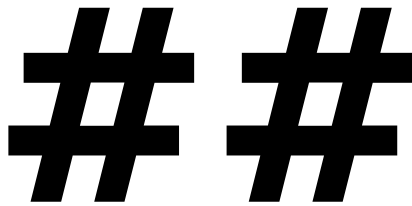


.NET

# .NET Webアプリケーション アーキテクチャパターン



森屋 英治

有限会社アークウェイ  
代表取締役

[www.archway.co.jp](http://www.archway.co.jp)

デブサミ2005  
**Developers Summit**

.NET

2

## Agenda

- 自己紹介 & 目的
- ASP.NETアーキテクチャパターン
- データセットインターナル
- Q&A

Developers Summit  
デブサミ2005

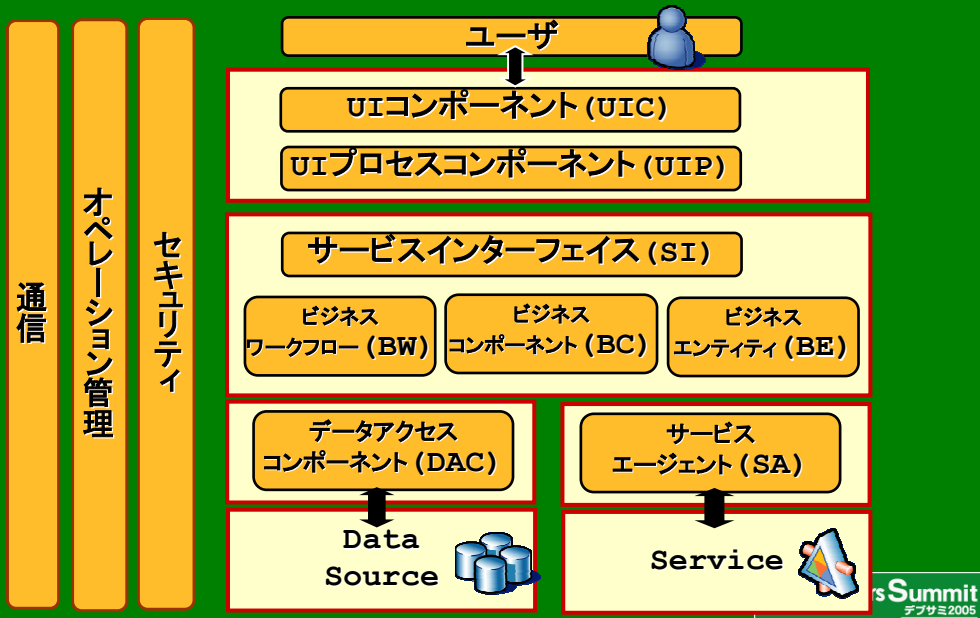
## セッションテーマ

---

- マイクロソフトから提供されている、3つのコマーンスサイトのサンプルの実装パターンを通して、ASP.NETの実開発者、あるいはチームにあったデザインの構成を再考する。
- ASP.NETの開発にもっとも利用するDataSet、参照系の利用は、非常に容易。今回は、やや複雑なデータ更新時の動作を中心に解説し、更新処理の内部構造をより深く理解する。

