【B1】Delphi/C++Builderテクニカルセッション

「マルチデバイスに対応できるDBアクセス形態を作るには？実践テクニック」

株式会社ドリームハイブ
代表取締役 ITコンサルタント 山本 悟

マルチデバイスに対応できるDBアクセス形態を作るには？実践テクニック

はじめに

自己紹介

- 会社：株式会社　ドリームハイブ
  - 会社URL　：http://www.dreamhive.co.jp/
  - お得なコンテンツ配信サイト　：http://dhive.jp/
- 名前：山本 悟 (やまもと　さとる)
  - ドリームハイブ 代表取締役 & ITコンサルタント
  - ブログ　：http://dhive.jp/blog/yama/
  - facebook　：http://www.facebook.com/kryu2
  - twitter　：http://twitter.com/kryu2
  - 山本はこんな感じの人：
    - 17歳からIT業界へ
    - Delphi は1.0からの親友
    - テレビ埼玉に出たり
    - ドリームハイブの経営、ITコンサルティング、システム開発、スピーカーなどが主な仕事

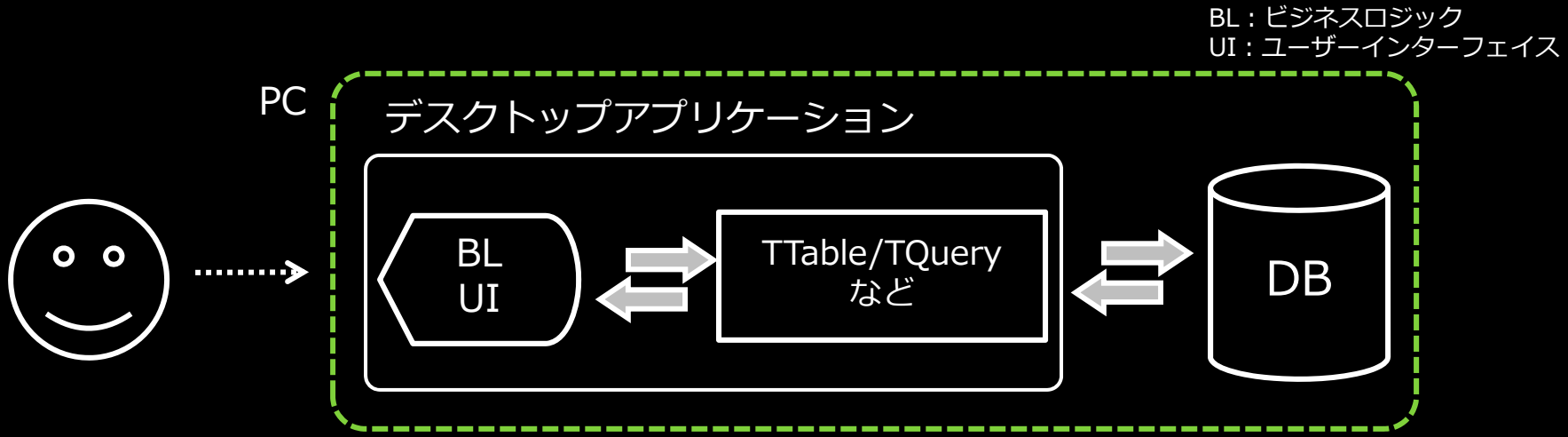
本日の内容

- 拡張しづらいシステム構成
- 問題が起こるケース
- 拡張しやすいシステム構成
- 構成を具体的に考える
- ベストプラクティスのまとめ

- このセッションは、テクニカル・セッションです
- Delphi/C++ Builder製のデータベースアプリケーションの
  構築方法について、私の主観と経験に基づいてお話しいたします
- あなたにとって最適解では無いかもしれませんが、参考になると思います

