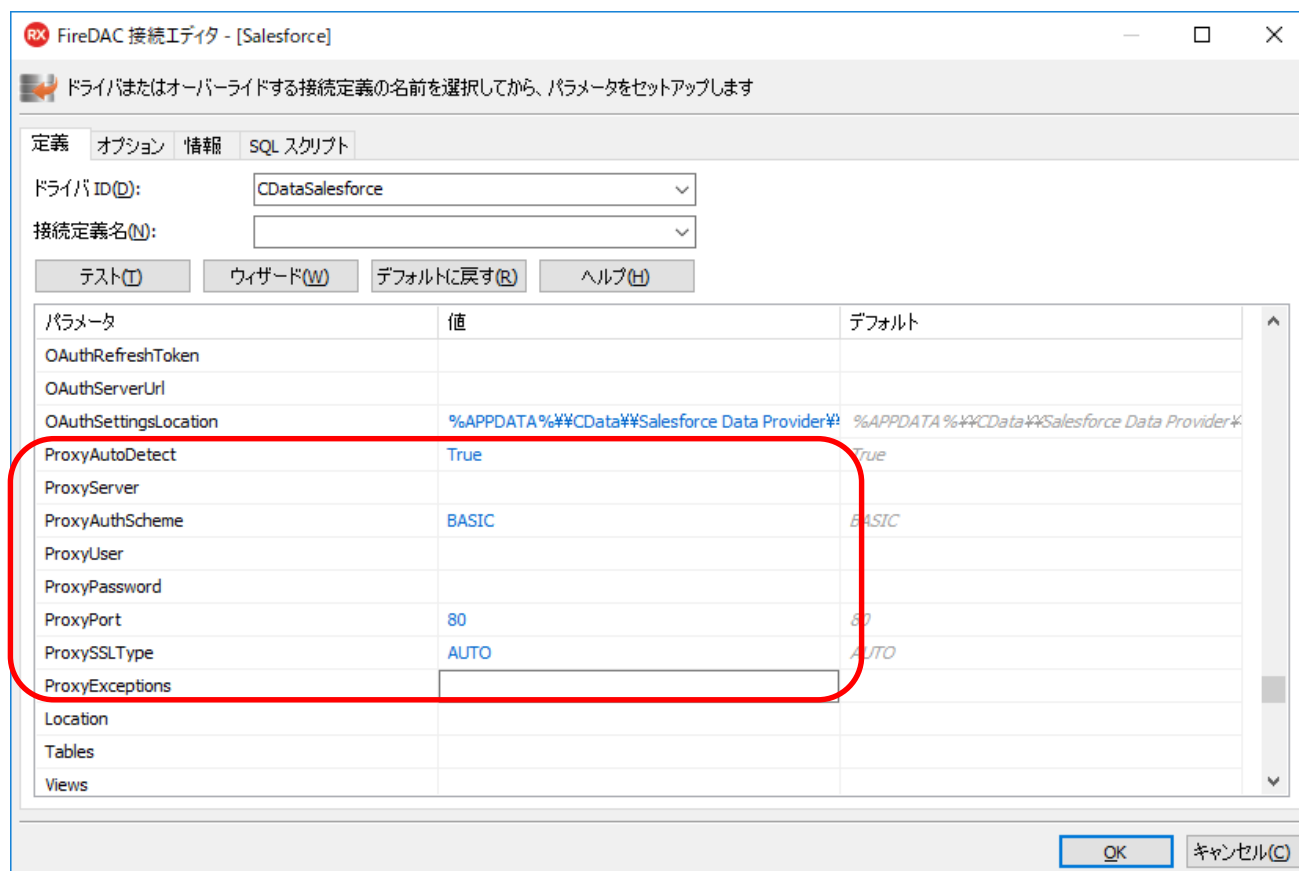
This is a table representing the Account entities in Salesforce.