## ASP.NET実装パターン

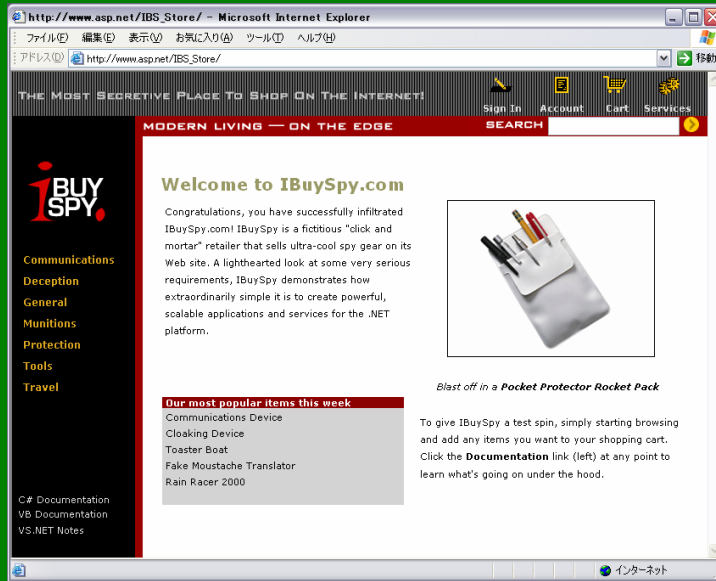
# Application Architecture for .NET



## 3つのコマースサンプルサイト

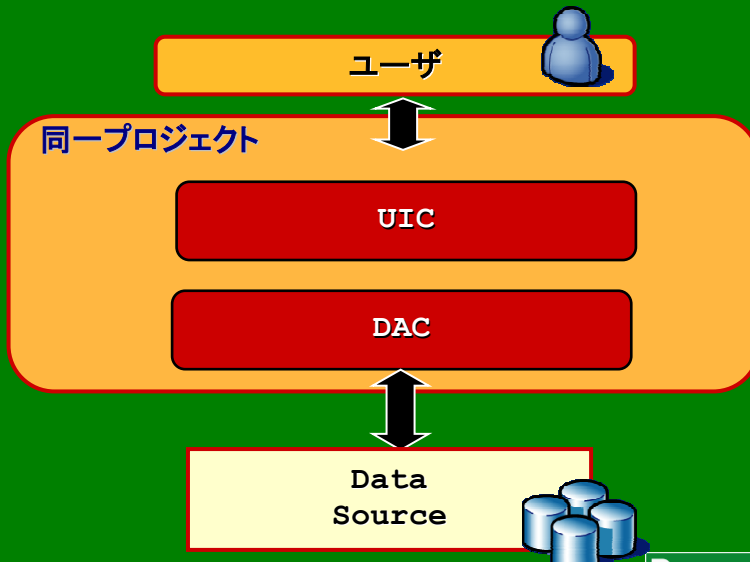
- マイクロソフトから適用されている3つのコマースサイトサンプルを例に構成を考える。
  - Commerce Starter Kits (IBuySpy)
  - Duwamish 7.0 (Book Store)
  - Pet Store 3.0 (Petshop)
- Application Architecture for .NETが作られる以前のサンプルなので、構成例として名前が統一されていない。

# IBuySpy Commerce



Developers Summit  
デブサミ2005

# Commerce Starter Kit構成



Developers Summit  
デブサミ2005

# IBuySpy.com

- 特徴
  - スパイ用品のCOMマースサイト
  - VB.NET, C#, J# & VS.NET, WebMatrix
  - シングルプロジェクト
  - UI, DataAccessの構成
  - フリーのASP.NETサイトでもサンプルとしてIBuySpyを利用可能
- UI
  - フォーム認証を利用(データは、データベース)
  - メニューなどは、UserControlを利用
  - バasketは、データベースを利用
- DataAccess
  - 接続文字列は、DBクラスでWeb.Configから取得
  - ストアドプロシージャを利用
  - パスワードの暗号化

Developers Summit  
デブサミ2005

# Duwmamish 7.0

Duwmamish 7.0 ホーム ページ - Microsoft Internet Explorer

アドレス http://localhost/Duwmamish7/default.aspx

Duwmamish7.0 ホーム ショッピング カート アカウント

検索

名前

実行

カテゴリの参照

ASP/ASP.NET  
COM/DCOM  
Exchange  
HTML/XML  
SQL  
Visual Basic  
Visual C#  
Visual C++  
Windows  
その他  
プログラミング言語一般  
ライブラリ リファレンス  
設計

シーンの背景

画像とソース

このサンプルに含まれている架空の財務データは静的なものであり、必ずしも現実の世界の値名と価格を反映しているわけではありません。

本日のお勧め

**Microsoft Exchange 2000 Server リソースキット 上 導入ガイド**

本書は、Exchange 2000 Serverを企業に導入する管理者向けの解説書です。Active DirectoryとExchange 2000を導入するうえでのさまざまな情報を提供します。企業の要件、組織形態、地理的な要件を考慮したActive DirectoryとExchange絡みの設計、パイロット計画やテスト環境の構築を含む導入計画、信頼性とセキュリティを実現するネットワーク環境の構築、およびバックアップと復元の計画について説明しています。このほか、チャットやインストールメモセージングなどのコラボレーション機能の導入についても解説しています。

当社の価格: ¥4,800

カートに追加

**C#プログラミングリファレンス**

「オブジェクト指向プログラミングとしてのC#」オブジェクト指向としてのC#という部分について、クラスについての詳細は第4章「クラスの定義と実装」、オブジェクト指向の観点に立った実践的なクラス設計のヒントは第5章「オブジェクト指向におけるクラスの活用」で詳解しています。