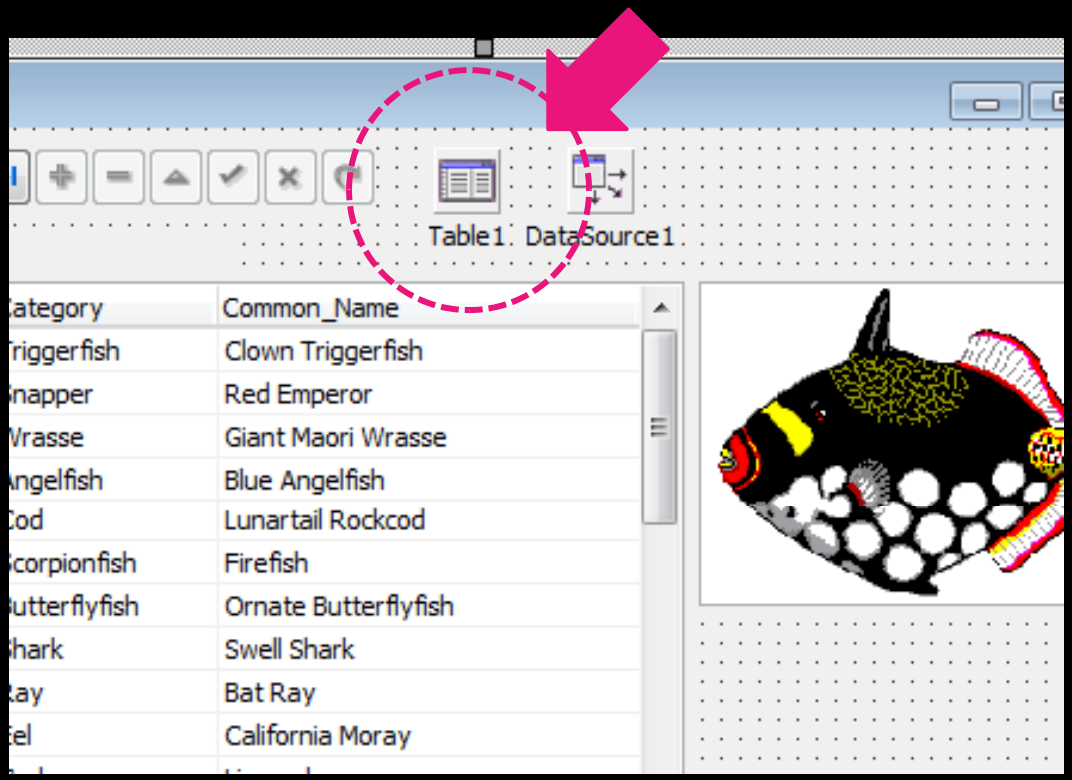
拡張しづらいシステムの特徴

- 単層/二層構造
- 単一クライアント
- デスクトップアプリケーション



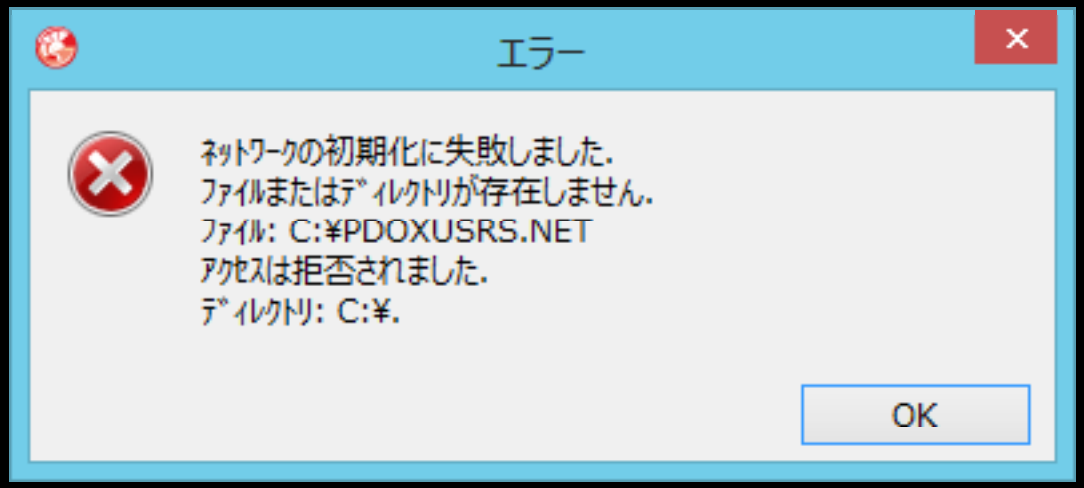
BDEのみはもっともダメなパターン

- FormにTTableとか直置きのアプリのパターン



そのままだと最新OSでは動きません

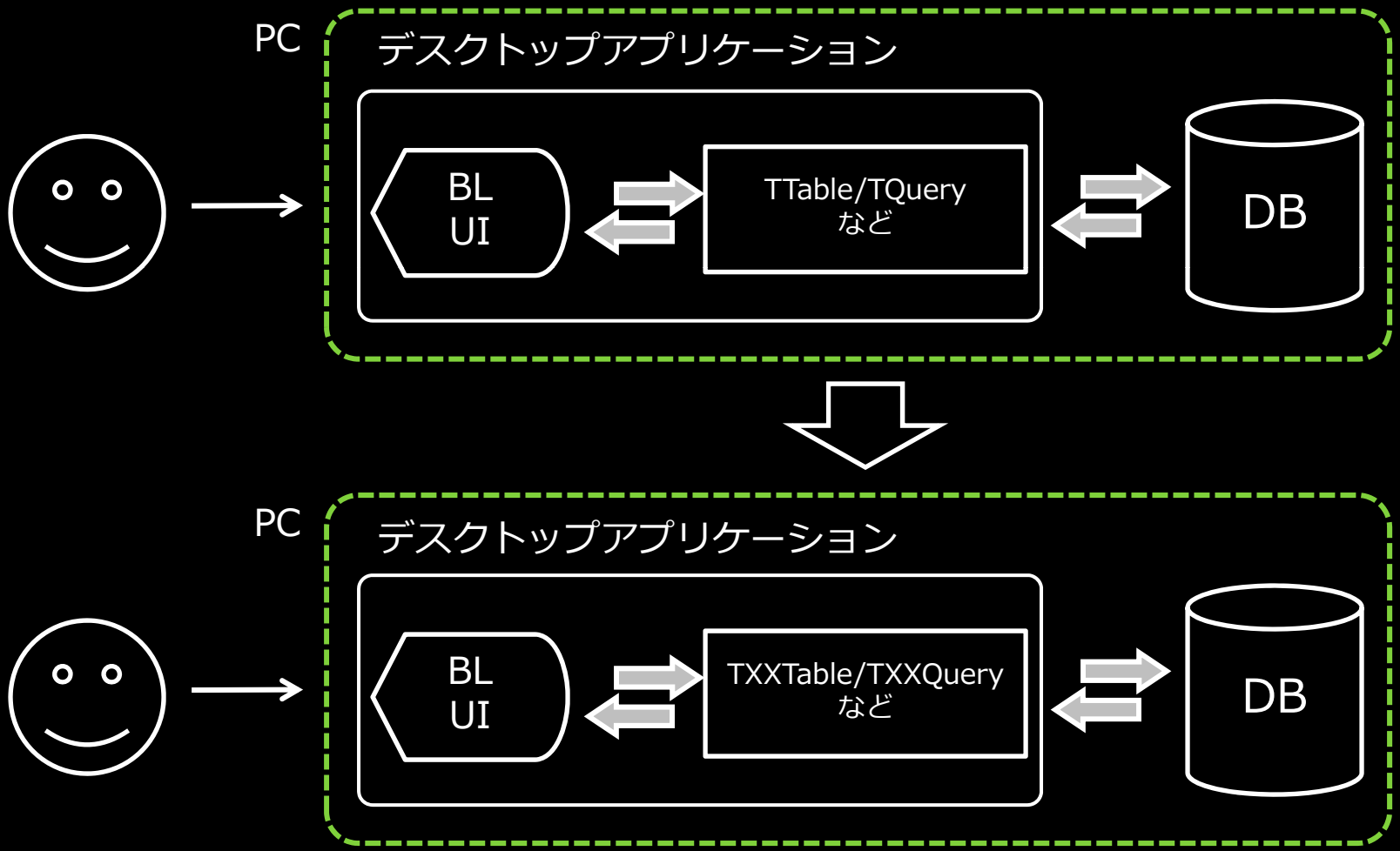
- ルートディレクトリにファイル書き込んじゃうし



対処方法 → http://dhive.jp/blog/yama/?p=2679

最低限、BDEアプリは互換コンポーネントで置き換えましょう

BL：ビジネスロジック
UI：ユーザーインターフェイス



※移行は簡単ですが、
ユーザーが単数の場合の暫定的な対処です

問題が起こるケース

- それは要求“変更”！！
  - 例)
    - 複数のPCから見られるのは当たり前だろオーダーが入った！
    - 社長からWebが主流らしいじゃんの鶴の一声が出た！
    - 最近流行のモバイルアプリに対応しろオーダーが入った！

- 何が問題となるのか
  - 複数クライアントから同時アクセスを想定しなければならない
  - クライアント環境が複数存在する事になる
  - クライアントへの配布が難しい
  - クライアントアプリケーションのバージョン管理が難しい
  - 他のシステムやライブラリを利用する確率が増える
  - 社内サーバに社外から直接アクセスできなければ、中継サーバが必要になる
  - 社内外間のネットワーク負荷が増大する
  - DBへの同時接続数の問題が発生する

拡張しやすいシステムの特徴

- 移り変わる運用環境
  - 企業システムの多くがWebを主軸に
  - BYODの増加
    - モバイル
    - タブレット
  - 整備されてきたクラウドサーバー環境
    - Amazon Web Services
    - Windows Azure Platform
    - Google App Engine

