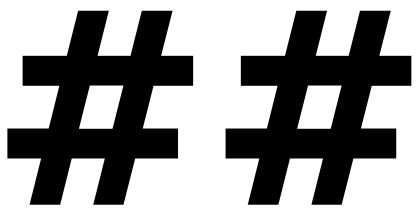
.NET

.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

デブサミ2005 Developers Summit

.NET

Agenda

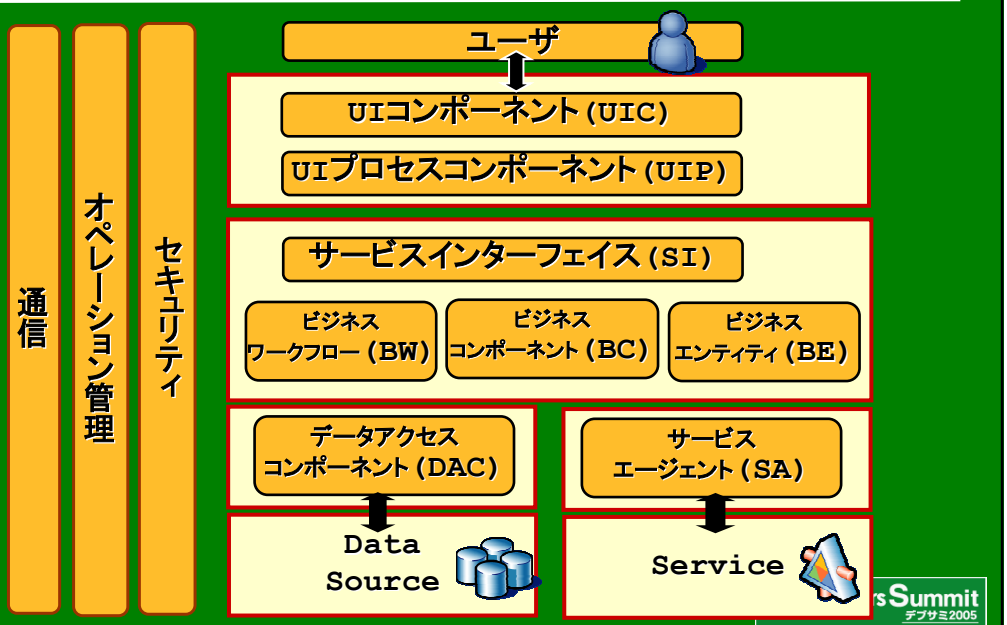
- 自己紹介＆目的
- ASP.NETアーキテクチャパターン
- データセットインターナル
- Q&A

