“delegate+generics”汎用化による量産の技術（俺流なので要注意！！！）

えムナウ（児玉宏之）

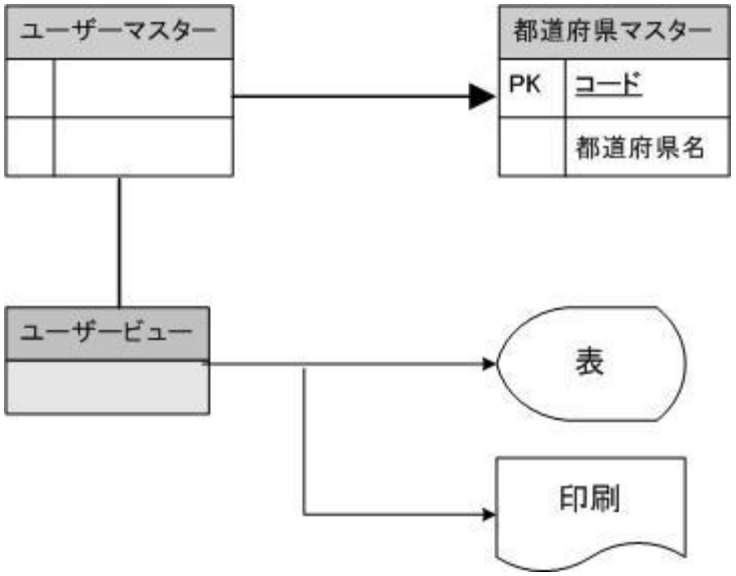
http://mnow.wankuma.com/
http://blogs.wankuma.com/mnow/
http://www.ailight.jp/blog/mnow/

アジェンダ

- データベースとADO.NET
- ラッパーカスタムコントロールのすすめ
- 画面要素のユーザーコントロール化
- 3段継承
- Genericで汎用化
- コントローラを作成してMVCパターン化
- Delegateを使おう

データベースとADO.NET

- 画面や印刷イメージに合わせてビューを作る



データベースとADO.NET

- SQLサーバーならJOINやCASE WHENやサブクエリーは当たり前。
- ビューを元にユーザーと表示イメージや印刷イメージを固められるほどにつめる。

データベースとADO.NET

- ストアドプロシージャを作る
- Fillは中心となるテーブルの
  Add/Update/Deleteを考慮
- VIEWの表示項目を網羅して手間が少なく
- 用意するストアドの例

  Fill/ Add/Update/Delete
  GetByKeys/Search

- メソッドの例

```
void FillTable(DataTable table);
void UpdateTable(DataTable table);
void FillTableByKeys(DataTable table,
    DataTableKeys keys);
DataTableKeys GetKeys(DataRow row);
void FillTableBySearch(DataTable table,
    SearchCondition condition);
```