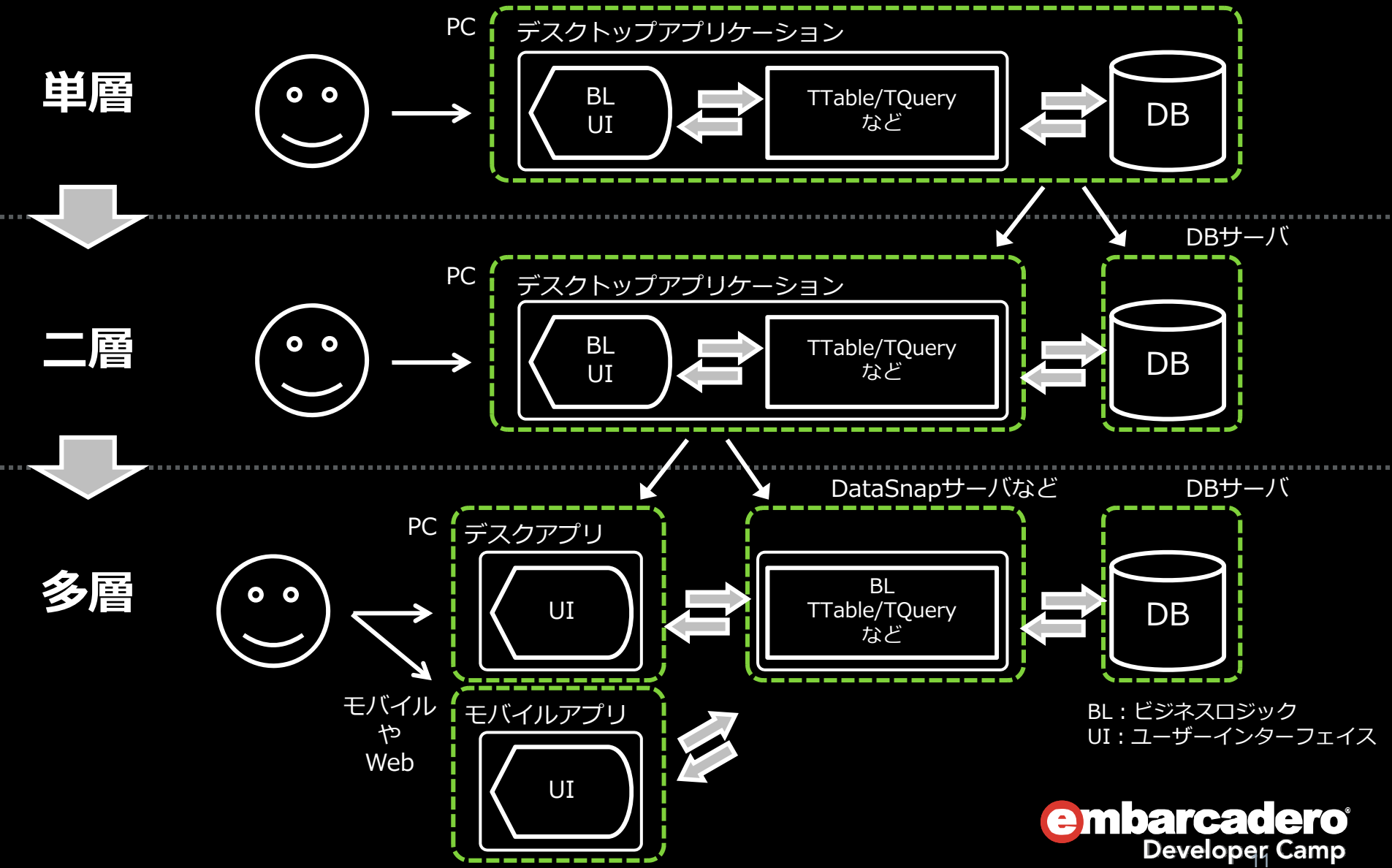
これまでも、これからも

今後ますます増える
クライアントデバイスの種類

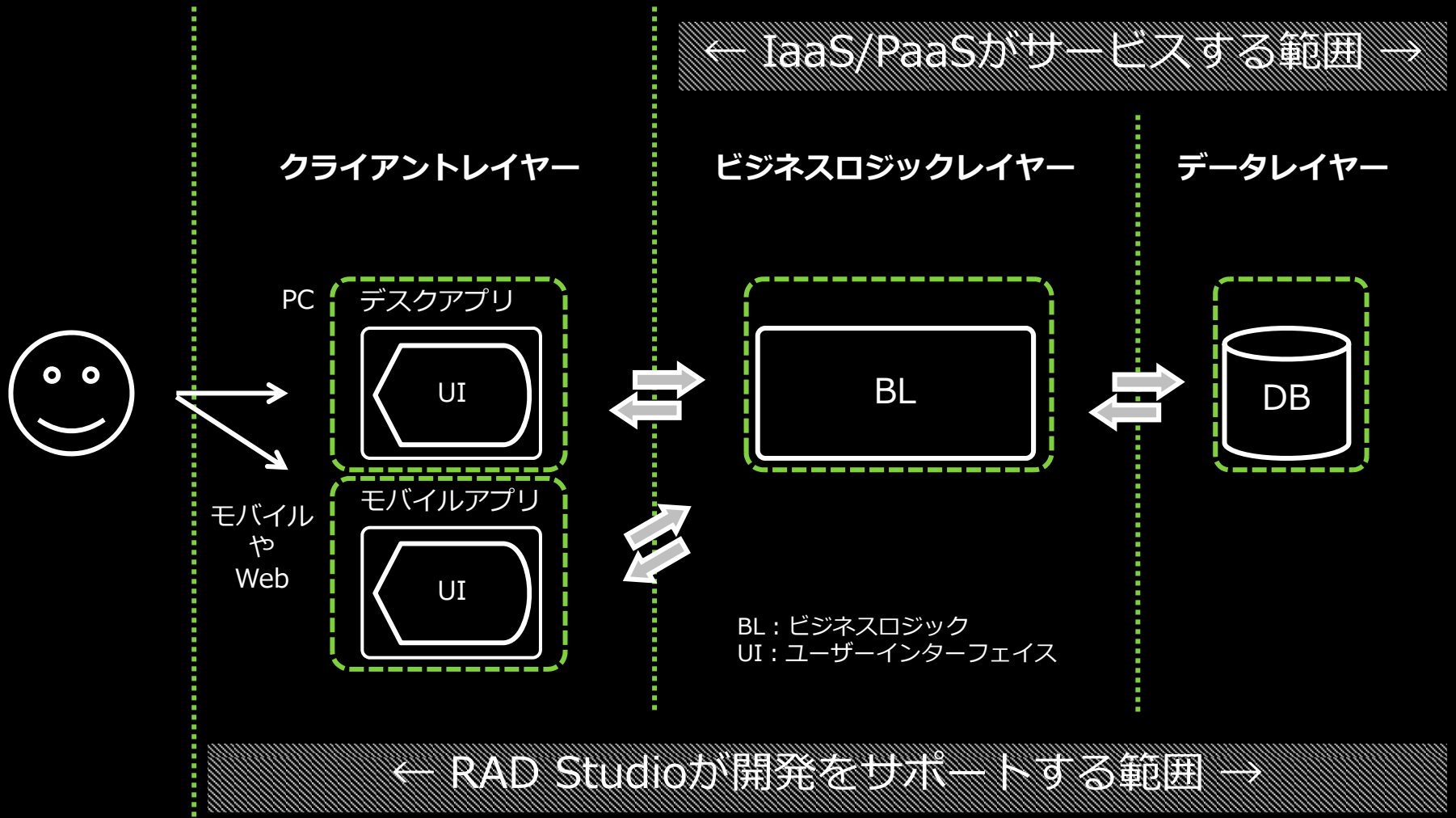
IaaS(Infrastructure as a Service)、PaaS(Platform as a Service)などがキーワード。
オートスケーリングなど柔軟な拡張性を持っています

拡張性を確保するには機能ごとのレイヤー分割が必須！

レイヤー分割＝多層化


単層
PC
デスクトップアプリケーション
BL
UI
TTable/TQuery
など
DB
二層
DBサーバ
PC
デスクトップアプリケーション
BL
UI
TTable/TQuery
など
DB
多層
DataSnapサーバなど
DBサーバ
PC
デスクアプリ
UI
BL
TTable/TQuery
など
DB
モバイル
や
Web
モバイルアプリ
UI
BL：ビジネスロジック
UI：ユーザーインターフェイス

