

.NET用アプリケーション・フレームワーク Open棟梁

2014年4月23日

はじめに

- Open棟梁の母体は、2007年に日立ソリューションズ（旧 日立システムアンドサービス）で開発されました。
- これまでの7年間で、当社の多数のプロジェクト（受託開発やプロダクト開発、小規模部門システムから大規模基幹システムまで）で採用されてきました。
- 品質・信頼性が高いシステムを実現するには、実績に裏打ちされた開発基盤の活用が効果的です。棟梁には、当社が.NETアプリケーション開発で長年蓄積してきたノウハウが集約されています。

IT技術の変化は激しく、近年は技術の選択肢が増え、複雑化の一途です。一方で、企業の成長にとってIT活用の重要性はより増しています。この度、日立ソリューションズは、これまで自社利用ソフトであった.NET向けのフレームワークを、Open棟梁という名称でオープンソースとして公開することにしました。.NET技術を今後も利用される組織やユーザが、棟梁を使うことにより、企業競争力や技術力が向上することに少しでも貢献できれば幸いです。

Contents

1. 概要

2. 特長

3. 通信制御機能

4. D層自動生成ツール機能

5. 動的パラメタライズド・クエリ機能

6. リッチクライアント機能

Contents

1. 概要

2. 特長

3. 通信制御機能

4. D層自動生成ツール機能

5. 動的パラメタライズド・クエリ機能

6. リッチクライアント機能

1.1 概要

Open棟梁は、.NET Framework 2.0 以上を前提とした、フルスタックのアプリケーション・フレームワークです。

Web(ASP.NET, ASP.NET MVC)、C/S(WinForm, WPF)、
バッチ、RIA(Silverlight)、WebAPI、組込み等の各種方式に
対応し、高品質なアプリケーション開発を可能にします。

■ ドキュメント

- ◆ 利用ガイド、チュートリアル
- ◆ オフショア開発のための英語版マニュアル

■ ライセンス

- ◆ source : Apache License, Version 2.0
- ◆ document : Creative Commons - CC BY 2.1 JP

1.2 前提環境

		製品名
開発環境		<ul style="list-style-type: none">• Microsoft Visual Studio 2010 – 2013• Microsoft Visual C# 2010 – 2013• Microsoft Visual Basic 2010 – 2013
実行環境	Run Time	<ul style="list-style-type: none">• .NET Framework 2.0 – 4.5.1• ASP.NET2.0、4.0(+ AJAX Extensions)• ASP.NET MVC 4• Windows Azure SDK for .NET• Silverlight、Windowsストアアプリ
	Data Provider	<ul style="list-style-type: none">• .NET Framework Data Provider for SQL Server• OLEDB.NET Data Provider• ODBC.NET Data Provider• Oracle Data Provider for .NET• IBM DB2.NET Data Provider• HiRDB.NET データ プロバイダ• MySQL Connector/.NET• PostgreSQL Npgsql.NET データプロバイダ
	Browser	<ul style="list-style-type: none">• Internet Explorer Version 6.0–11.0