Name	Type	ReadOnly	References
Id [KEY]	String	True	
IsDeleted	Boolean	True	
MasterRecordId	String	True	Account.Id
Name	String	False	
Type	String	False	
ParentId	String	False	Account.Id
BillingStreet	String	False	

# Enterprise Connectors利用時の補足

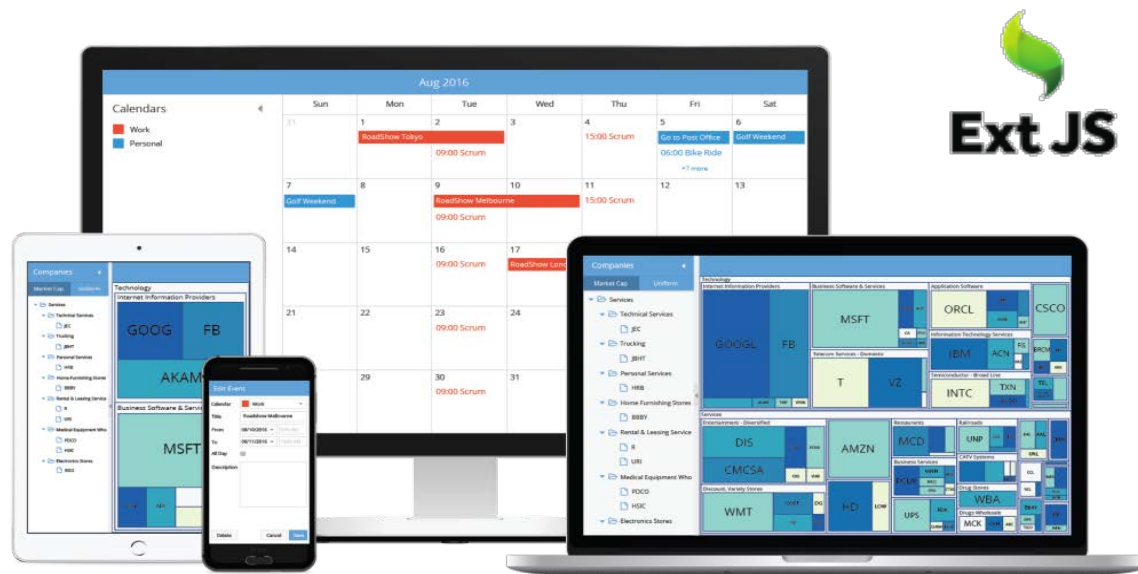
## ■ 補足：プロキシ対応

プロキシの設定もTFDConnectionのパラメータで対応可能  
(サーバを配置する環境によっては便利)



# Sencha等のWebアプリケーションへの連携

- Delphi以外のアプリケーションでの利用  
中間サーバから一般的なJSON形式でRESTの結果を提供すれば、Sencha等で作成したWebアプリケーションで利用することも可能。  
(Senchaとの連携については午後のセッションをご参考ください)



# Sencha等のWebアプリケーションへの連携

- 一般的なJSON形式でRESTの結果を戻す

```
procedure TCUSTResource1.Get(const AContext: TEndpointContext; const ARequest:
TEndpointRequest; const AResponse: TEndpointResponse);
var
  JSONResponse: TJSONObject;
  JSONArray: TJSONArray;
  JSONRecord: TJSONObject;
  aField: TField;
  count: Integer;
begin
  JSONResponse := TJSONObject.Create;
  JSONArray := TJSONArray.Create;
  count := 0;
```