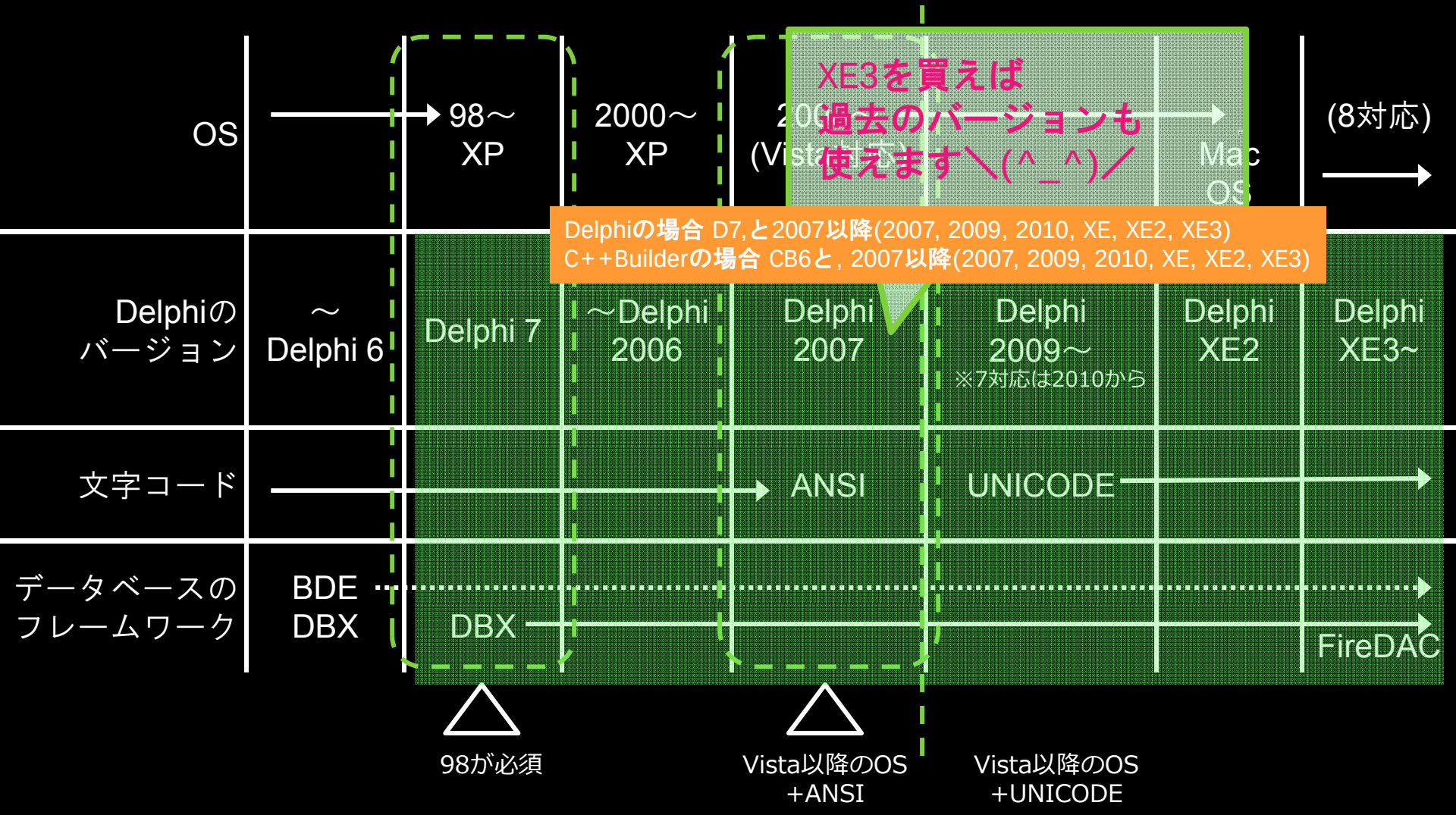
各レイヤーごとに変更を吸収できる



Delphi/C++ Builderの構成を具体的に考える

- 拡張に強い設計を考える前に気をつけるべきポイント
  - 動作するOSのバージョン
  - 開発するDelphi/C++ Builderのバージョン
  - 使用する文字コード
  - データベース接続で利用しているフレームワークの種類

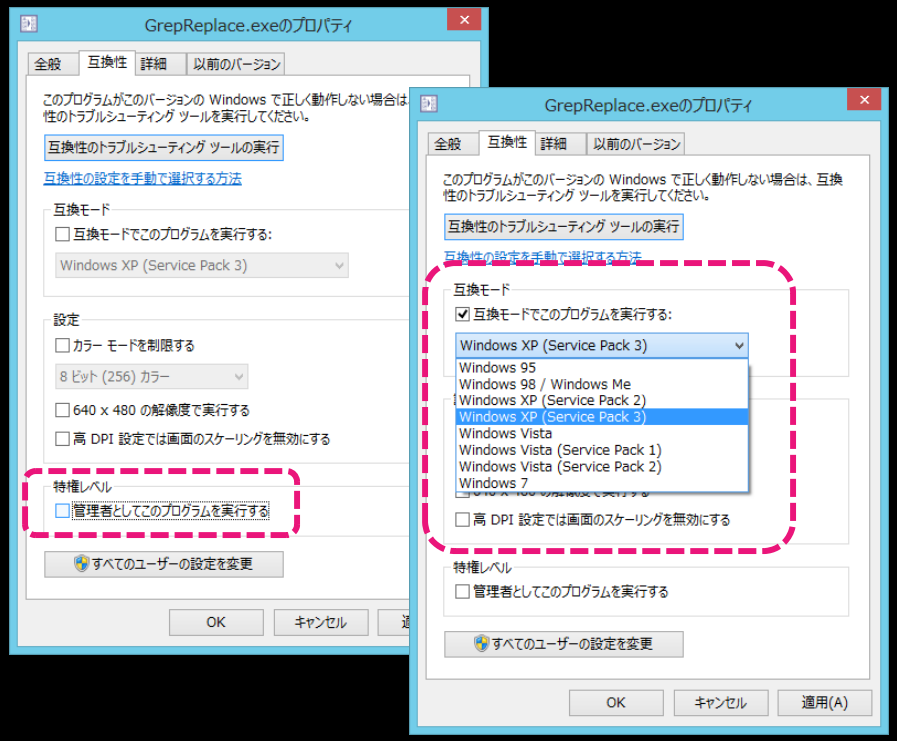
構築する環境の基準は？


OS
98～
XP
2000～
XP
(8対応)
Mac
OS
XE3を買えば
過去のバージョンも
使えます＼(^_^)／
Delphiの場合 D7,と2007以降(2007, 2009, 2010, XE, XE2, XE3)
C++Builderの場合 CB6と，2007以降(2007, 2009, 2010, XE, XE2, XE3)
Delphiの
バージョン
～
Delphi 6
Delphi 7
～Delphi
2006
Delphi
2007
Delphi
2009～
※7対応は2010から
Delphi
XE2
Delphi
XE3~
文字コード
ANSI
UNICODE
データベースの
フレームワーク
BDE
DBX
DBX
FireDAC
98が必須
Vista以降のOS
+ANSI
Vista以降のOS
+UNICODE

過去のプロジェクトの移行はどうする？

とりあえず、だいたい動きます＼(^_^)／

- プログラムの「互換性」タブを使用するとか



- 新しいDelphiに切り替えるだけでも効果ありです
  - ドライバ・パッケージ・ソースなどが更新されています
  - 新しいコンパイラによる効果(最適化や新機能が有効になる)があります