Contents

1. 概要

2. 特長

3. 通信制御機能

4. D層自動生成ツール機能

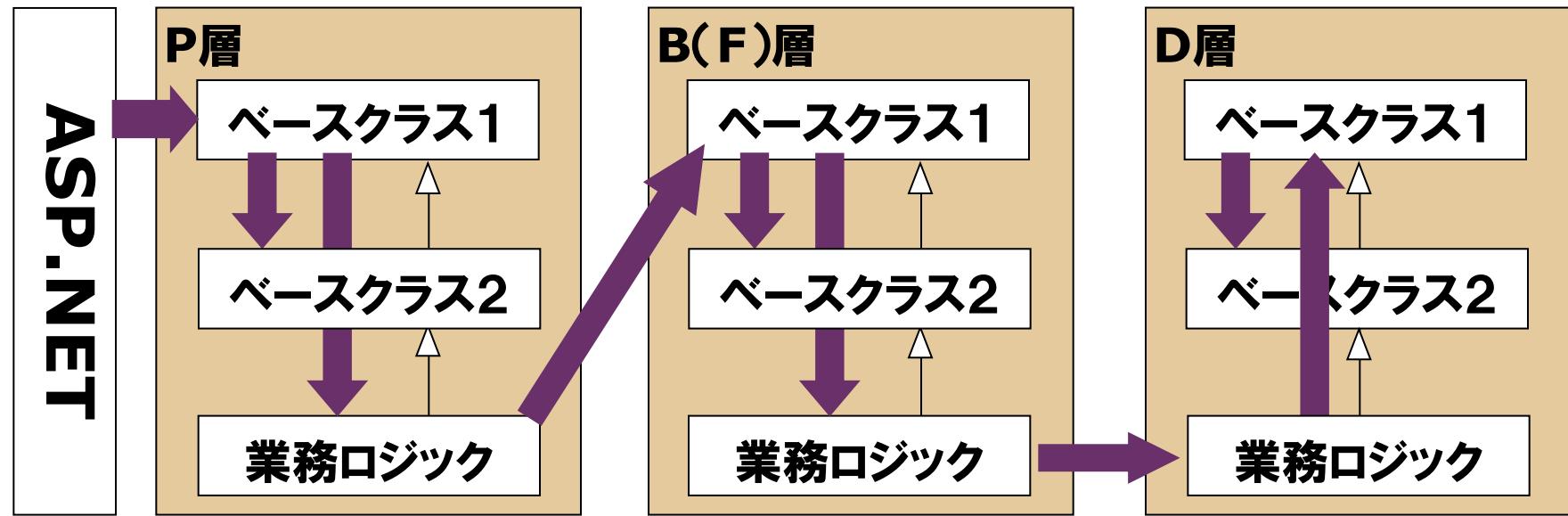
5. 動的パラメタライズド・クエリ機能

6. リッチクライアント機能

2.1 汎用性とカスタマイズ性を両立させた3層アーキテクチャ

各層の処理フローを規程したアーキテクチャを提供

2段階のベースクラスにより、汎用性とカスタマイズ性を両立



ベースクラス1(固定)とベースクラス2(カスタマイズ化)で使う共通機能を提供

認証	セッション管理	データアクセス	トランザクション	例外処理
権限	メッセージ取得	入力チェック	子画面表示	セキュリティ

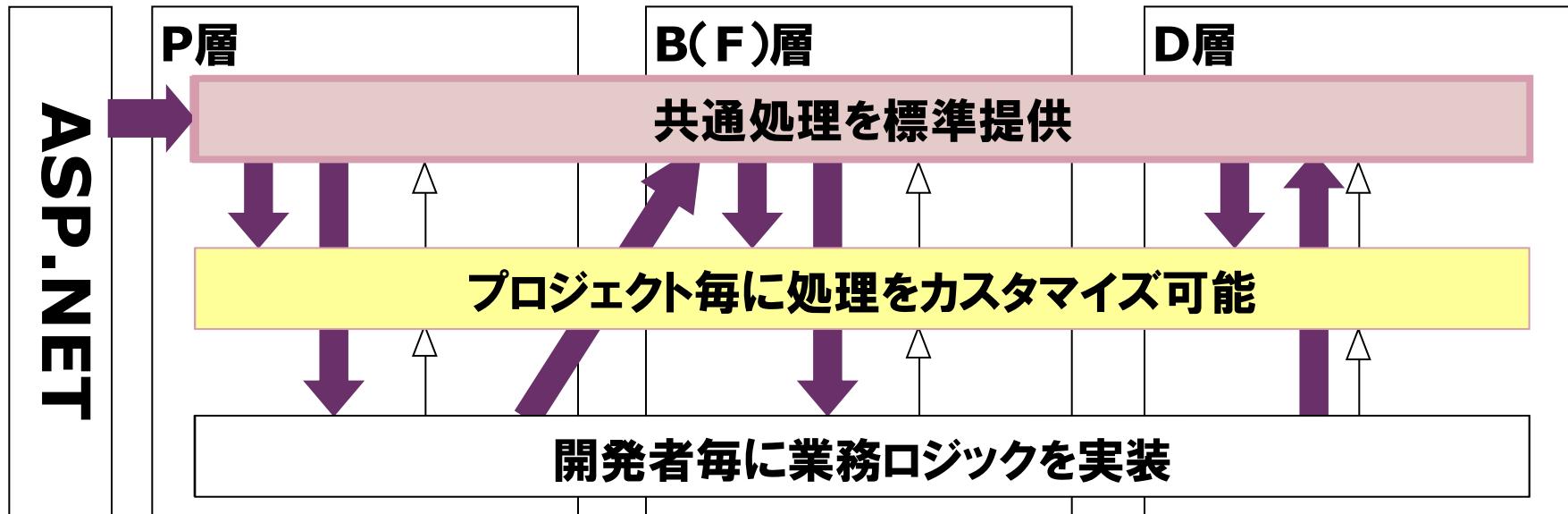
2.2 各レイヤの機能

層	機能
P 層	<p>Visual Studioのデザイナの操作性をス poイルしない造り。</p> <ul style="list-style-type: none">◆ 画面遷移制御、セッション管理◆ 不正操作防止、入力チェック.etc
B 層	<ul style="list-style-type: none">◆ フロー制御<ul style="list-style-type: none">下記の基本処理(カスタマイズ可能)のフローを制御<ul style="list-style-type: none">■ 開始終了処理、例外処理■ DBコネクション管理、トランザクション管理■ ログ出力、性能測定.etc◆ 通信制御機能<ul style="list-style-type: none">各種プロトコルを用いたサーバ間のリモート処理機構を提供。これによりオンプレやクラウドを含む多彩なシステム構成に対応可能。
D 層	<ul style="list-style-type: none">◆ MyBatisライクなデータアクセス・ライブラリ<ul style="list-style-type: none">■ 動的パラメタライズド・クエリ■ 動的パラメタライズド・クエリ定義・検証ツール◆ バッチ処理用SQL生成部品.etc