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

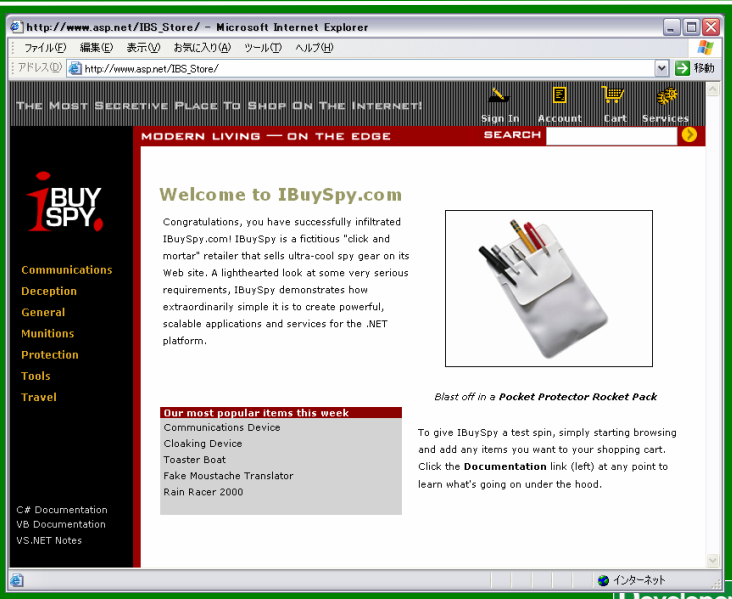
Application Architecture for .NET



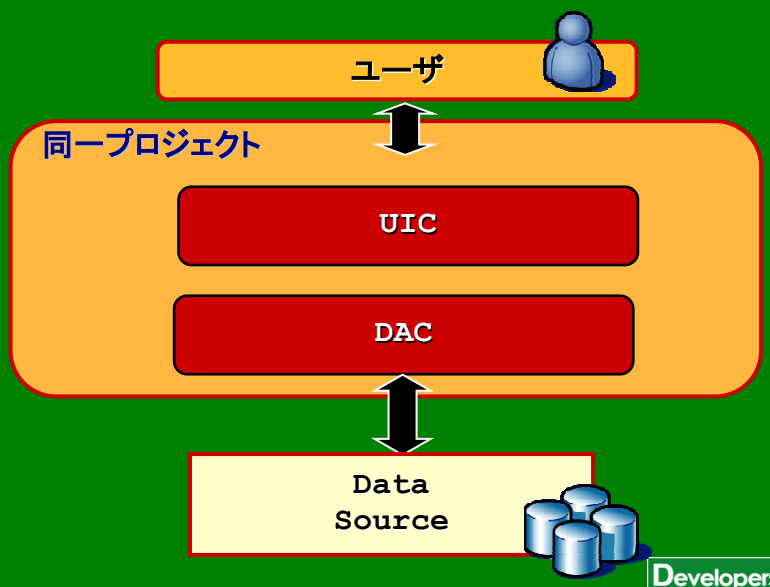
3つのコマースサンプルサイト

- 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)
  - Duwamish7.0(Book Store)
  - Pet Store3.0(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



Commerce Starter Kit構成

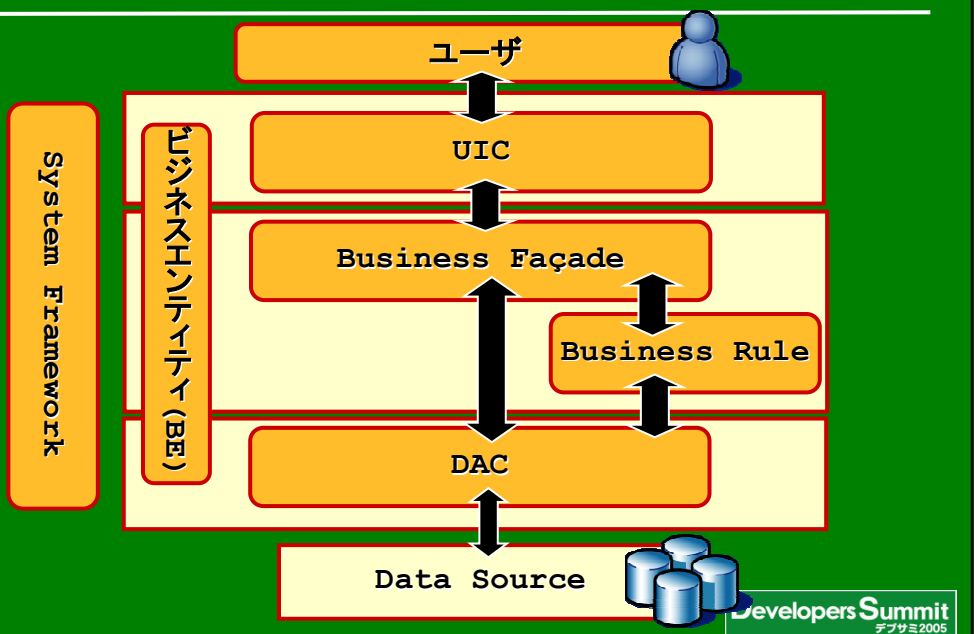


IBuySpy.com

- 特徴
  - スパイ用品のコマースサイト
  - 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
  - バスケットは、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
  - ストアードプロシージャを利用
  - パスワードの暗号化

