CM-08 コードリスト管理機能能

■ 概要

本機能は Enum やデータベース、ビジネスロジックの実行結果等といったデータソースから取得した列挙値を、データソースの種類を意識することなく取得管理する機能である。
ComboBox、ListBox といった UI コントロールと、各種データソースから取得した列挙値をデータバインドし、コードリストを選択肢として表示することができる。
データの取得元に関する情報は、SelectableValuesExtension 継承クラスを実装し、事前に登録しておく。開発者は、SelectableValuesProvider クラスを Visual Studio のデザイナ上で画面に張り付けて ComboBox、ListBox といった UI コントロールのプロパティエディタで事前に定義したデータ取得元を指定するだけで、簡単にコードリストを表示させることができる。
本機能では、ComboBox、ListBox といった UI コントロールとバインド可能なコードリストを格納する汎用的なデータクラスとして SelectableValue クラスを提供する。
表示対象となる画面を起動する際に、SelectableValuesManager クラスは、SelectableValuesExtension 継承クラスで定義した情報に基づき、データソースからデータを取得して SelectableValue オブジェクトのリストを生成し、UI コントロールとデータバインドすることでコードリストを表示する。なお、生成した SelectableValue はキャッシュされ、それ以降のコードリストの取得はキャッシュを利用することで効率的に処理される。