# Sencha等のWebアプリケーションへの連携

- 一般的なJSON形式でRESTの結果を戻す

```
// データセットの結果を1件ずつ取得してJSONArrayに追加する
FDTable1.Open;
while (not(FDTable1.Eof)) do
begin
    JSONRecord := TJSONObject.Create;

    for aField in FDTable1.Fields do
    begin
        JSONRecord.AddPair(LowerCase(aField.FieldName), aField.AsString);
    end;

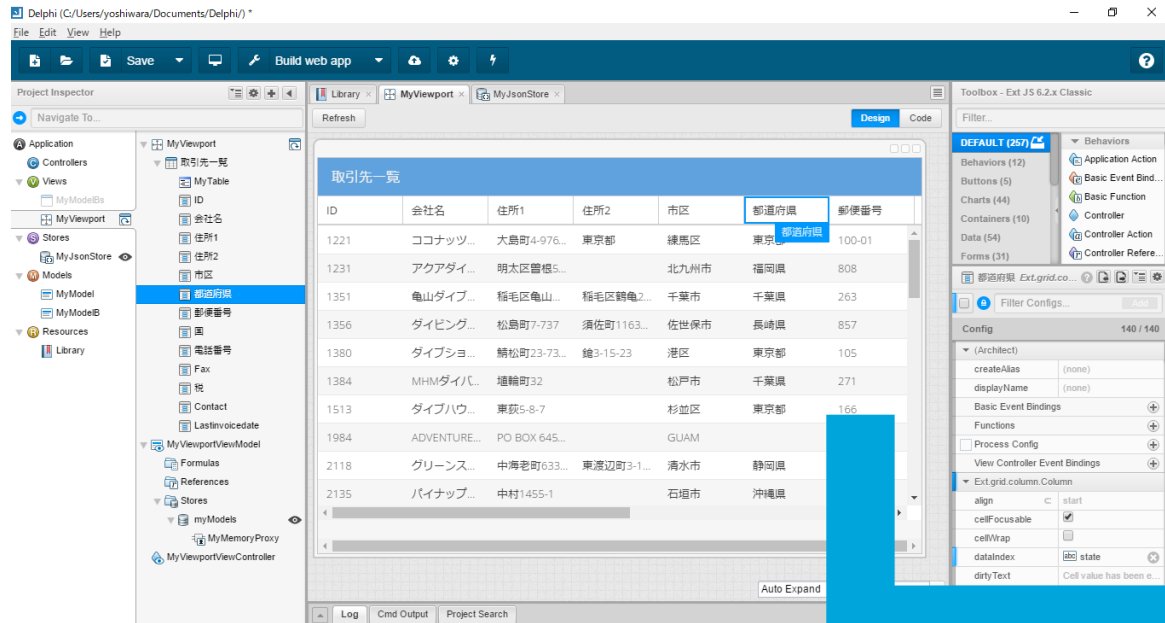
    JSONArray.Add(JSONRecord);
    FDTable1.Next;
    inc(count);
end;

AResponse.Body.SetValue(JSONArray, True);
end;
```

# Sencha等のWebアプリケーションへの連携

- SenchaからのアクセスでもDelphiと同じようにデータを利用

## Sencha Architect



## ブラウザで実行



クロスドメインがネックになる場合は、RAD Serverのemsserver.iniで設定するか、JSONPで返却する等の工夫が必要  
**Marco Tech Blogが参考になります**

<http://blog.marcocantu.com/blog/2017-october-ajax-jsonp-radserver-extjs.html>

# まとめ

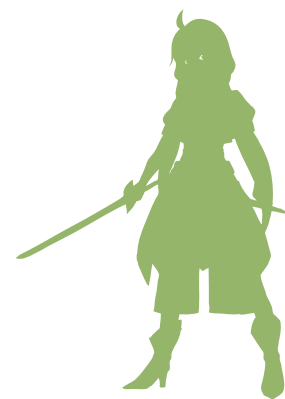
- 多層アプリケーションによる機能の分離はメリットが多く、マルチデバイスでシステムを展開していく上では非常に有効。
- DataSnap Serverはサーバ開発用のSDK。開発できる自由度が高い。
- RAD Serverは標準で豊富な機能が提供されるサーバ。
- Enterprise Connectorsを中間サーバで利用すれば、クラウドサービスやパッケージなどの連携が容易。
- 中間サーバのサービスで返却するJSON形式を配慮すればDelphi以外のアプリケーションでも活用できる。