2.3 部品、ツール

区分	機能
ライブラリ群 共通	<ul style="list-style-type: none">◆ 文字列編集、入力チェック機能◆ ログ出力、共有情報・メッセージ管理◆ 国際化対応<ul style="list-style-type: none">■ ローカル時刻 ⇄ UTC変換、■ 各種メッセージリソースのCultureInfo対応◆ 非同期イベント制御、共有メモリ管理
コントロール カスタム	<ul style="list-style-type: none">◆ WebFormとWindowsFormsに対応◆ 文字列編集、入力チェック機能、Grid内表示
ツール 自動生成	<ul style="list-style-type: none">◆ テーブルCRUDのD層自動生成ツール◆ テーブル・メンテナンス画面の自動生成ツール

2.4 “棟梁”による標準化の促進



“棟梁”によるアプリケーション アーキテクチャの標準化

1. P / B / D層に渡る、全レイヤの標準化が可能です。
2. 基盤処理の実装がベースクラス1、2に分割されます。
 - ・ベースクラス1：共通処理(実行エンジン)
 - ・ベースクラス2：プロジェクト毎にカスタム
3. これにより、開発者は、サブクラスへの業務ロジック実装に専念することができます。

2.5 プロジェクト・テンプレート

特定プロジェクトのアーキテクチャに合わせてカスタマイズされたオンライン処理やバッチ処理のテンプレートを『プロジェクト・テンプレート』と呼びます。

このプロジェクト・テンプレートを事前に準備し、プロジェクトに展開することで、開発プロジェクトの迅速な立ち上げを可能にします。

この準備作業を容易にする『テンプレート・ベース』を公開しています。活用方法は、『Tutorial_Template_development.doc』参照して下さい。

案件毎のアーキテクチャを反映した
『プロジェクト・テンプレート』

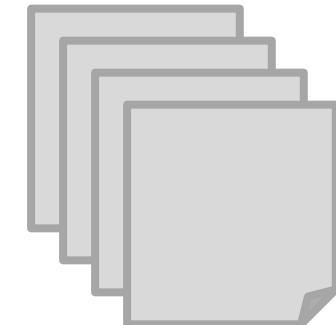
カスタマイズ可能レイヤのカスタマイズ

カスタマイズ可能な標準化フレームワーク
(P / F / D 層 の構造と実装箇所の規定)

共通部品
(通信制御、動的パラメタライズド・クエリ)

ランタイム フレームワーク
(e.g. ASP.NET, WPF, WCF, ADO.NET)

ランタイム (.NET CLR)



各種ドキュメント

- 利用ガイド
- チュートリアル
- サンプル.etc

2.6 各方式対応 – Silverlight・ストアアプリ

Silverlight・ストアアプリをUIに選定した場合も、“棟梁”の提供する実績ある処理方式で業務アプリケーションを開発可能です。
また、.NET ⇄ 非.NETシステム間の相互運用も実現可能です。

Silverlight ストアアプリ



REST or SOAP

TEXT

TEXT

- ・画面上にDTOをBind
- ・画面上での一覧編集内容をDTOが保持

.NET プログラム



DTO



- ・一覧取得
- ・編集内容取得と一括更新

Windows Forms / WPF



- ・DTOとDataTableの相互変換を実現
- ・画面上での一覧編集内容をDTOに変換

非.NET(Javaなど)(*)