●Visual basic 6.0やC++との比較、VB6やC++を別にマスターしている方に対して、「for VB6」や「for C++」という「ワンポイント」コラムがあり、VB6やC++とC#との違いが説明されています。

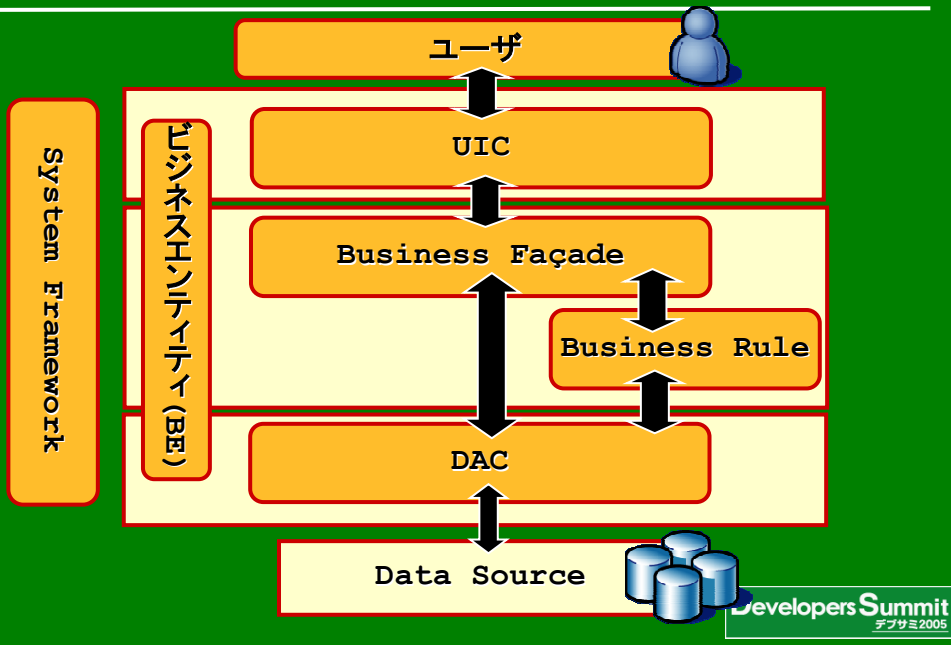
当社の価格: ¥3,000

カートに追加

Microsoft .net The Business Internet

Summit デブサミ2005

## Duwamish 7.0



## Duwamish 7.0

- 特徴

- 書籍のコマースサイト
- VB.NET, C#
- Visual Studioから作られるエンタープライズサンプルの典型的な例
  - ログイン
  - アサーション
  - 構成管理
- 6つのプロジェクト(エンタープライズテンプレートから作成)
- Web(ASP.NET)
- BusinessFacade, BusinessRules
- DataAccess
- Common, SystemFramework

## Duamish 7.0 各レイヤ

- UI(Web)
  - フォーム認証を利用(データは、データベース)
  - メニューなどは、UserControlを利用
- Business Façade(Service Interface)
  - ビジネスサービスへのエントリーポイント
  - 参照系の場合は、直接DACを呼び出す。
- Business Rule
  - ビジネスルールを構成
  - 正規表現、検証、税金の計算ロジックなど
  - DACを呼び出す
- Data Access Component
  - 接続文字列は、DBクラスでWeb.Configから取得
  - ストアードプロシージャを利用
  - トランザクションの実装 (IDbConnection.BeginTransaction())