規模が小さい
プロジェクトの移行であれば

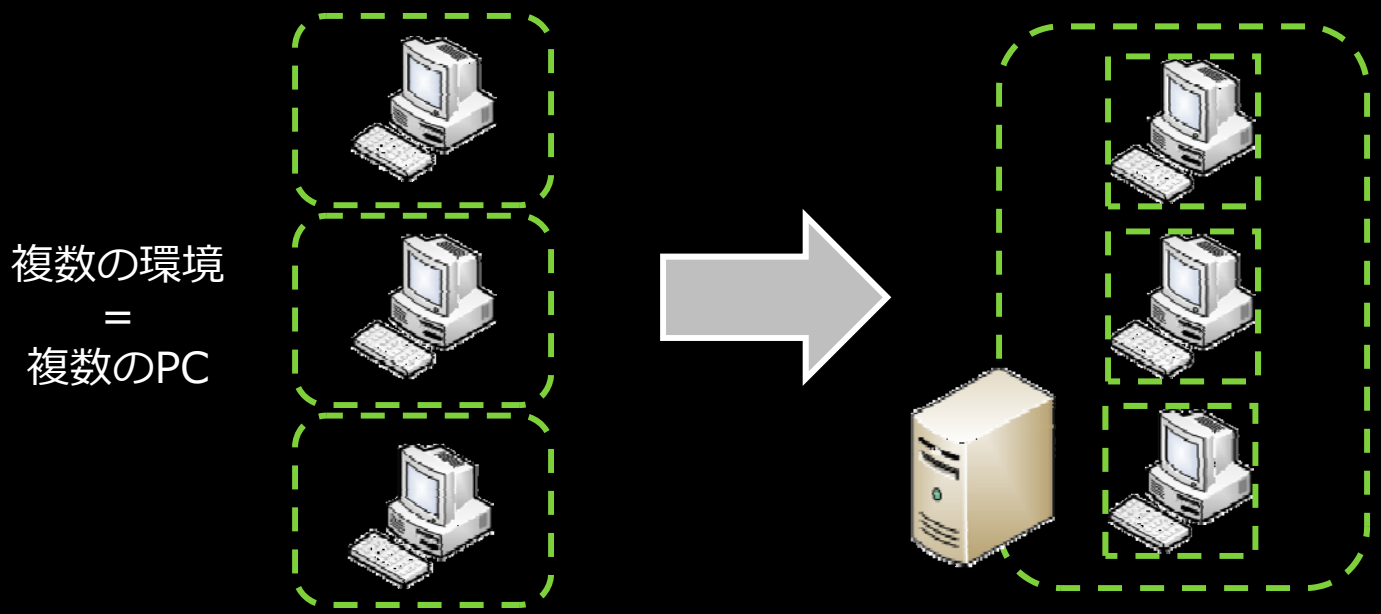
- とりあえず新しいDelphiで開いてみましょう

ただし、規模の大小に関わらず、

- Delphi 2007より前のプロジェクトについては
  一度Delphi 2007へ移行してから、
  最新のDelphiに移行することを強くオススメします

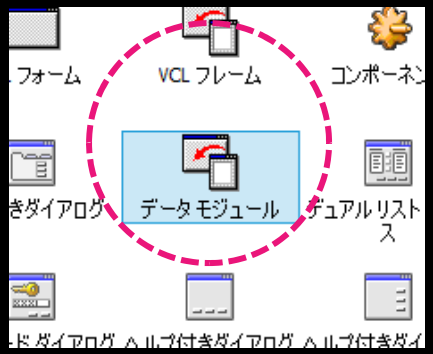
開発環境はどうする？

- 仮想環境を構築するのがオススメ
  - 仮想化：1台のコンピュータを、あたかも複数台のコンピュータであるかのように論理的に分割し、それぞれに別のOSを動作させることで、複数の環境を少ないリソースで比較的安全に構築することができる
    - Hyper-V：マイクロソフト
    - VMWare：ヴイエムウェア
    - Xen：シトリックス・システムズ・ジャパン など



拡張しやすい構成

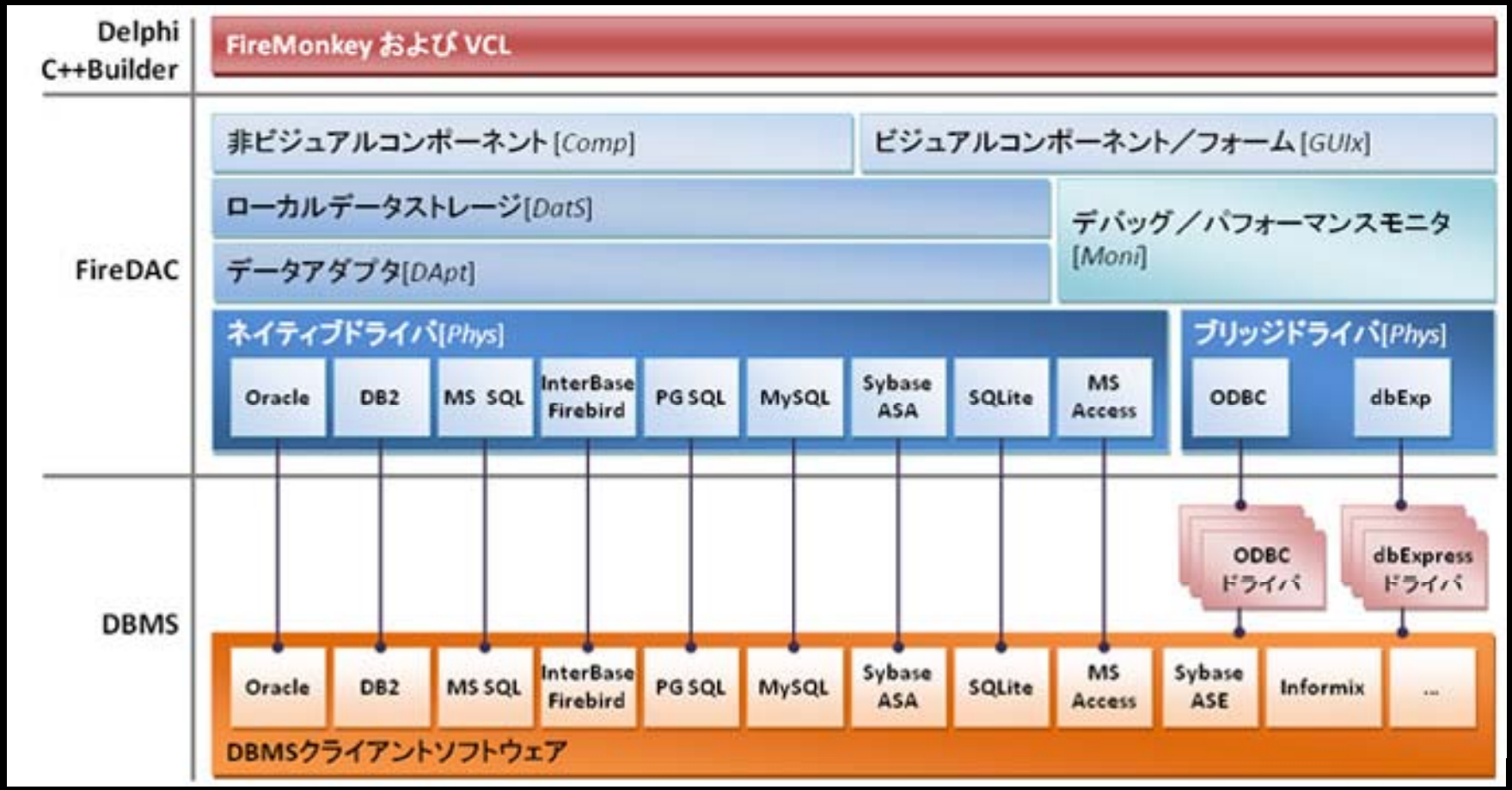
- 拡張しやすいとは
  - 理想は、変更数ゼロ、コードの追加のみで機能追加などができること
- 設計の例
  - データモジュールを利用する
  - クラスだけでなく、例外処理などもカプセル化する
  - BDE+ClientDataSetへの変更作業を中間に挟む
  - FireDACを利用する
  - メソッド内でオブジェクト参照を付け替える




```
var
  MyMemo: TMemo;
begin
  MyMemo := Memo1;

  MyMemo.Lines.Add('AAA');
  MyMemo.Lines.Add('BBB');
  MyMemo.Lines.Add('CCC');
end;
```