DTO

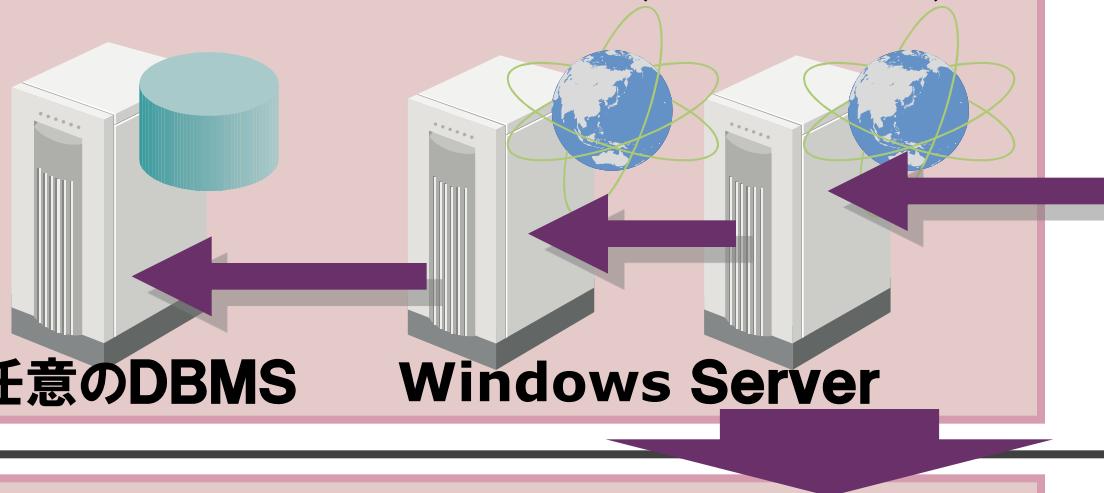


- ・一覧取得
- ・編集内容取得と一括更新

(*) 非.NET側に、同様のDTO部品の作成が必要です。

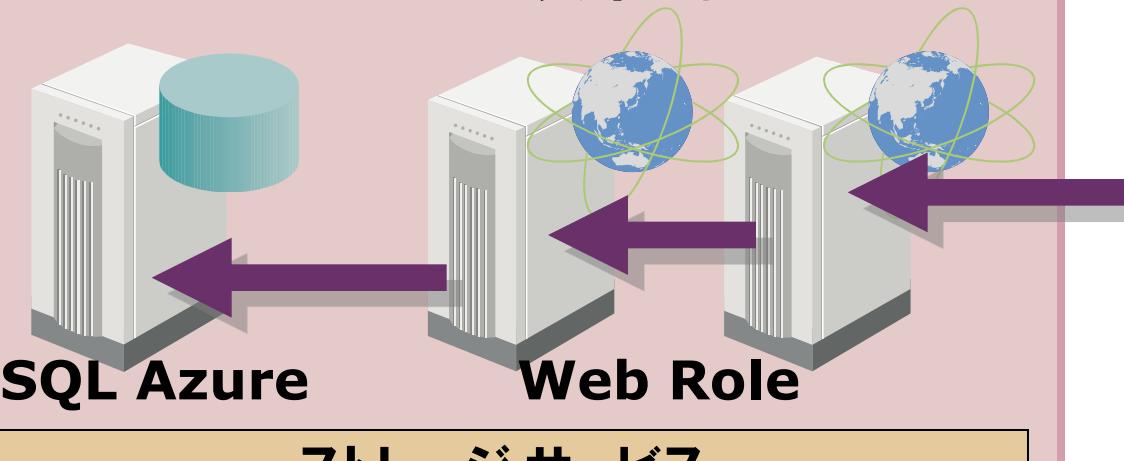
2.7 各方式対応 – Windows Azure (PaaS)