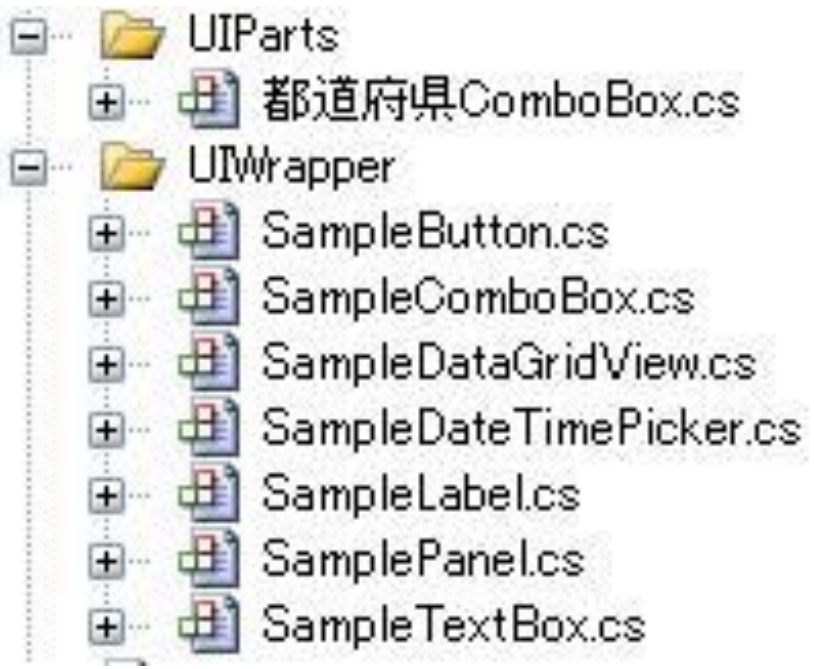
データベースとADO.NET

- TableAdapterをFormのContainerに参加させる。
- TableAdapter は partial class で拡張する
- TableAdapterの資源をDisposeする
- StoredProcedure の戻り値を取得する
- CommandTimeout を指定する

http://mnow.wankuma.com/cs2005_mnowcontrol3.html

ラッパーコントロールのすすめ