Duwamish 7.0

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

    ロギング
    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

Duwamish 7.0 各レイヤ

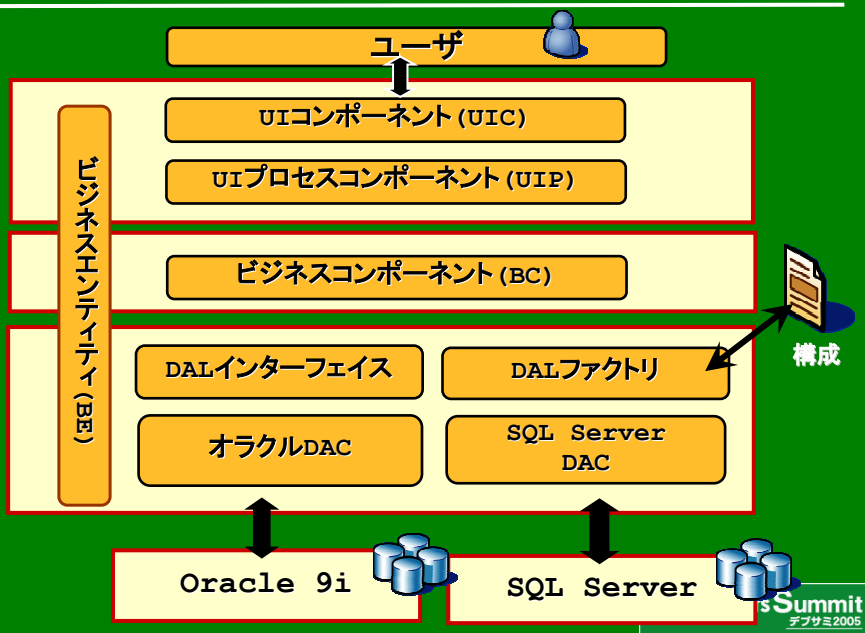
- UI(Web)
  - フォーム認証を利用（データは、データベース）
  - メニューなどは、UserControlを利用
- Business Façade(Service Interface)
  - ビジネスサービスへのエントリーポイント
  - 参照系の場合は、直接DACを呼び出す。
- Business Rule
  - ビジネスルールを構成
  - 正規表現、検証、税金の計算ロジックなど
  - DACを呼び出す
- Data Access Component
  - 接続文字列は、DBクラスでWeb.Configから取得
  - ストアードプロシージャを利用
  - トランザ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

Duwamish 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]）
  - Duwamish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
  - ペット販売のコマースサイト
  - Java ブループリント**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）
  - インターフェースベースで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Oralce,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用のインターフェース
  - 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

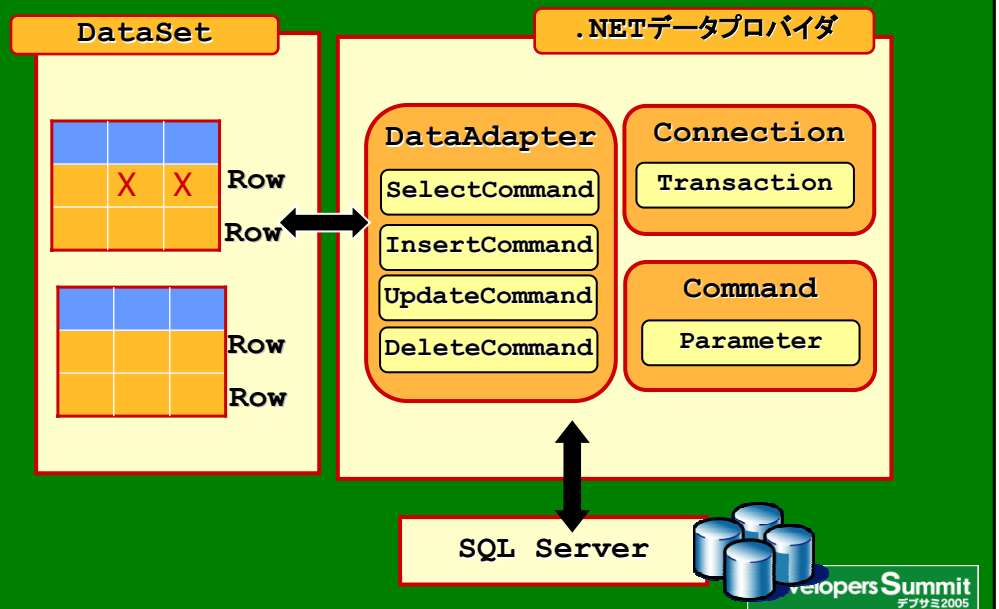
- 完璧なサンプルはない。
- ユーザニーズ、テクノロジー、ユーザXPは、多種多様。
- いつも“シンプル”なアーキテクチャを採用する。(非同期、分散トランザクション,WS-*)(Architecture Design)
- 縦割りか、横割りのチームを構成(Team Model)
- 大多数の開発者のスキルセットを考慮する。
- OO,SQL,Script,HTML経験の構成比を考慮。
- トリッキーなコード、フレームワークを採用しない。(Simple)
- 信じれるものは、動作するアプリケーションとテストのみ。(TDD)
- テストがあるサンプルを作る。
- 認証、EAI、バッチ処理、トラン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
- パラメターセットとして(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