Windowsプラットフォーム（オンプレミス）



設定ファイルの
変更のみで対応可能

Windows Azure プラットフォーム



Contents

1. 概要

2. 特長

3. 通信制御機能

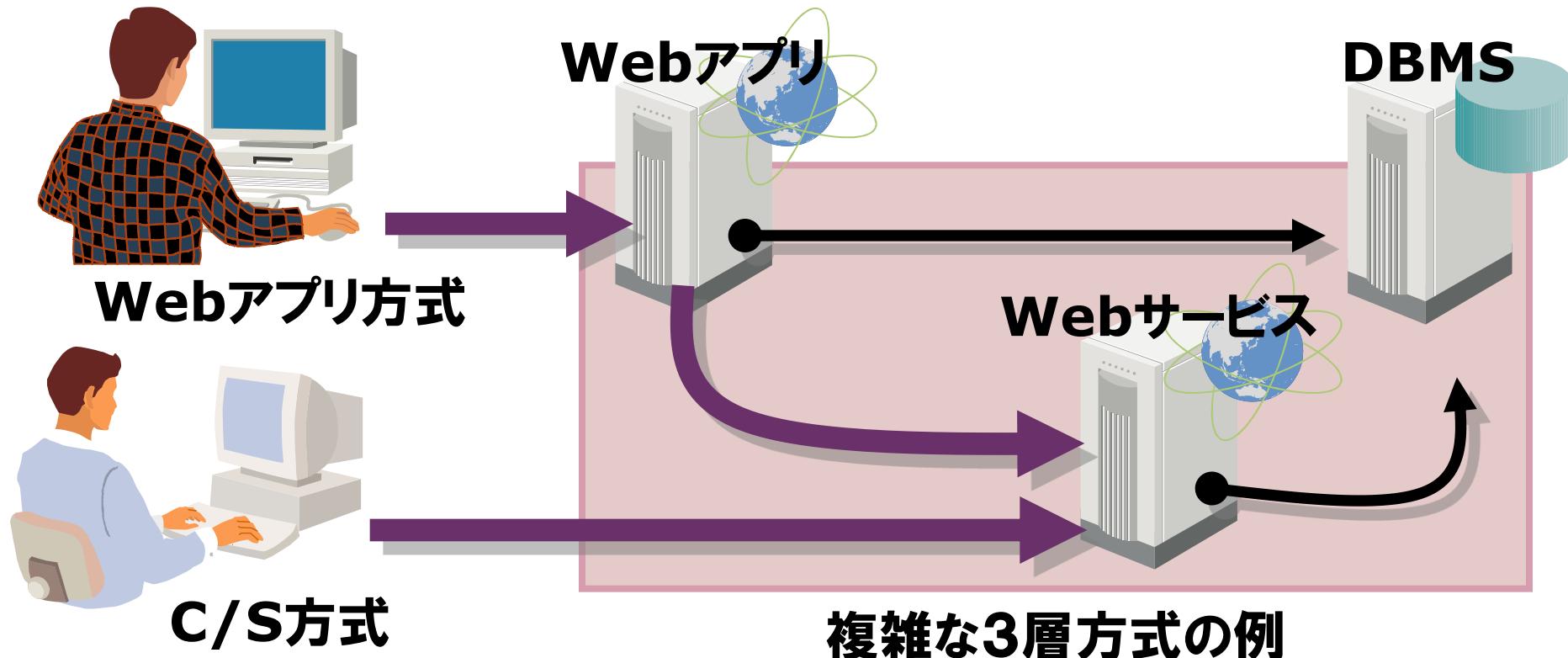
4. D層自動生成ツール機能

5. 動的パラメタライズド・クエリ機能

6. リッチ・クライアント機能

3.1 通信制御機能 – 概要

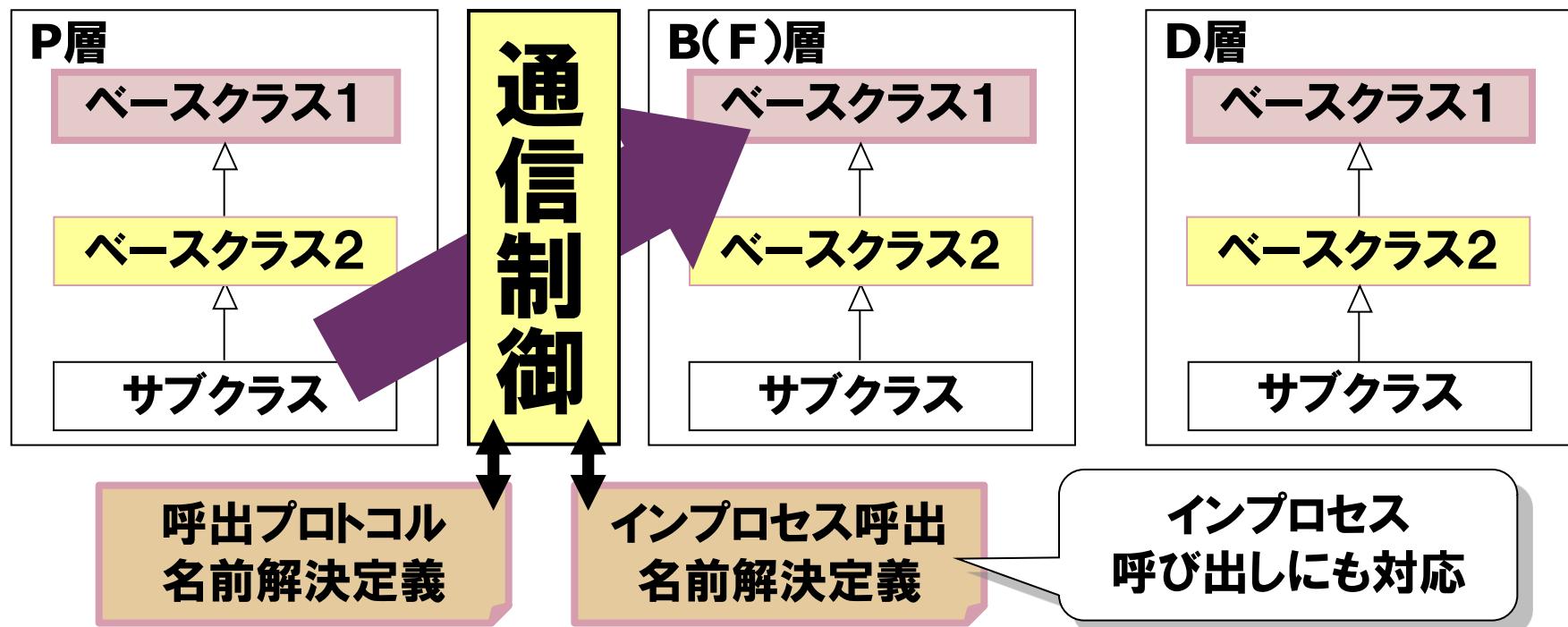
通信制御機能は、複雑な3層方式の開発を容易にします



通信制御機能が面倒な通信処理を隠蔽することで、開発者は業務ロジックの実装に専念できます。

3.2 通信制御機能 – アドイン可能な構造

通信制御機能は、既存の開発にアドインとして追加できます。
分散オブジェクト的な機能をWebサービス系のプロトコルを使用して実現します。

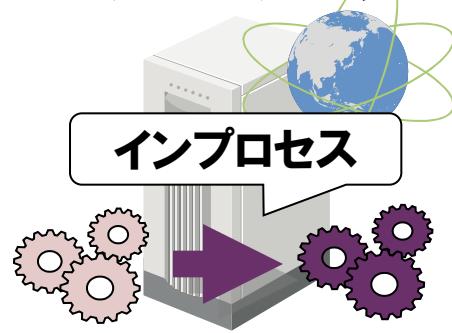


3.3 通信制御機能 – スケーラビリティの向上

クライアント アプリ



サーバ アプリ



サービス ゲートウェイ
/インターフェイス部品

B/D層

・位置透過性

定義によるインプロセス
/NW経由呼出の切替、呼出
先WASの変更を実現

・規模透過性

スケールアウト(垂直、水平
分散)を実現

・異種透過性

.NET以外の異種開発技術
との連携も可能

Webサービス化

配置先の変更

F/Wを超えも可能

Webサービス

インプロセス

Webサービス

インプロセス

Contents

1. 概要

2. 特長

3. 通信制御機能

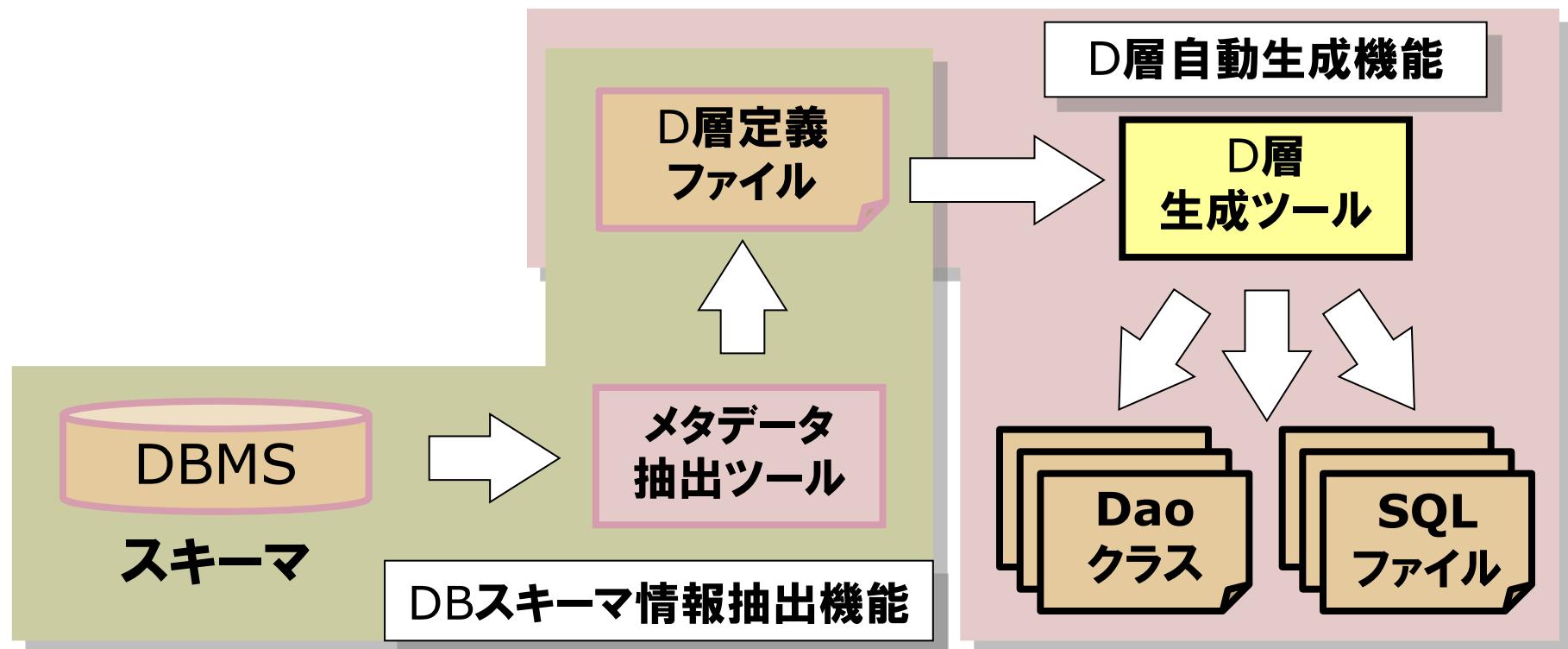
4. D層自動生成ツール機能

5. 動的パラメタライズド・クエリ機能

6. リッチクライアント機能

4. D層自動生成ツール – 自動生成

スキーマ情報から、テーブル・ビューに対する、CRUD処理のDao/SQLを100%自動生成します。生成された処理は、後述の動的パラメタライズド・クエリを活用しています。また、実装漏れを起こしやすいWebアプリケーションのタイムスタンプ楽観排他処理も生成されます。