- 使用するコントロールはラッパーを作って、それを使用すると一括で変更できて便利
- Enter時に背景色を変える。
- エラーが発生していたら背景色を赤にする



カスタムコントロールのすすめ

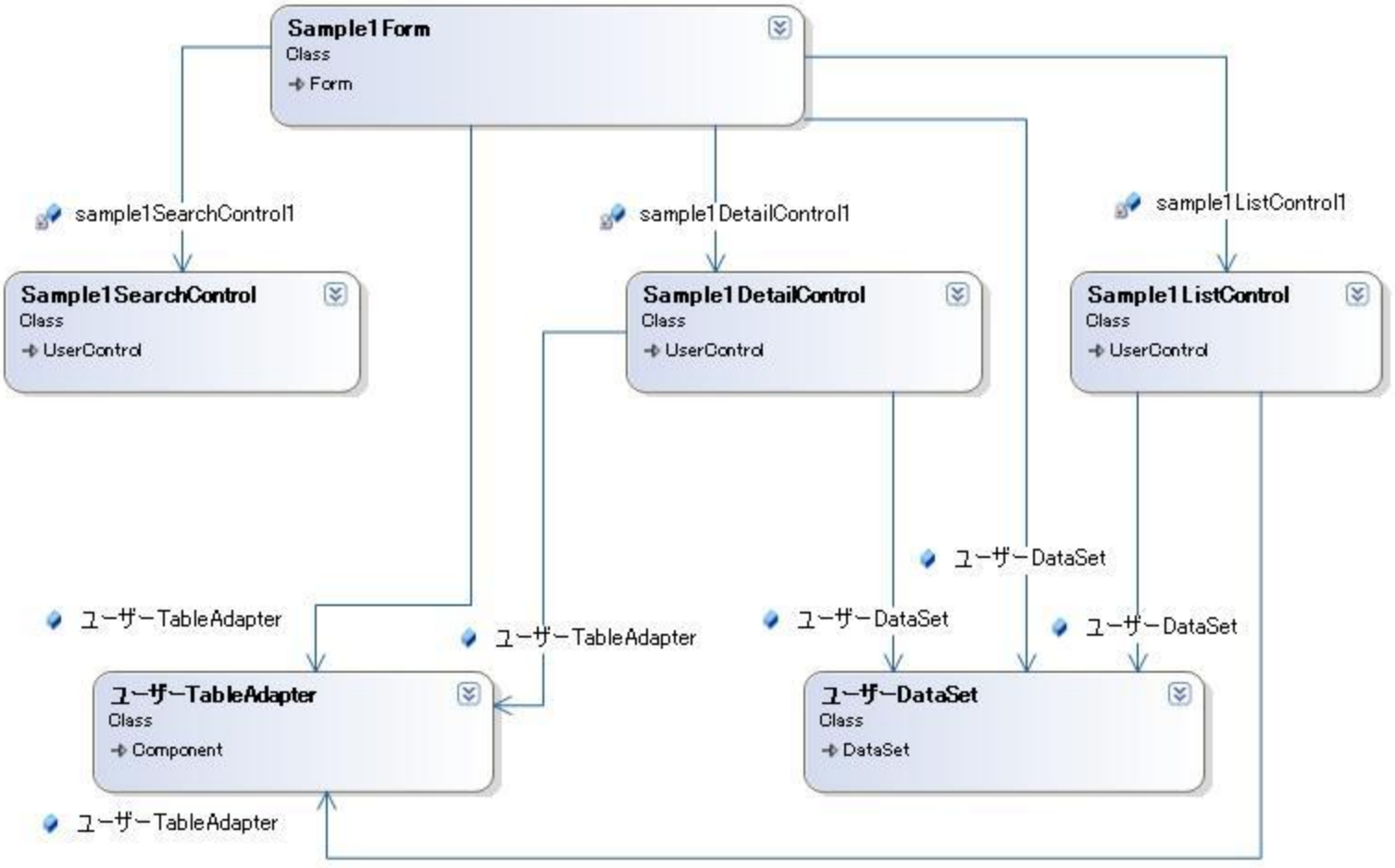
- 例：データベース参照のコンボボックス
- インスタンスを一つだけ作るシングルトンパターンでデータセットを作成する。
- データベースの読み込みはDesignModeがfalseのときだけにしよう。
- DesignModeが有効になるのはInitLayout()かISupportInitializeのEndInit()がおすすめ。