# THANKS!

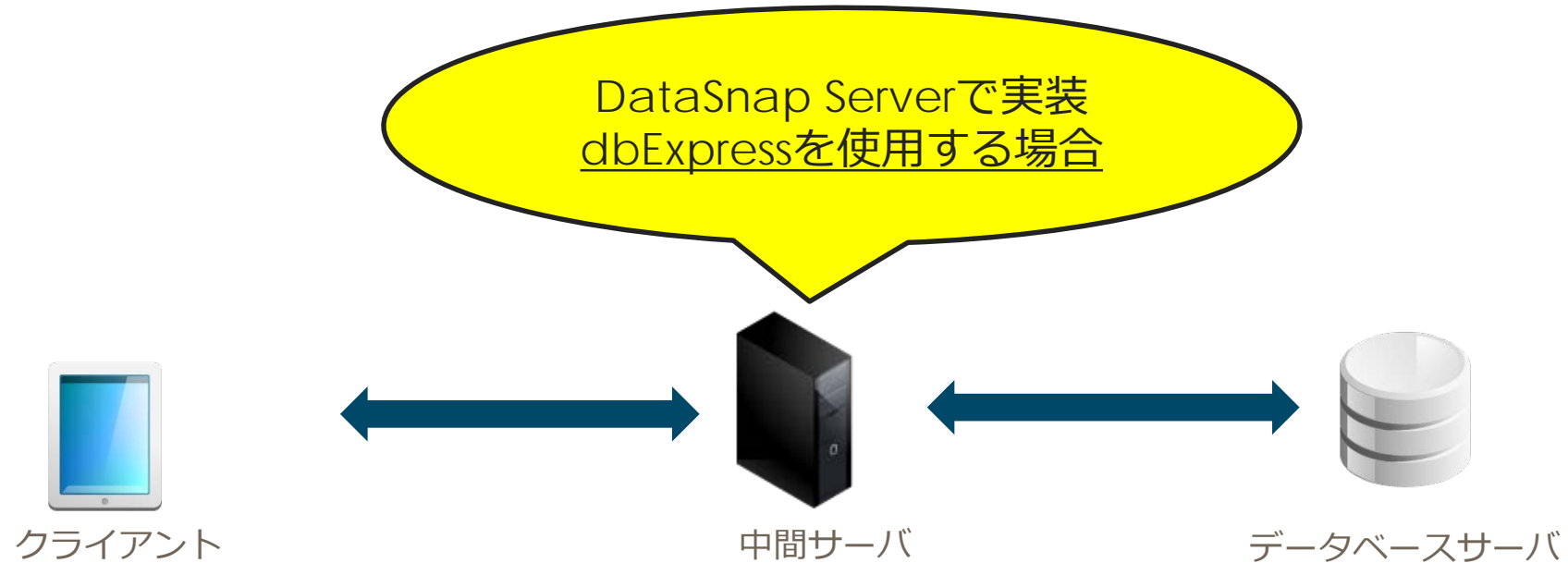
[www.embarcadero.com/jp](http://www.embarcadero.com/jp)