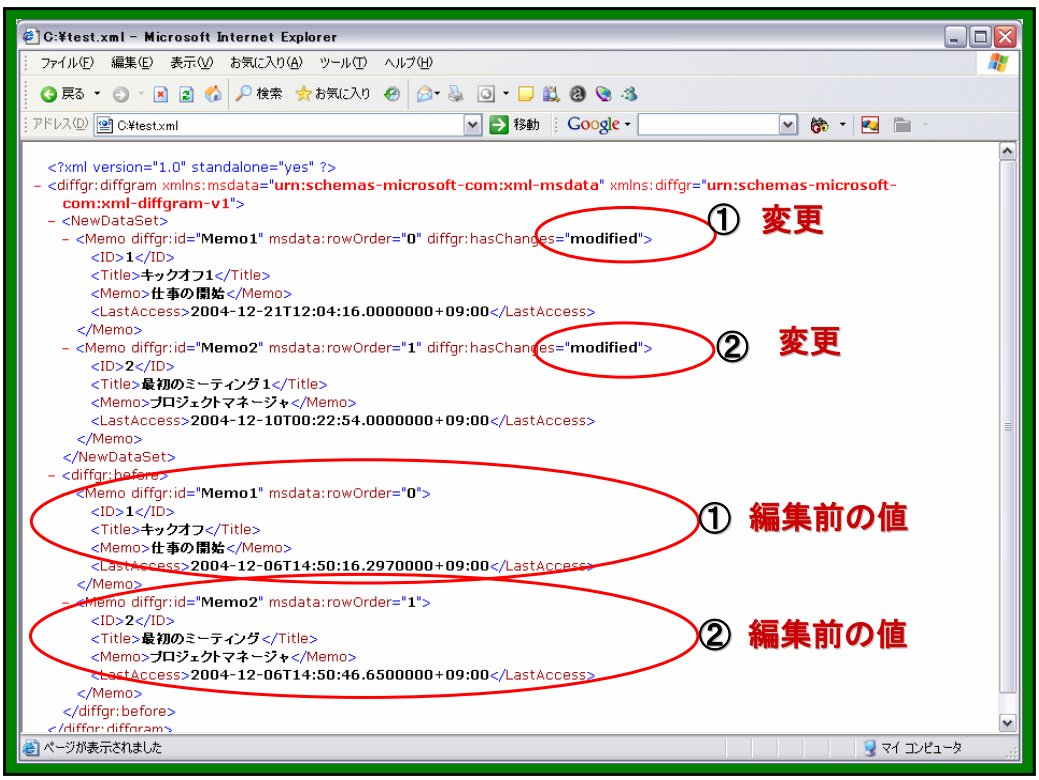
バージョンの確認：dataRow.hasVersion

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



保留中変更内容のデータベース反映

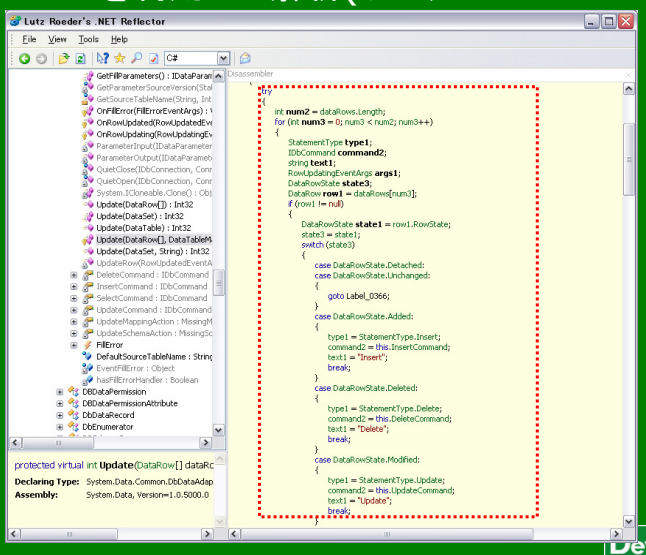
- データアダプタを利用したデータベースへの変更の反映

```
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
  - RowSateの属性とAdapterコマンドの種類にマップ
    - **Detached, Unchanged 次のRowへ**
    - **Added InsertCommnad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

ご質問をどうぞ

Q & A