ソースを見ましょう。

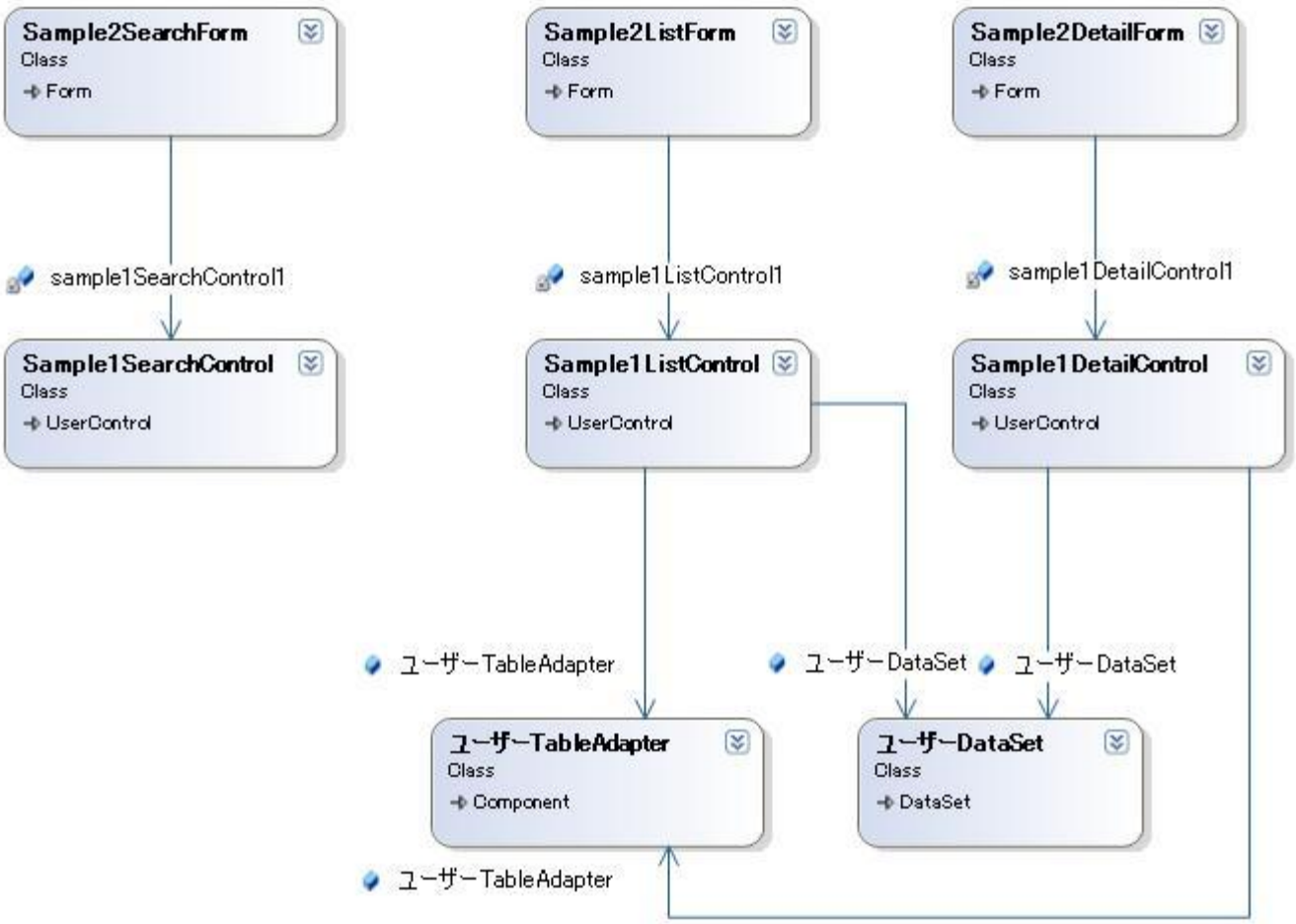
画面要素のユーザーコントロール化

- 画面で一般的に何をしますか？
- 検索一覧詳細のパターンは多いでしょうか？
- 検索と一覧と詳細を一つ一つユーザーコントロールにしましょう。
- 検索一覧詳細の入力Formだけではなく、一覧選択Form・コードと名称から詳細Fromを見れたり応用が効きます。