第35回 エンバカデロ・デベロッパーキャンプ

# 補足：DataSnap Serverを使った実装手順（dbExpress）



# 補足 : DataSnap Serverを使った実装手順 (dbExpress)

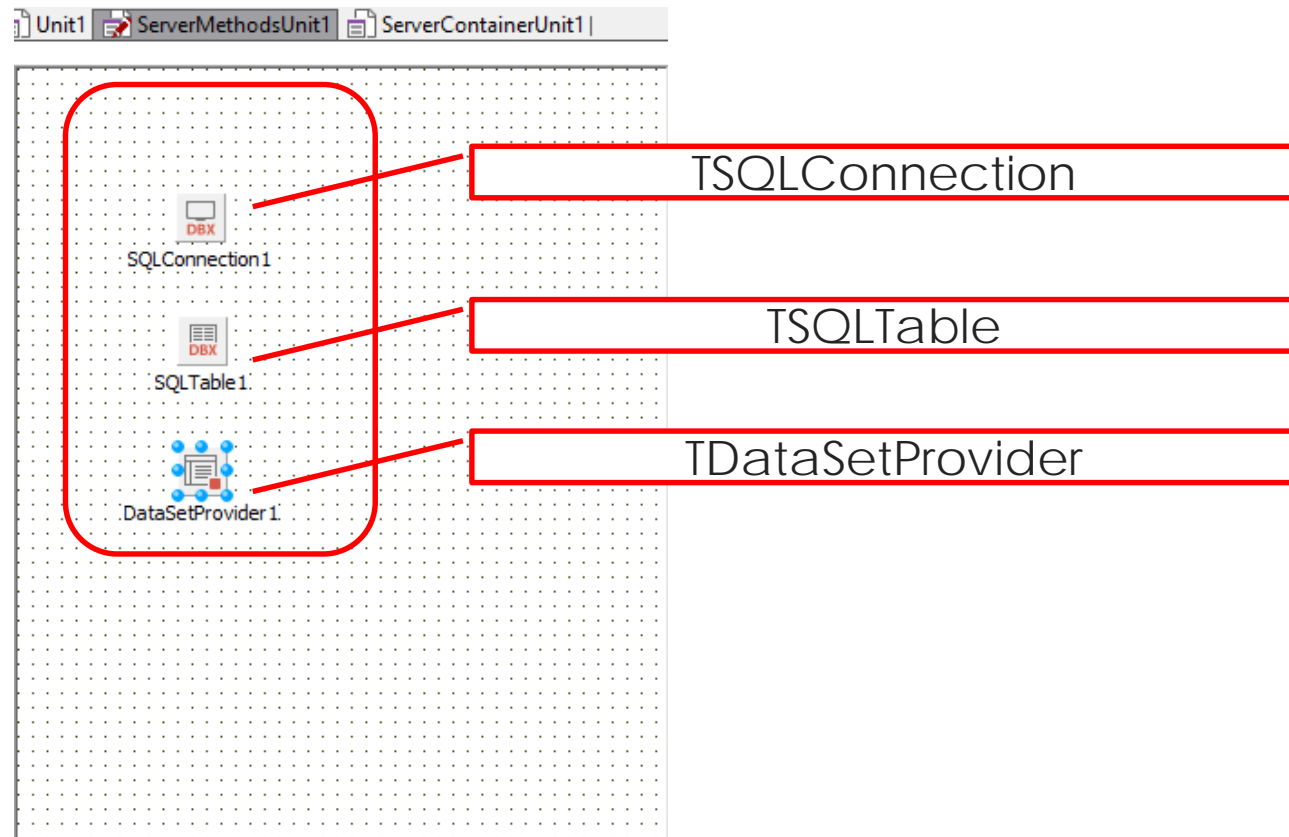


# 補足 : DataSnap Serverを使った実装手順 (dbExpress)

## ■ 実装手順1

サーバ機能のコンポーネント配置

ServerMethodsユニットに機能を実装



# 補足 : DataSnap Serverを使った実装手順 (dbExpress)

- 実装手順2  
コンポーネントの設定

## TSQLConnection

キー	値
DriverName	Interbase
Database	localhost:C:#ProgramData#Embarcadero#InterBase#gds_db#examples#database#employee.gdb
RoleName	RoleName
User_Name	sysdba
Password	masterkey
ServerCharSet	
SQLDialect	3
ErrorResourceFile	
LocaleCode	0000
BlobSize	-1
CommitRetain	False
WaitOnLocks	True
IsolationLevel	ReadCommitted
Trim Char	False

## TSQLTable

SQLTable1 TSQLTable

検索

プロパティ イベント

IndexName	
LiveBinding デザイン	LiveBinding デザイン
MasterFields	
MasterSource	
MaxBlobSize	-1
Name	SQLTable1
NumericMapping	<input type="checkbox"/> False
ObjectView	<input type="checkbox"/> False
SchemaName	
SQLConnection	SQLConnection1
TableName	CUSTOMER
Tag	0

## TDataSetProvider

DataSetProvider1 TDataSetProvider

検索

プロパティ イベント

Constraints	<input checked="" type="checkbox"/> True
DataSet	SQLTable1
Exported	<input checked="" type="checkbox"/> True
LiveBinding デザイン	LiveBinding デザイン
Name	DataSetProvider1
Options	xt,poUseQuoteChar
ResolveToDataSet	<input type="checkbox"/> False