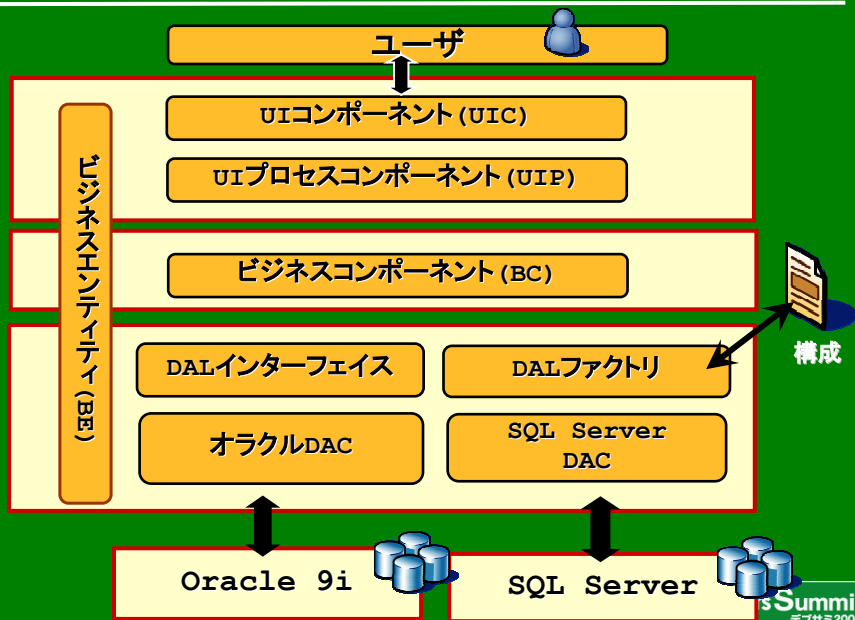
## Duamish 7.0 各レイヤ(続き)

- Business Entities(Common)
  - カスタムデータセット(パラメータとして[DTO])
  - Duamish固有の構成情報
- SystemFramework
  - 構成情報の構成管理
  - アサーション
  - ログ

# Microsoft .NET Pet Shop 3.x



# Microsoft .NET Pet Shop 3.x





## Microsoft .NET Pet Shop 3.x

---

- 特徴
  - ペット販売のCOMマースサイト
  - Java プループリント**Java Pet Store** の.NET版 (Oracle,DB2,Sybaseなどを対応)
  - 11プロジェクトで構成
  - PetShop3.xでは、Oracle,SQL Serverの対応(インターフェイスベース)
  - マルチデータベース対応の為、サービスコンポーネントを利用して分散トランザクションを実行
- User Interface Component(Web)
  - メニューなどは、UserControlを利用
- User Interface Processing
  - ユーザセッション情報の取り扱い
  - ページナビゲーション

## Microsoft .NET Pet Shop 3.x

---

- Business Component
  - ビジネスコンポーネント
  - OrderInsertのみサービスコンポーネント(分散トランザクション)を利用
  - OrderRead(参照系のみも用意)
  - インターフェイスベースでDACをコール
  - DALファクトリクラスを通してDACを生成。
  - ファクトリクラスは、構成情報を利用してOracle,SQL Serverを切り替え
- Business Entities
  - 各層をパッシングするカスタムデータクラス(シリアル化可能)
- Data Access Layer Components
  - Oracle 9i,SQLServer用のファクトリクラス
  - Oracle 9i,SQLServer用のインターフェイス
  - Oracle 9i,SQLServer用のデータアクセスクラス
  - Oracle 9i,SQLServer用DAABを利用

## サンプルはいつも最初の友達

- 初心者、中級者の方
  - Commerce Starter Kit
- 中,上級者の方
  - Duwamish7.0
  - Application Blocksのサンプル
- Javaと併用して開発されている方
  - PetShop3.0
  - Application Blocks(UI Navigation Blocks)