画面要素のユーザーコントロール化



画面要素のユーザーコントロール化



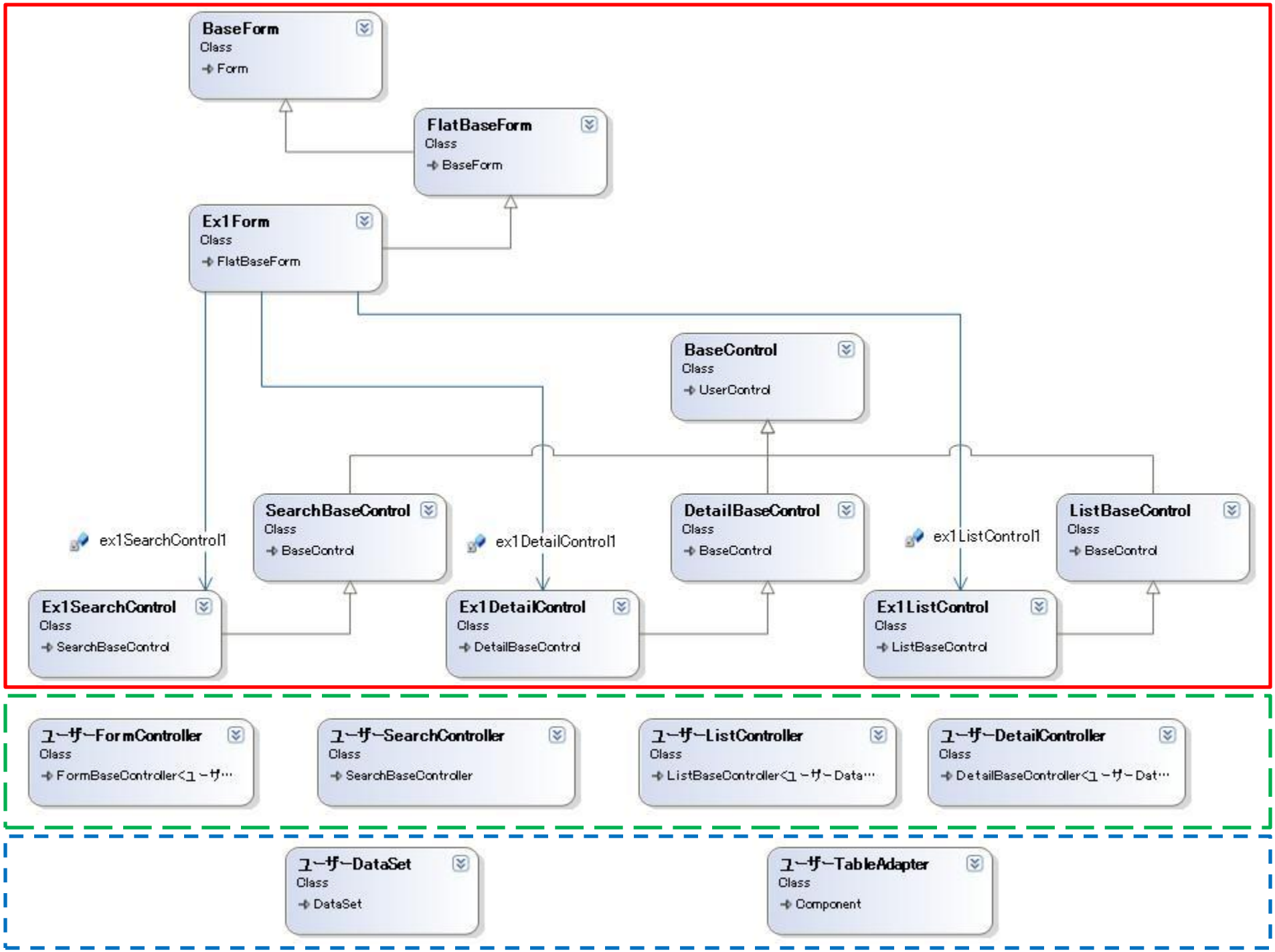
画面要素のユーザーコントロール化

- 検索一覧詳細の入力画面でも別々の3画面でも使うユーザーコントロールは一緒です。
- 検索一覧詳細の入力画面ではデータセットのインスタンスは共通で使えます。
- 別々の3画面ではデータセットのインスタンスは別に用意でします。もちろん詳細はーレコードだけ読み込みます。

デモを見ましょう。

3段継承

- FormやUserControlを継承し元になるクラスを作成し共通部分を書きます。
- 元になるクラスを継承したクラスを作成し目的の機能をすべて実装したFormやUserControlを作成します。
- 目的の機能を持ったFormやUserControlを継承してBindingSource、DataSet、TableAdapterを配置した実際に使うFormやUserControlを作成します。


BaseForm
Class
Form
FlatBaseForm
Class
BaseForm
Ex1Form
Class
FlatBaseForm
BaseControl
Class
UserControl
SearchBaseControl
Class
BaseControl
DetailBaseControl
Class
BaseControl
ListBaseControl
Class
BaseControl
ex1SearchControl1
ex1DetailControl1
ex1ListControl1
Ex1SearchControl
Class
SearchBaseControl
Ex1DetailControl
Class
DetailBaseControl
Ex1ListControl
Class
ListBaseControl
ユーザーFormController
Class
FormBaseController<ユーザ…
ユーザーSearchController
Class
SearchBaseController
ユーザーListController
Class
ListBaseController<ユーザーData…
ユーザーDetailController
Class
DetailBaseController<ユーザーDat…
ユーザーDataSet
Class
DataSet
ユーザーTableAdapter
Class
Component

コントローラを作成してMVCパターン化

- MVC（Model-View-Controller）
- Modelはデータのことです。データを変更するビジネスルールも含みます。
- Viewはユーザにどのように見えるのかつまり画面のことです。
- Controllerはアプリケーションの制御を行います。ViewとModelの橋渡しをするだけではなく、ViewとViewの橋渡しも担当します。

コントローラを作成してMVCパターン化

- 画面はバインドする為にBindingSource、DataSet、TableAdapterを持ちます。
- このMVCパターンではデータを扱う処理はModelに書きますが、データそのものはViewに含まれています。
- Visual Studio の環境を便利に使って画面を作成するにはデータそのものはViewに含めたほうがいいからです。

コントローラをGenericで汎用化

```
public class DetailBaseController<DS, DT, DR>
    where DS : DataSet
    where DT : DataTable
    where DR : DataRow
```