Optionプロパティの  
"poAllowCommandText" を  
"True" に設定しておく  
とクライアント側からSQLの指定も可能



# 補足：DataSnap Serverを使った実装手順（dbExpress）

DataSnap Serverで実装  
dbExpressを使用する場合



クライアント



中間サーバ

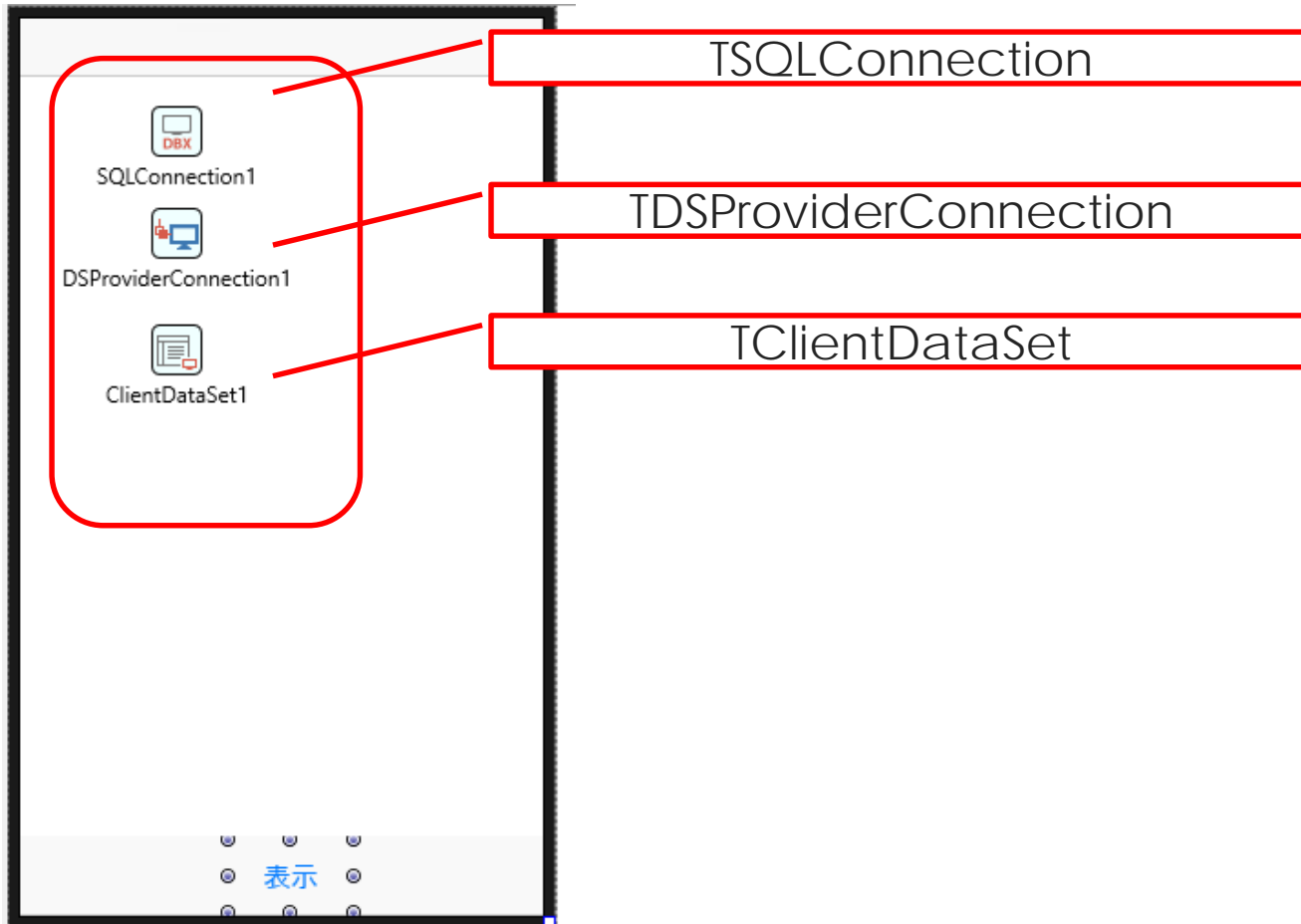


データベースサーバ

# 補足 : DataSnap Serverを使った実装手順 (dbExpress)

## ■ 実装手順3

クライアントアプリケーションのコンポーネント配置



# 補足 : DataSnap Serverを使った実装手順 (dbExpress)

## ■ 実装手順4 コンポーネントの設定

### TSQLConnection

キー	値
DriverName	DataSnap
HostName	localhost
port	211

ドライバはDataSnapを指定し、サーバIPやポートを設定  
※localhostでは開発端末上でしか接続できません。

### TDSProviderConnection

ServerClassName: TServerMethods1

サーバ側のクラス名を指定 (直接入力)