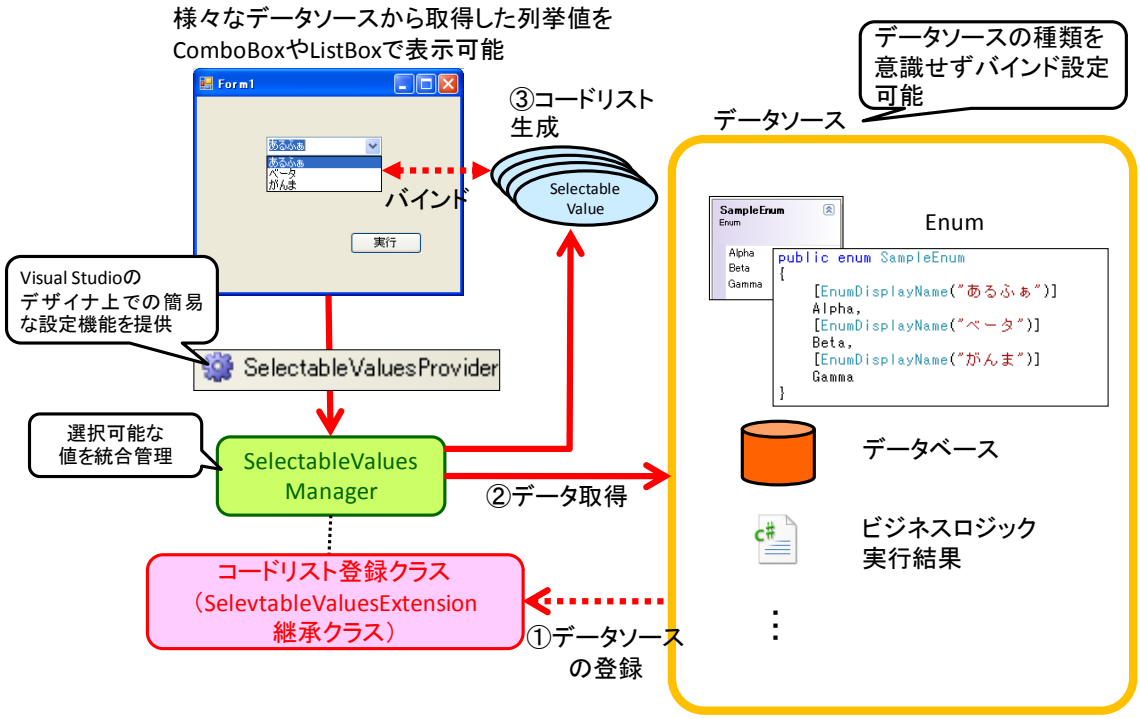

図 1 コードリスト管理機能の動作概念図

■ 使用方法

◆ Enum を使ったコードリスト

Enum を使ったコードリストを実装する場合、TERASOLUNA フレームワークが提供する「コードリスト用 Enum クラス」カスタムテンプレートを使用する。

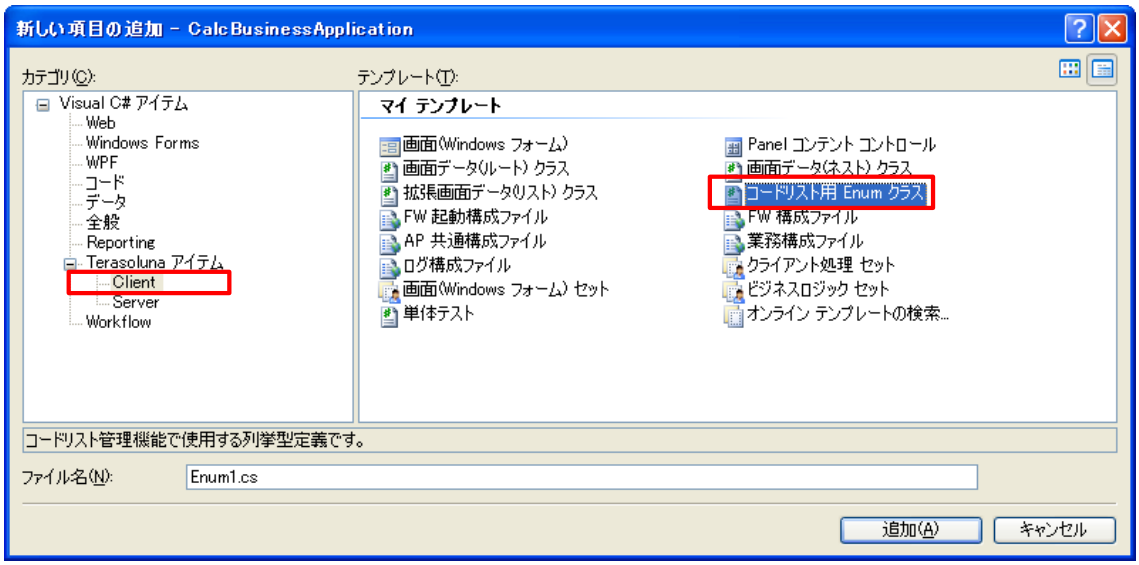

図 2 コードリスト用 Enum クラスカスタムテンプレート

生成されたテンプレートをベースに、Enum の値に対してそれぞれ EnumDisplayName 属性を付与し、画面への表示名を定義する。なお、表示名に対するコードの値は Enum になる。

以下に実装例を示す。この例では、UI コントロール上(ComboBox、ListBox 等)には、「飛行機」、「車」、「電車」、「バス」、「船」を一覧表示し、UI コントロールの Item として、AirPlane、Car、Train、Bus、Ship という Enum 値のリストを登録する。

```
public enum TransportationEnum
{
    [EnumDisplayName("飛行機")]
    AirPlane,
    [EnumDisplayName("車")]
    Car,
    [EnumDisplayName("電車")]
    Train,
    [EnumDisplayName("バス")]
    Bus,
    [EnumDisplayName("船")]
    Ship
}
```

リスト 1 Enum によるコードリストの実装例

◆ DB を使ったコードリスト

DB を使ったコードリストを実装する場合、コードリストを保持するテーブルを用意し、表示名を表すカラムとコードの値の表すカラムを定義する。この時、テーブルにこれらのカラム以外が存在しても問題ない。
コードリストの表示名を表すカラム名は「DisplayName」、値を表すカラムは「Value」である。ただし、SQL の Select 文で取得する時に AS 句を使用してカラム名の別名を「DisplayName」や「Value」にすることで、カラム名は任意の文字列を定義することが可能である。
以下に、コードリストを保持するテーブルの例を示す。この例では、表示名を表す列が「CategoryName」、コードの値を表す列が「CategoryCode」であるので、SQL の Select 文で取得するときに AS 句の指定が必要である。

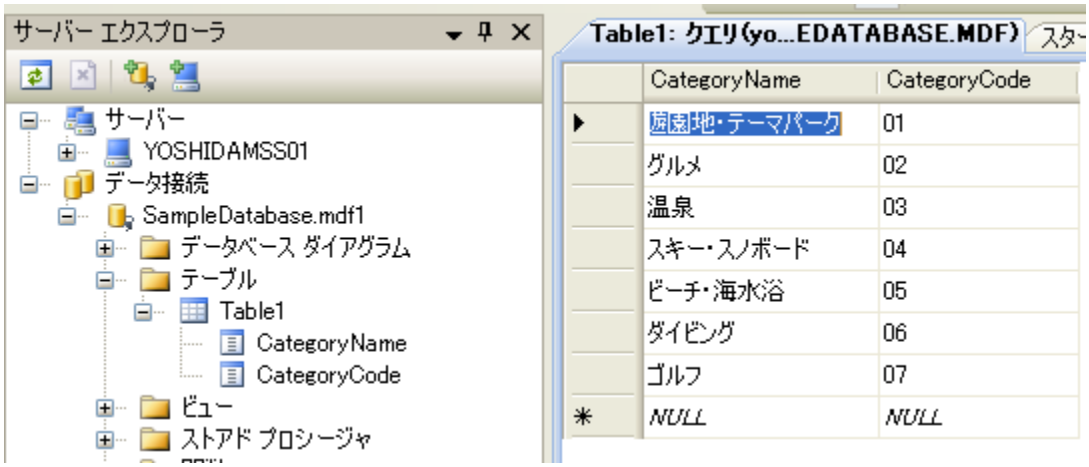

図 3 データベースによるコードリストの例

また、DB ベースの接続情報は、AP 共通構成ファイル(TerasolunaApplication.config)または業務構成ファイル(“アセンブリ名”.config)で、Terasoluna.Data.IDbProviderManager インタフェースの DI 設定で記述する。IDbProviderManager インタフェースの標準実装として、Terasoluna.Data.DbProviderManager クラスを提供している。
DbProviderManager では ConnectionName プロパティを指定して、アプリケーション構成ファイル(App.config)の /configuration/connectionStrings/add タグで指定した接続文字列の name 属性の値を指定する。