- こんな定義をします、DS, DT, DRと仮に名前付けて実体を定義するときにDSはDataSet、DTはDataTable、DRはDataRowを継承したものを当てはめてください、という意味合いです。

コントローラをGenericで汎用化

- DSという仮の型で定義されたものはDataSetで定義されている機能はすべて使えます。
- DetailBaseControllerというクラスではDSという型でDataSetの機能を使っておいて、継承したクラスで実際の型をあてはめればDetailBaseController側でDSに対してつかった機能は、継承したクラスで実際の型で使ったのと同じことになります。

コントローラをGenericで汎用化

public class ユーザーDetailController :
DetailBaseController
<ユーザーDataSet,
ユーザーDataTable,
ユーザー DataRow>

- こんな定義をするとユーザーDetailControllerはDetailBaseController側でDSに対してつかった機能は、継承したクラスでユーザーDataSetで使ったのと同じことになります。

コントローラをGenericで汎用化

- つまり、Genericを使って抽象化や汎用化ができるようになります。
- これでいろんなものが抽象化できてGenericを使えばコントローラを作成でき、ばんばいざいですよね。
- じゃ作ることを考えてみましょう。

コントローラをGenericで汎用化

- コントローラはBindingSource、DataSet、TableAdapterの情報を理解する必要があります。
- BindingSourceはBindingSourceクラス
- 型付きDataSetの基底クラスはDataSet
- ではTableAdapterの基底クラスは何でしょう？

Delegateを使おう

- はい、答えはComponentです。System.ComponentModel.Component
- TableAdapterはComponentModelから踏み外していることが一部で有名ですが、さらにひどいのはTableAdapterの特徴を説明する基底クラスを用意していないところです。
- つまり基底クラスのルールは守らないくせに基底クラスとしてルール付けして欲しいところはサポートしない、めちゃくちゃなやつです。

Delegateを使おう

- そこで出番はdelegateです。

```
public delegate void
   FillDelegate(DataTable table);
public delegate void
    UpdateDelegate(DataTable table);
```

Delegateを使おう

- Delegateって使ったことありますか？
- Delegateは簡単に言うとメソッドのポインタと考えてください。
- 定義してあるメソッドをDelegateを介して呼んでやります。