## アーキテクチャ、フレームワークの適用

- ターゲットアプリケーションの特性、チームスキル、パフォーマンス、開発規模に併せたデザインの選択



## ASP.NET実装のまとめ

---

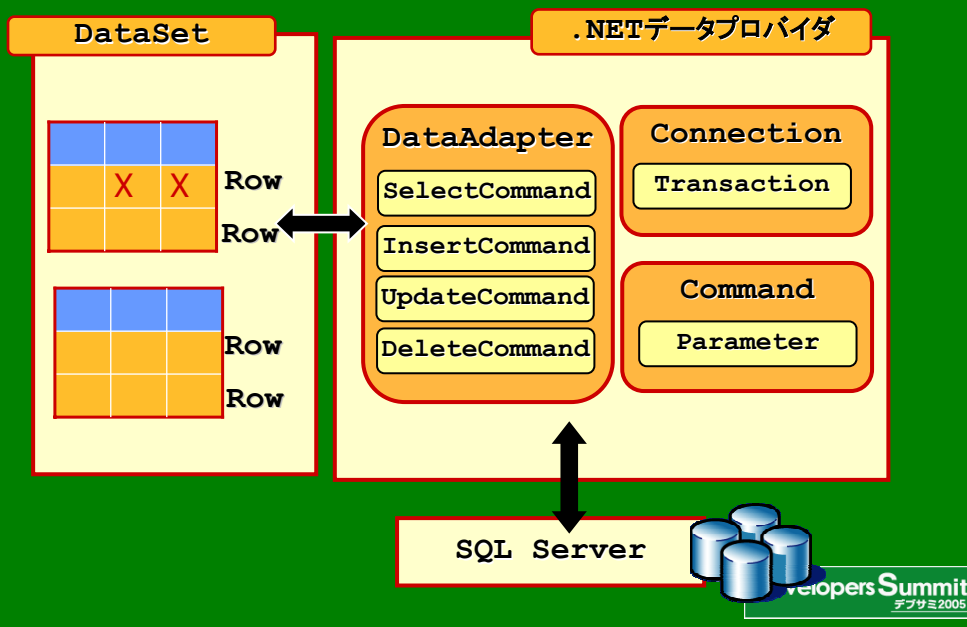
- 完璧なサンプルはない。
- ユーザニーズ、テクノロジー、ユーザXPは、多種多様。
- いつも“シンプル”なアーキテクチャを採用する。(非同期、分散トランザクション,WS-\*)(Architecture Design)
- 縦割りか、横割りのチームを構成(Team Model)
- 大多数の開発者のスキルセットを考慮する。
- OO,SQL,Script,HTML経験の構成比を考慮。
- トリッキーなコード、フレームワークを採用しない。(Simple)
- 信じれるものは、動作するアプリケーションとテストのみ。(TDD)
- テストがあるサンプルを作る。
- 認証、EAI、バッチ処理、トランザクションの利用は、いつも問題になる。スパイク！スパイク！
- 層別のシンプルなルールのリスト、アーキテクチャの図、テスト付きの共通サンプルの作成をお勧めします。

## データセットインターナル

## DataSet利用シナリオ

- ASP.NET(DataGrid等)でデータバインド(バインド可能)
- リードオンリーのキャッシュとして
- Webサービスの戻り値として(シリアル化可能)
  - スマートクライアント
- パラメータセットとして(DTOパターン)
  - MSアーキテクチャ(ビジネスエンティティ)
- データアクセスの結果セットのコンテナとして
- もっとインテリセンス(TypedDataSet)
- バッチ更新(オプティミスティックモデル)

## データセットと更新



## DataRowの変更パターン

- DataRowのItemプロパティを使って列の値を変更
    - 直接編集
  - 上記に加えて、DataRowオブジェクトのBeginEdit,EndEditメソッドの利用
    - いったん変更をバッファリングする
  - DataRowオブジェクトのItemArrayを利用
    - 配列ベースの変更
- ※ データセットでは、DataRowの変更を行っても、データベースが変更されるわけではありません。データベースの変更には、DataAdapterを利用します。

## DataRowの状態管理

| Row State | 説明                 |
|-----------|--------------------|
| Unchanged | 保留状態の変更内容が含まれていない  |
| Detached  | Data Tableに含まれていない |
| Added     | DataTableに行が追加された  |
| Modified  | 保留状態の変更内容が含まれる     |
| Deleted   | 行は削除保留中            |

## DataRowのバージョン管理

| Row Version | 説明                       |
|-------------|--------------------------|
| Current     | 列にある現在の値                 |
| Original    | 列に格納された元の値               |
| Proposed    | 列に提示中の値 (BeginEditでの編集中) |
| Default     | 既定の値                     |

バージョンの確認: `dataRow.hasVersion`

## 保留状態の変更内容の受入、拒否

- 受入 `AcceptChanges()`メソッド
  - `DataAdapter`が変更を正常終了した場合、内部的に`AcceptChanges`を呼び出します。
  - **Added** および **Modified** の行は **Unchanged** になり、**Deleted** の行は削除されます。
  - `DataSet`, `DataTable`, `DataRow`の3つレベルで用意。
- 拒否 `RejectChanges()`メソッド
  - 新しい行は、削除され、保留状態の変更内容を破棄し、`Row`の状態を **Unchanged**に設定。