ちょっとだけFireDACの説明



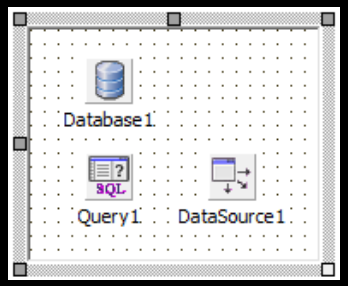
拡張しやすい構成への変更例

絶対ダメ構成

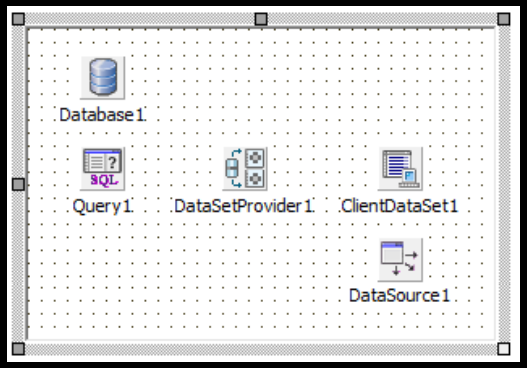
とりあえず逃げの構成

素晴らしい構成

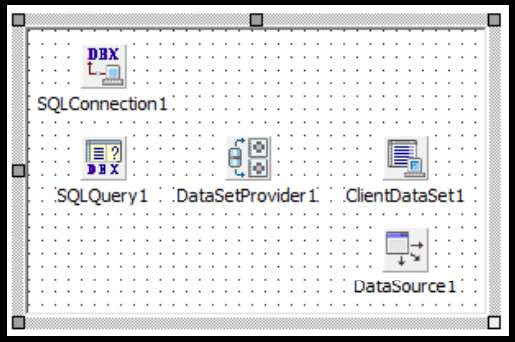
BDE構成の
データモジュール



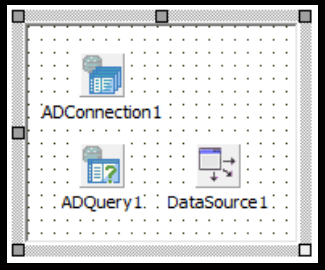
BDE+ClientDataSet構成の
データモジュール



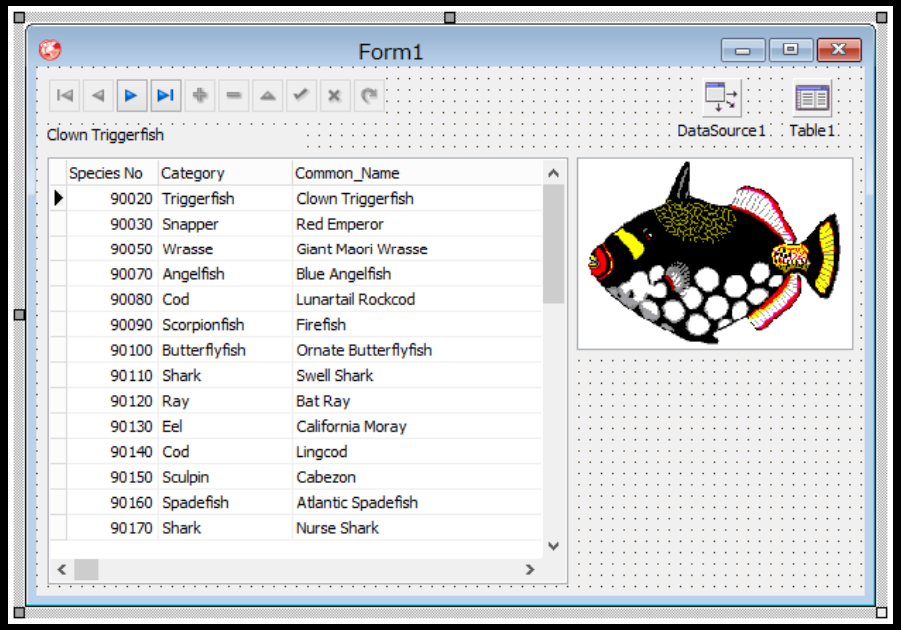
DBX構成の
データモジュール

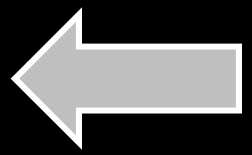


FireDAC構成の
データモジュール

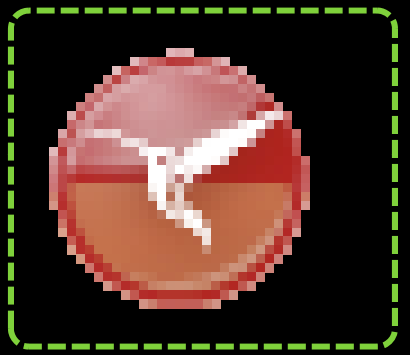


BDEのみ

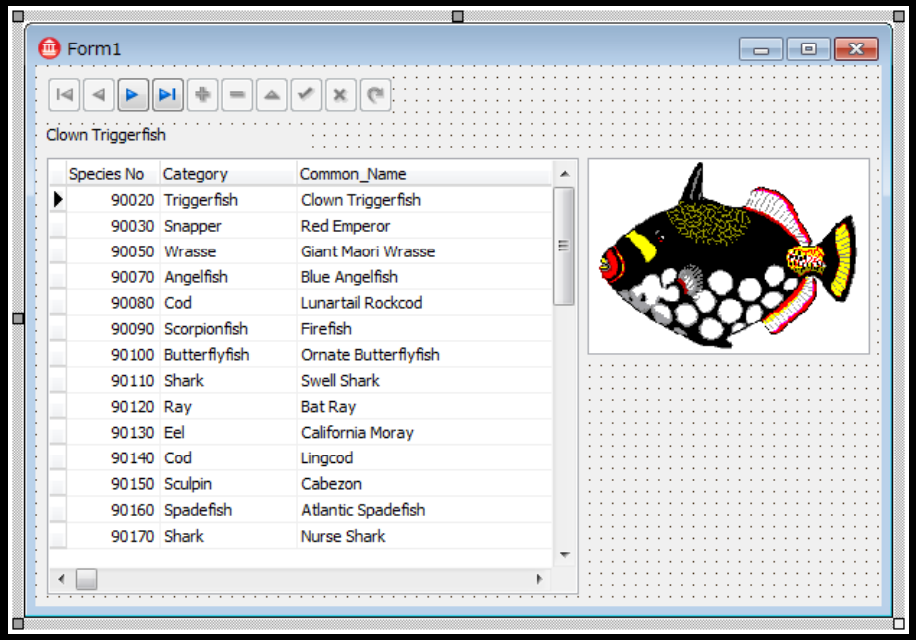


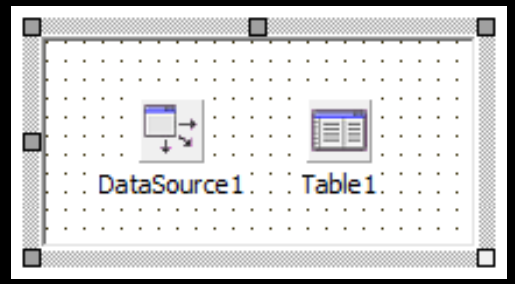


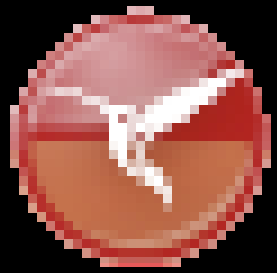
データベース



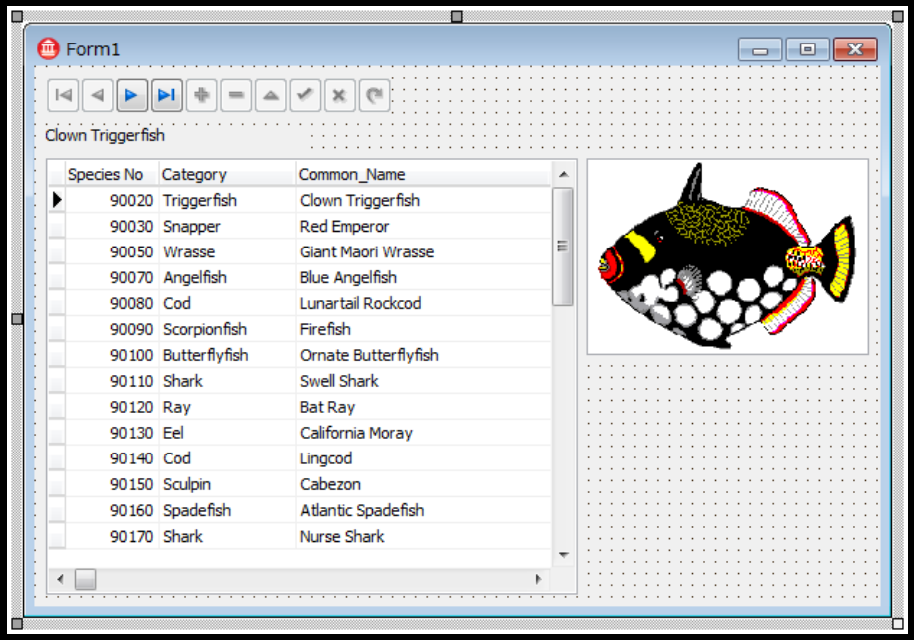
BDE+DataModule

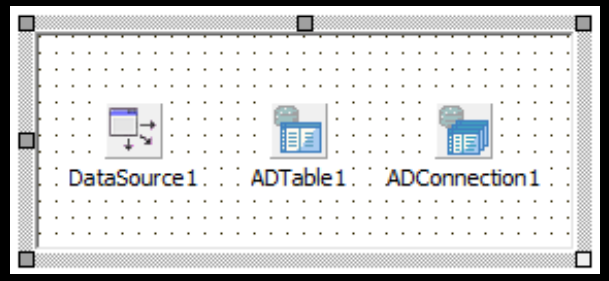






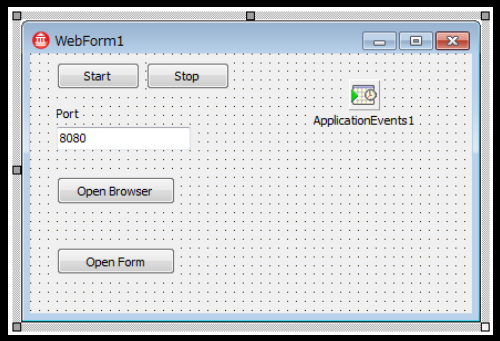
FireDAC + DataModule

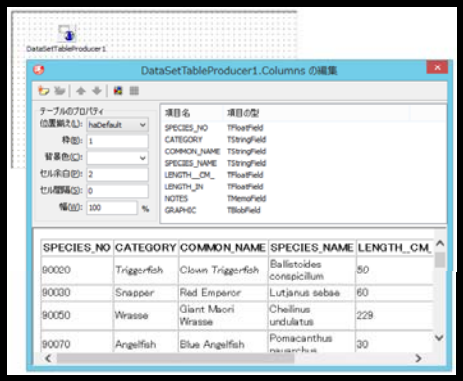


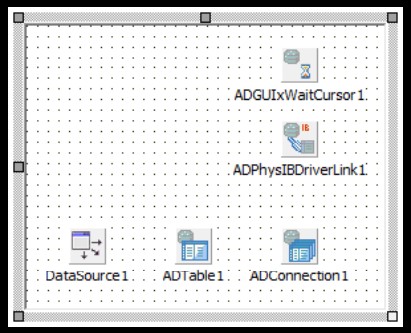


FireDAC+DataModule+IntraWeb

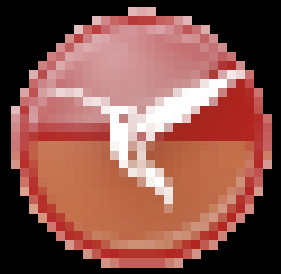
アプリ構成



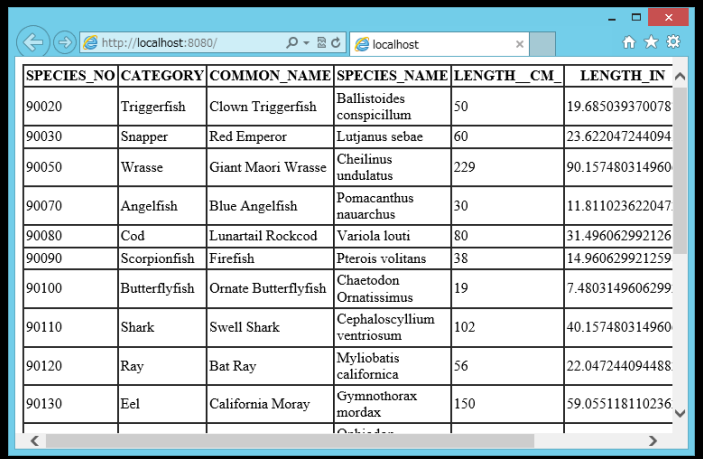




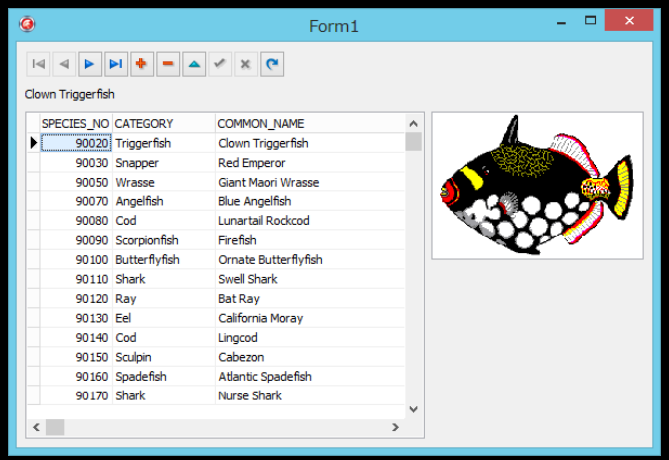
データベース



実行結果



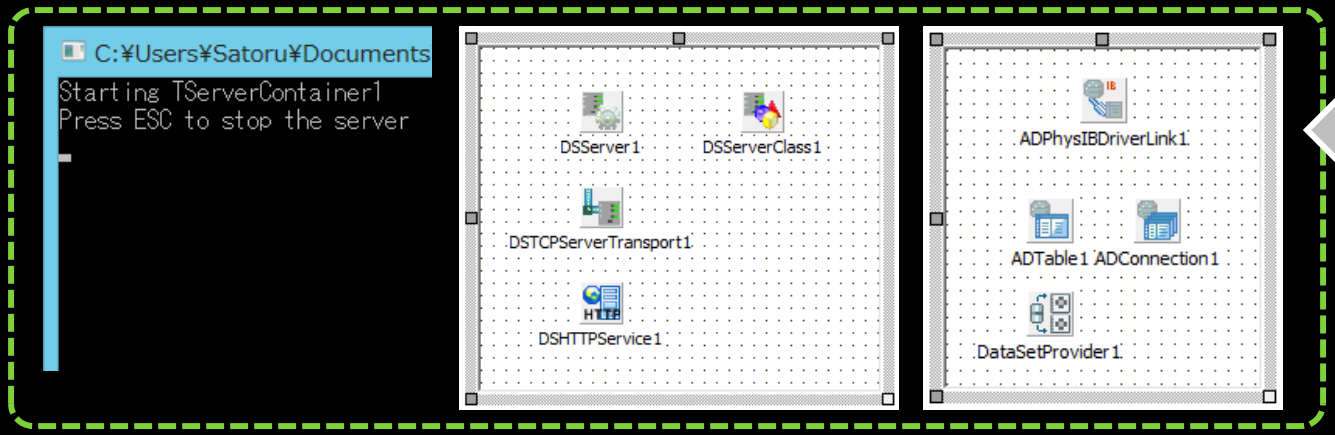
SPECIES_NO	CATEGORY	COMMON_NAME	SPECIES_NAME	LENGTH_CM_	LENGTH_IN