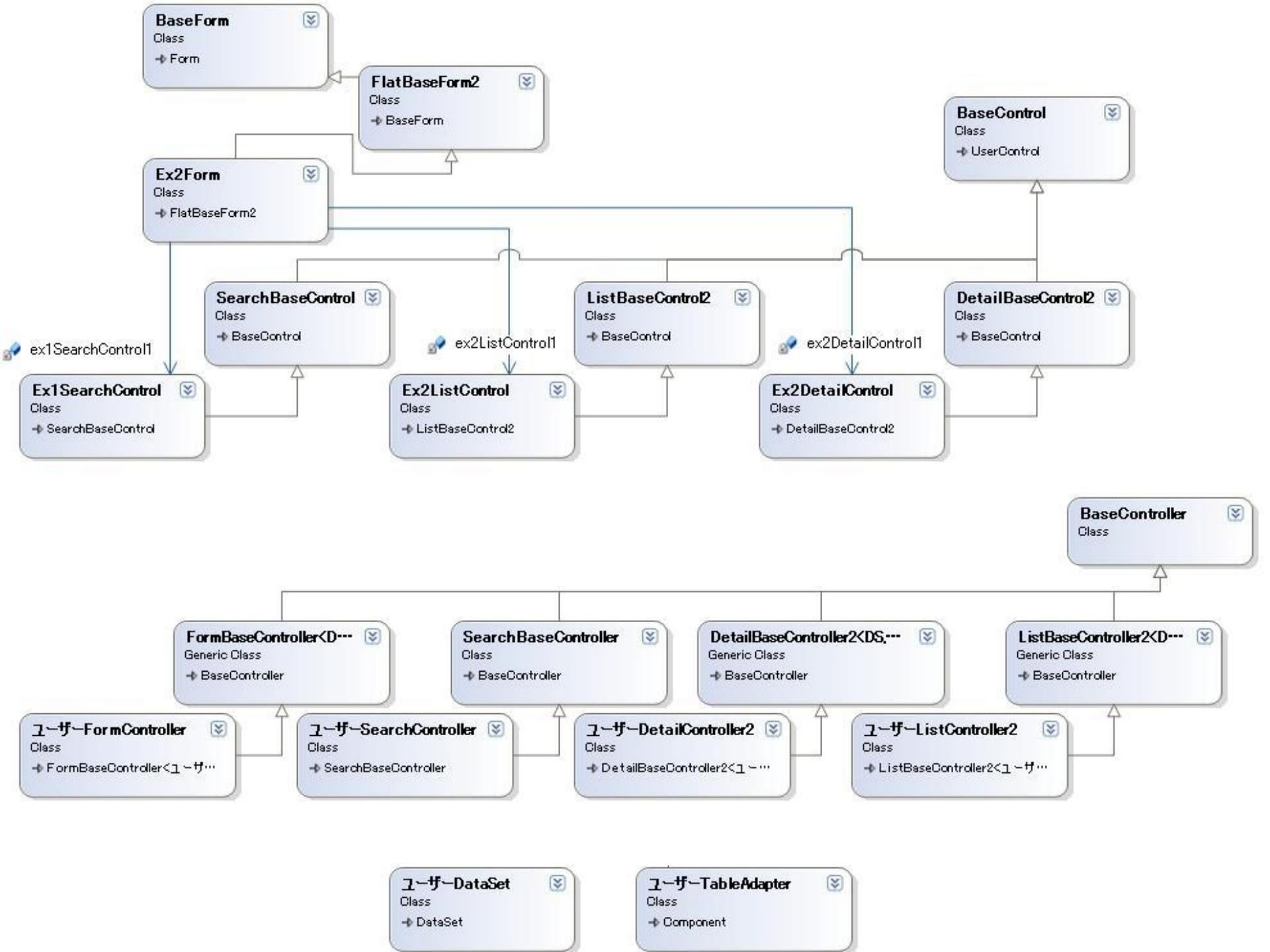
Delegateを使おう

- イベントハンドラもデリゲートです。

```
private void xxx_Click(object sender,
    EventArgs e)
public event EventHandler Click


public delegate void EventHandler ( Object
    sender, EventArgs e )
```

Delegateを使おう

```
public delegate void UpdateDelegate(DataTable table);

private UpdateDelegate _update;
public UpdateDelegate Update
{
    get { return _update; }
    set { _update = value; }
}

Update(DataTable);
```


BaseForm
Class
Form
FlatBaseForm2
Class
BaseForm
BaseControl
Class
UserControl
Ex2Form
Class
FlatBaseForm2
SearchBaseControl
Class
BaseControl
ListBaseControl2
Class
BaseControl
DetailBaseControl2
Class
BaseControl
ex1SearchControl1
ex2ListControl1
ex2DetailControl1
Ex1SearchControl
Class
SearchBaseControl
Ex2ListControl
Class
ListBaseControl2
Ex2DetailControl
Class
DetailBaseControl2
BaseController
Class
FormBaseController<D···
Generic Class
BaseController
SearchBaseController
Class
BaseController
DetailBaseController2<DS,···
Generic Class
BaseController
ListBaseController2<D···
Generic Class
BaseController
ユーザーFormController
Class
FormBaseController<ユーザ···
ユーザーSearchController
Class
SearchBaseController
ユーザーDetailController2
Class
DetailBaseController2<ユー···
ユーザーListController2
Class
ListBaseController2<ユーザ···
ユーザーDataSet
Class
DataSet
ユーザーTableAdapter
Class
Component

- Generic・Delegateで単純化してコントローラを作成してMVCパターン化しました。

- 実際のソースを見てみましょう。

時間が余ったらおまけ

わんくま同盟ではないですが、
TimberLandChapel.comで、
2月位に今回＋αで2時間やります。
興味のある方は是非よろしく。