Contents

1. 概要

2. 特長

3. 通信制御機能

4. D層自動生成ツール機能

5. 動的パラメタライズド・クエリ機能

6. リッチクライアント機能

5.1 動的パラメタライズド・クエリ – これまでのSQL

文字列

今まででは、動的SQLを処理するプログラム中に文字列、
文字列連結、それに伴うIF文が散在していました。

```
' ****  
'* SQL文作成  
'* ****  
str_Sql = "SELECT COUNT(DISTINCT xxxx) AS xxxx FROM xxxx"  
  
If intXXXXX = 0 Then  
    str_Work = " WHERE xxxx = " & strXXXXX & " AND xxxx LIKE ' " & strXXXXX & "%'"  
ElseIf intXXXXX = 2 Then  
    str_Work = " WHERE xxxx = " & strXXXXX & " AND xxxx = ' " & strXXXXX & "'"  
ElseIf intXXXXX = 3 Then  
    str_Work = " WHERE xxxx = " & strXXXXX & " AND xxxx = ' " & strXXXXX & "' " &  
        " AND xxxx IN (SELECT xxxx FROM xxxx WHERE xxxx = " & strXXXXX & ")"  
Else  
    str_Work = " WHERE xxxx = " & strXXXXX & " AND xxxx LIKE ' " & strXXXXX & "%'" &  
        " AND xxxx IN (SELECT xxxx FROM xxxx WHERE xxxx = " & strXXXXX & ")"  
End If
```

文字列連結

IF文

動的SQLは、

- WHERE、JOIN句の付与・削除
 - AND、OR演算子の付与・削除
 - IN句のパラメタを条件数に合わせ用意
 - 脆弱性の問題を潜在的に内包する
- など、制御が面倒で実装が難しい。