90020	Triggerfish	Clown Triggerfish	Ballistoides conspicillum	50	19.685039370078
90030	Snapper	Red Emperor	Lutjanus sebae	60	23.622047244094
90050	Wrasse	Giant Maori Wrasse	Cheilinus undulatus	229	90.157480314960
90070	Angelfish	Blue Angelfish	Pomacanthus nauarchus	30	11.811023622047
90080	Cod	Lunartail Rockcod	Variola louti	80	31.496062992126
90090	Scorpionfish	Firefish	Pterois volitans	38	14.960629921259
90100	Butterflyfish	Ornate Butterflyfish	Chaetodon Ornatissimus	19	7.4803149606299
90110	Shark	Swell Shark	Cephaloscyllium ventriosum	102	40.157480314960
90120	Ray	Bat Ray	Myliobatis californica	56	22.047244094488
90130	Eel	California Moray	Gymnothorax mordax	150	59.055118110236



SPECIES_NO	CATEGORY	COMMON_NAME
90020	Triggerfish	Clown Triggerfish
90030	Snapper	Red Emperor
90050	Wrasse	Giant Maori Wrasse
90070	Angelfish	Blue Angelfish
90080	Cod	Lunartail Rockcod
90090	Scorpionfish	Firefish
90100	Butterflyfish	Ornate Butterflyfish
90110	Shark	Swell Shark
90120	Ray	Bat Ray
90130	Eel	California Moray
90140	Cod	Lingcod
90150	Sculpin	Cabezon
90160	Spadefish	Atlantic Spadefish
90170	Shark	Nurse Shark

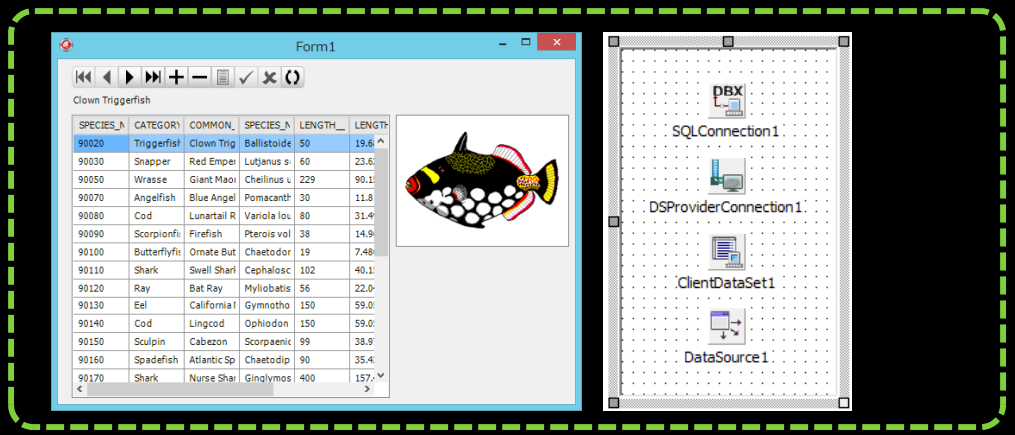
FireDAC+DataModule+DataSnap +FireMonkey