### TClientDataSet

ProviderName: DataSetProvider1

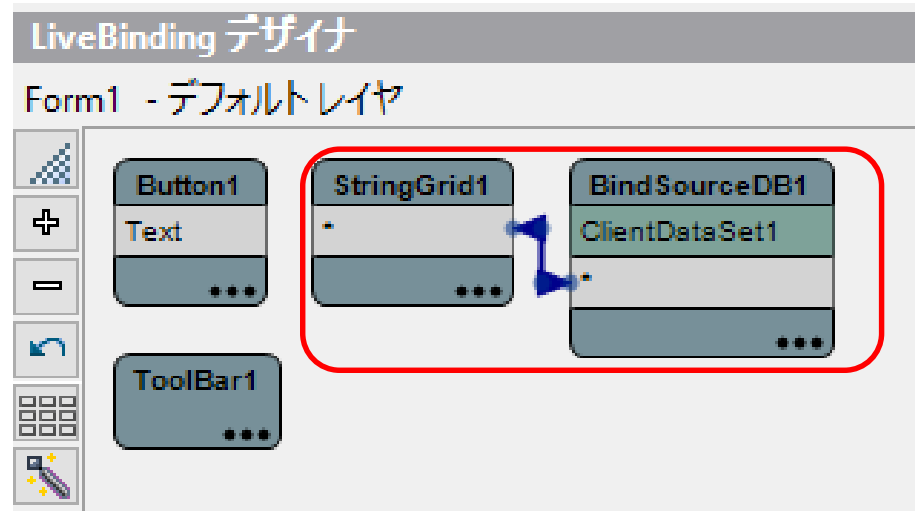
RemoteServer: DSPProviderConnection1

RemoteServerを指定すれば、ProviderNameは候補から選択 (サーバ側が起動していること)

# 補足 : DataSnap Serverを使った実装手順 (dbExpress)

- 実装手順5  
コンポーネントの設定

ライブバインディング設定



# 補足 : DataSnap Serverを使った実装手順 (dbExpress)

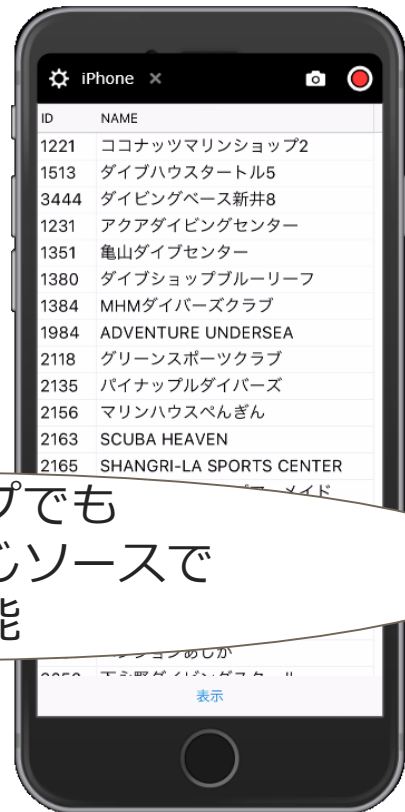
- 実装手順6  
データ取得の機能を実装

```
procedure TForm1.Button1Click(Sender: TObject);  
Begin  
    //設定しているClientDataSetを開くだけ  
    ClientDataSet1.Open;  
end;
```

# 補足 : DataSnap Serverを使った実装手順 (dbExpress)

## ■ 実行

ID	NAME
1221	ココナッツマリンショップ2
1513	ダイブハウスタートル5
3444	ダイビングベース新井8
1231	アクアダイビングセンター
1351	亀山ダイブセンター
1380	ダイブショップブルーリーフ
1384	MHMダイバーズクラブ
1984	ADVENTURE UNDERSEA
2118	グリーンスポーツクラブ
2135	パイナップルダイバーズ
2156	マリンハウスペンぎん
2163	SCUBA HEAVEN
2165	SHANGRI-LA SPORTS CENTER
2353	
2975	ダイブリゾートハワイ
2984	サンセットダイビングサービス
3041	マリンショップアクア



デスクトップでも  
モバイルでも同じソースで  
実装可能

クライアントアプリ



中間サーバ



データベースサーバ

# 補足：DataSnap Serverを使った実装手順（dbExpress）

## ■ DBエンジンコンポーネントの組み合わせ

サーバ側をFireDACのコンポーネントを使い、クライアント側をdbExpressのコンポーネントで開発することも可能。

実装内容はコンポーネントが変わるだけでdbExpressの作り方とほぼ同じです。

例) 第29回のエンバカデロ・デベロッパークャンプの  
「こんなに簡単！ Delphi によるiOS/Android業務アプリケーション開発！」  
のセッション内容がこの構成になっています。アーカイブの資料等をご参考ください。  
<https://www.embarcadero.com/jp/events/developer-camp/archive>

## DataSnap Serverで可能な組み合わせ

組み合わせ	
クライアント側（サーバ接続）	サーバ側（DB接続）
FireDAC	FireDAC
dbExpress	FireDAC
dbExpress	dbExpress