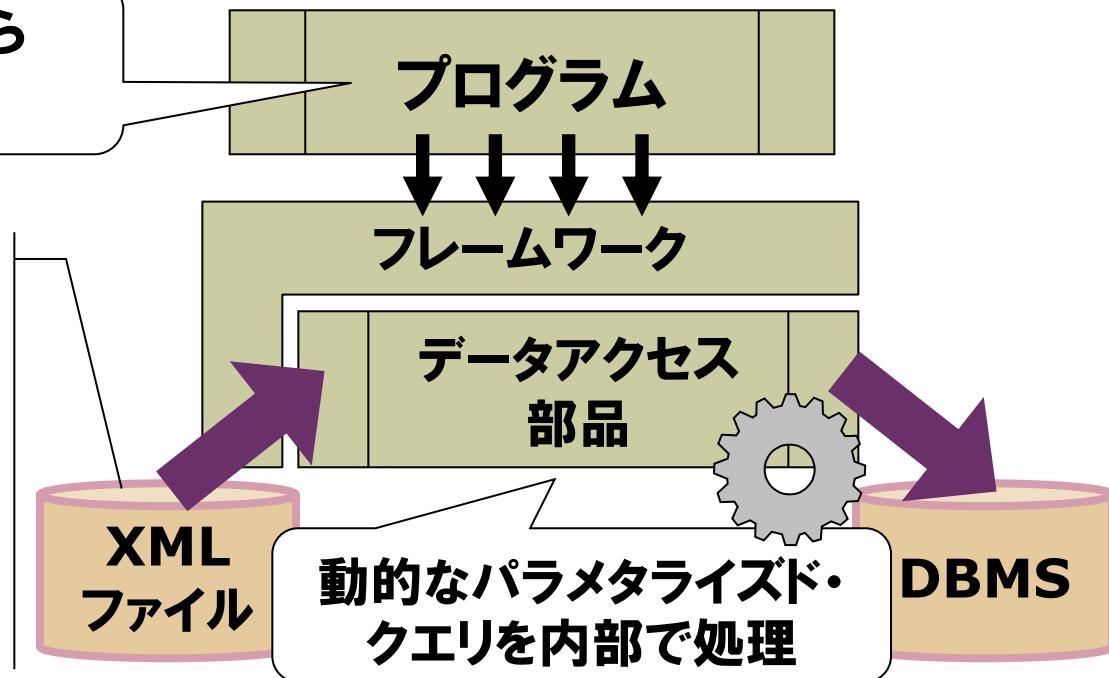


5.2 動的パラメタライズド・クエリ – 概要

動的パラメタライズド・クエリ機能により、文字列、文字列連結、それに伴うIF文がなくなり、プログラム実装が容易になります。

プログラムからは、APIから
パラメタを設定するだけ

```
<?xml version="1.0" encoding="shift_jis" ?>
<ROOT>
  SELECT
    A, B, C, D
  FROM T
  <WHERE>
    WHERE
      <IF>AND A = @A</IF>
      <IF>AND A LIKE @A_LIKE</IF>
      <IF>AND B = @B</IF>
      <IF>AND B LIKE @B_LIKE</IF>
      <IF>AND C = @C</IF>
      <IF>AND C LIKE @C_LIKE</IF>
      <IF>AND D = @D</IF>
    </WHERE>
  ORDER BY <IF name="SEQUENCE">A<ELSE>B</ELSE></IF>
</ROOT>
```



パラメタ設定だけで動的なパラメタライズド・クエリを実行でき、
開発者は、複雑な文字列連結処理の制御から解放されます。
また、XMLでの記述が不要な静的パラメタライズド・クエリもサポート。

Contents

1. 概要

2. 特長

3. 通信制御機能

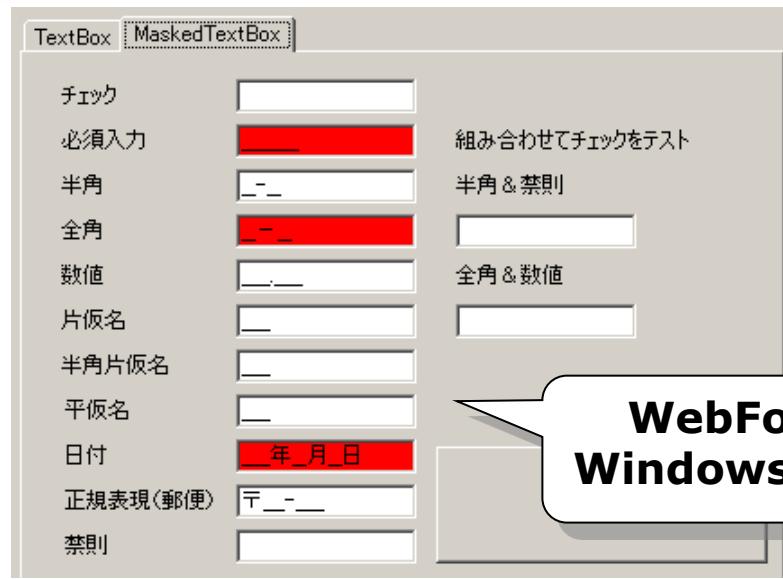
4. D層自動生成ツール機能

5. 動的パラメタライズド・クエリ機能

6. リッチクライアント機能

6.1 カスタムコントロール & バリデーション

カスタム コントロール(Web Form, Windows Forms)や、バリデーション フレームワーク(WPF)を使用して単項目チェックの実装を容易に。また、VSデザイナやXAMLから属性ベースでチェック条件を選択することが可能。