DataSnapServer.exe



データベース

DataSnapClient.exe



embarcadero Developer Camp

サンプル(見せる？作る？)

- DataModuleへの切り出し
- 既存クライアントの作成
- Webアプリの作成
- 更に・・
- DataSnapインターフェイスの作成
- iOSアプリの作成
- クラウドに展開してみる
- 案件成功ばんざーい＼(^_^)／

http://docwiki.embarcadero.com/RADStudio/ja/TDataModule_%E3%81%AE_ClassGroup_%E7%96%91%E4%BC%BC%E3%83%97%E3%83%AD%E3%83%91%E3%83%86%E3%82%A3

工数を比較してみる

	規模	単層/二層	多層
設計	小	☺☺☺	☺☺☺😐😐
	大	☺☺☺😐😐	☺☺☺😐😐
製造	小	☺☺☺	☺☺☺😐😐
	大	☺☺☺😐😐	☺☺☺😐😐
テスト	小	☺☺☺	☺☺☺
	大	☺☺☺😐😐	☺☺☺😐
移行	小	☺☺☺	☺☺☺
	大	☺☺☺😐😐☹☹	☺☺☺😐
拡張	小	☺☺☺😐😐☹☹	☺☺☺
	大	☺☺☺😐😐☹☹	☺☺☺

多層は設計に
時間がかかる

多層は製造にも
時間がかかる

多層はテストが楽

多層は移行が楽

多層は
拡張しやすい

まとめ

- レイヤー構成の範囲を明確にすること
  - 利用するプラットフォームによって、利用できる技術が変わります。
  - 対応するクライアントによって、製作のしやすさが変わります
- データベースの構成はDBXのみ、もしくはFireDACにすること
  - DBXフレームワークを利用していれば、多層化対応のコンポーネントが多数揃っています。
  - 既存のプロジェクトの移行案件であれば（特にBDEからの移行案件であれば）、FireDACの構成へ変更するのが簡単です。
- リファクタリングはこまめに行うこと
  - RADの本来の機能を十分に使いましょう
  - Delphi 2005以降、リファクタリングのための機能が数多くIDEに統合されています
- 複数のバージョンがテストできる環境を用意すること
  - ここ数年で、仮想環境構築のコストは極端に下がっています
  - 実機よりも仮想環境！ 環境依存の問題を検証をするのにかかる人件費よりも安いです

Thank you!

メルマガもご登録ください(※期間限定)

↓↓↓

http://www.dreamhive.co.jp/25thdc/

パスワード：25thdc

embarcadero Developer Camp