## 保留中変更内容の確認

- 変更差分の取得
  - dataSet.GetChange()
  - rowの状態を選択可能(引数にrowstateを指定)
  - 差分のみのデータセットを取得可能
  - 変更の確認 bool dataSet.hasChanges()
  - スマートクライアントの利用シナリオで特に有効
- DiffGramの利用
  - XMLベースの変更情報を含むスキーマ
  - dataSet.WriteXml("diff.xml", XmlWriteMode.DiffGram);
  - 変更内容を含むXMLデータ

```

<?xml version="1.0" standalone="yes" ?>
<diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
  <NewDataSet>
    <Memo diffgr:id="Memo1" msdata:rowOrder="0" diffgr:hasChanges="modified">
      <ID>1</ID>
      <Title>キックオフ</Title>
      <Memo>仕事の開始</Memo>
      <LastAccess>2004-12-21T12:04:16.0000000+09:00</LastAccess>
    </Memo>
    <Memo diffgr:id="Memo2" msdata:rowOrder="1" diffgr:hasChanges="modified">
      <ID>2</ID>
      <Title>最初のミーティング</Title>
      <Memo>プロジェクトマネージャ</Memo>
      <LastAccess>2004-12-10T00:22:54.0000000+09:00</LastAccess>
    </Memo>
  </NewDataSet>
  <diffgr:before>
    <Memo diffgr:id="Memo1" msdata:rowOrder="0">
      <ID>1</ID>
      <Title>キックオフ</Title>
      <Memo>仕事の開始</Memo>
      <LastAccess>2004-12-06T14:50:16.2970000+09:00</LastAccess>
    </Memo>
    <Memo diffgr:id="Memo2" msdata:rowOrder="1">
      <ID>2</ID>
      <Title>最初のミーティング</Title>
      <Memo>プロジェクトマネージャ</Memo>
      <LastAccess>2004-12-06T14:50:46.6500000+09:00</LastAccess>
    </Memo>
  </diffgr:before>
</diffgr:diffgram>
  
```

## 保留中変更内容のデータベース反映

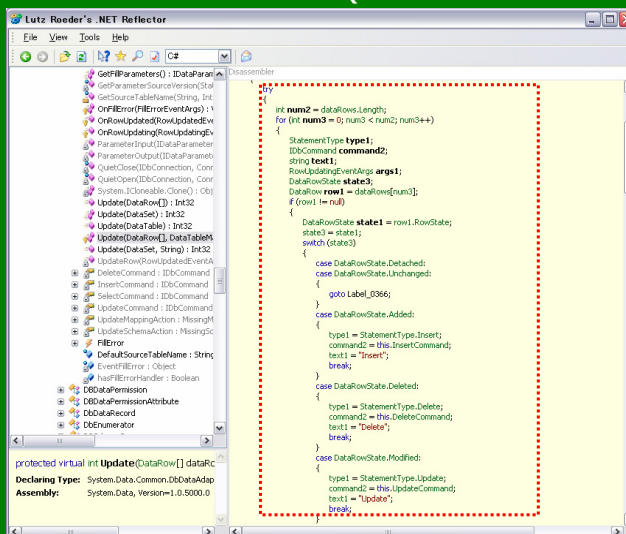
- データアダプタを利用したデータベースへの変更の反映

```
DataSet ds = new DataSet();
cnn.Open();
adapter.Fill(ds,"aTable");
//データセットの変更
adapter.Update(ds,"aTable");
cnn.Close();
```

← 変更をDiffGramで確認

## Rowの状態とAdapterのコマンドのマップ

- Reflectorを利用した解析(デモ)





## Rowの状態とAdapterのコマンドのマップ

- DbDataAdapter::Updateメソッドフロー
- 変更のあるRowの配列をループ
  - RowStateの属性とAdapterコマンドの種類にマップ
    - **Detached, Unchanged** 次のRowへ
    - **Added** InsertCommandをマップ
    - **Deleted** DeleteCommandをマップ
    - **Modified** UpdateCommandをマップ
  - コマンドの実行
  - 成功した場合  
row.AcceptChanges()

## データセットを利用した更新

- 実際の内部処理を理解して有効活用しよう。
  - Rowの状態、バージョンの管理
  - 変更内容の確認
  - 更新のメカニズム
- 同時実行制御の実装が課題
  - データソース内のデータは、ユーザが読み取る時にロックされない。そのため他のユーザは、元のユーザが読み取った後に行を読み取り、更新することが可能
  - データアダプタ構成ウィザードでの解決
    - DataRowのバージョンを利用して、Where句を利用して値を比較
  - 主キーとタイムスタンプタイムスタンプ列の使用

# Q & A