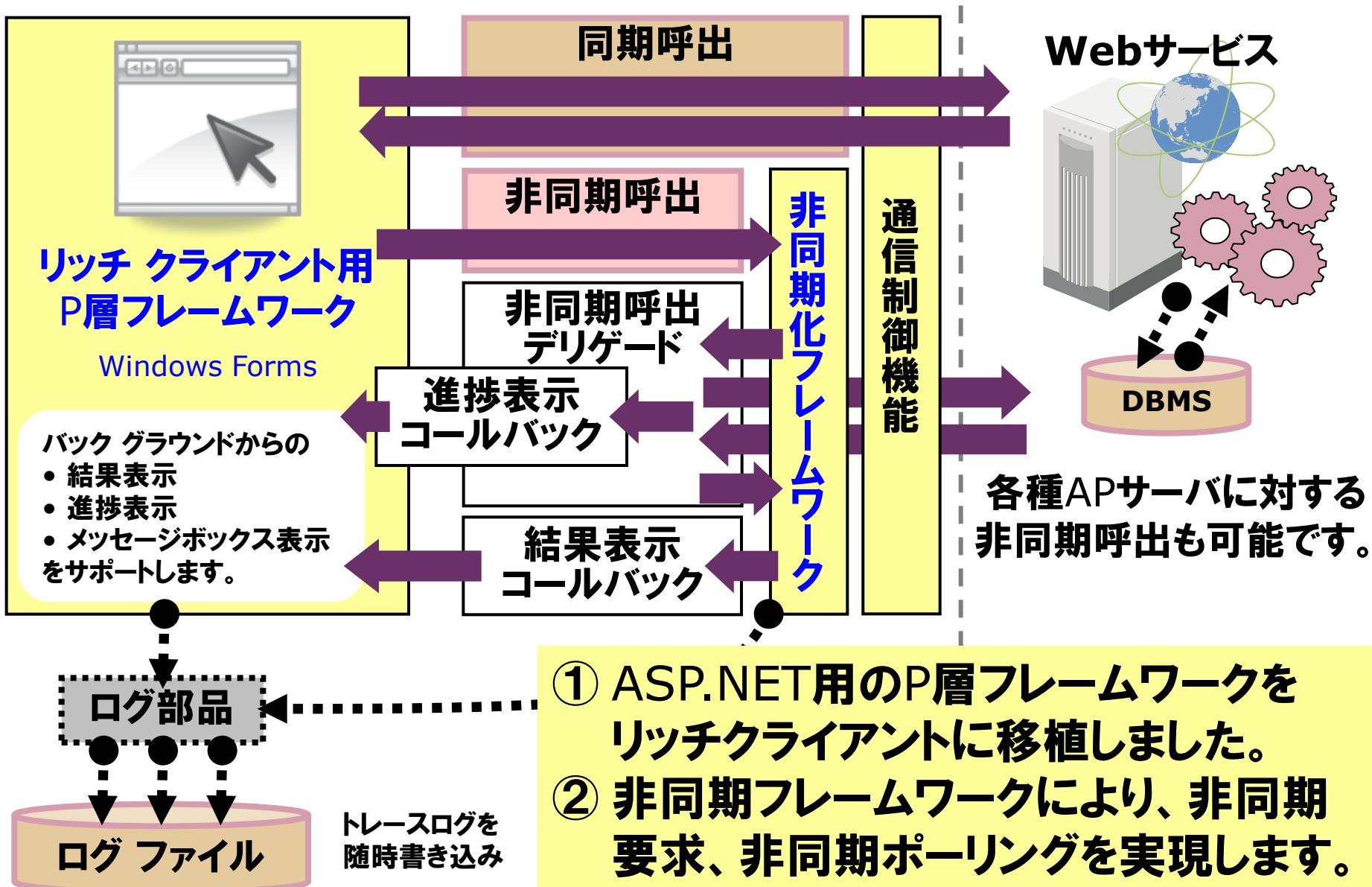


CheckProhibitedChar	False
CheckRegExp	
CheckType	IsDate
IsDate	True
IsHankaku	False
IsHanKatakana	False
IsHiragana	False
IsIndispensable	False
IsKatakana	False
IsNumeric	False
IsZenkaku	False



```
<Label Height="23">カスタムのValidationRule (正の整数 + 最大値設定可:200)
<TextBox Height="23" Margin="5" Name="textBox7"
    Validation.Error="textBox7_Error">
    <Binding Path="SourceProperty7"
        UpdateSourceTrigger="LostFocus"
        ValidatesOnDataErrors="True" />
    <Binding.ValidationRules>
        <!-- カスタムのValidationRule (最大値:200) -->
        <my:MyValidationRule Max="200"/>
    </Binding.ValidationRules>
</TextBox>
```

6.2 リッチクライアント対応 P層フレームワーク



6.3 リッチクライアントWebデプロイツール