以下に、構成ファイルの記述例を示す。

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <unity>
    <typeAliases>
      <typeAlias alias="string" type="System.String" />
      <typeAlias alias="int" type="System.Int32" />
      <typeAlias alias="IDbProviderManager"
            type="Terasoluna.Data.IDbProviderManager, Terasoluna" />
      <typeAlias alias="DbProviderManager"
                  type="Terasoluna.Data.DbProviderManager, Terasoluna" />
    </typeAliases>
    <containers>
      <container>
        <types>
          <type type="IDbProviderManager" mapTo="DbProviderManager">
            <lifetime type="singleton" />
            <typeConfig>
              <!-- 接続文字列のname属性の値を指定する -->
              <property name="ConnectionStringName" propertyType ="string">
                <value value="SampleDatabaseConnectionString" type="string"/>
              </property>
            </typeConfig>
          </type>
        </types>
      </container>
    </containers>
  </unity>
</configuration>
```

リスト 2 AP 共通構成ファイル(TerasolunaApplication.config)の記述例

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <connectionStrings>
    <add name="SampleDatabaseConnectionString"
         connectionString="Data
         Source=.¥SQLEXPRESS;AttachDbFilename=|DataDirectory|¥SampleDatabase.mdf;Integrat
         ed Security=True;User Instance=True"
        providerName="System.Data.SqlClient" />
  </connectionStrings>
・・・
</configuration>
```

リスト 3 アプリケーション構成ファイル(App.config)に設定した接続文字列

◆ ビジネスロジックの処理結果によるコードリスト

クライアントまたはサーバビジネスロジックの処理結果をもとにコードリストを実装する場合、以下のどちらかを実行できる。この時、メソッドは引数なしで、戻り値を IList<SelectableValue>型にすること。この時、SelectableValue のコンストラクタの第1引数に表示名、第2引数にコードの値を指定する。
サーバビジネスロジックを直接実行する場合については、Visual Studio の「サービス参照の追加」により WSDL をもとに生成された WCF クライアントを使用する。
また、クライアントビジネスロジックを介してサーバビジネスロジックを実行する場合は、クライアントビジネスロジックの中で、WCF クライアントを呼び出すコードを実装する。WCF クライアントの呼び出しについては、「CL-04 サーバ通信機能」を参照のこと。
以下に、クライアントビジネスロジックの実装例を示す。

```
public class BizLogicA
{
    public IList<SelectableValue> Execute()
    {
        //ビジネスロジックの処理結果をIList<SelectableValue>で返却
        List<SelectableValue> result = new List<SelectableValue>();
        result.Add(new SelectableValue("一般ユーザ", "01"));
        result.Add(new SelectableValue("管理者", "02"));
        return result;
    }
}
```

リスト 4 クライアントビジネスロジックの実装例

◆ コードリストの登録

本機能を利用するには、SelectableValuesExtension 継承し Extension クラスを実装し、各開発プロジェクトで必要なコードリストを登録する。通常は、アーキテクトまたは業務共通開発者が Extension クラスを実装し、事前に AP 共通構成ファイル(TerasolunaApplication.config)に登録しておくことで、各開発者がこれら設定を意識することなく、用意されたコードリストを利用できるようにしておくのがよい。また、個別業務内でのみ利用したいコードリストがある場合は、当該業務プロジェクト内に、別途 SelectableValuesExtension 継承クラスを作成し業務構成ファイルに登録することも可能である。
なお、各業務プロジェクトで登録したコードリストは「業務用 UnityContainer」で個別に保持されるが、AP 共通構成ファイルに登録設定した共通コードリストも、実行時には各「業務用 UnityContainer」で個別に保持されることになるので注意が必要である。
TERASOLUNA フレームワークで管理される UnityContainer の種類については、「CM-02 インスタンス管理機能」の機能説明書も併せて参照のこと。
実装方法としては、SelectableValuesExtension の Setup メソッドをオーバライドして、コードリストを登録する。
コードリストの登録には、以下の Register メソッドを利用する。ここで、第1引数の「カテゴリ名」は、コードリストの種類を一意に識別する名前である。また、開発者が ComboBox や ListBox といった UI コントロールとバインド設定する際に Visual Studio のプロパティエディタ上に表示される前

として利用される。

表 1 SelectableValuesExtension クラスの Register メソッドの一覧

項番	Register メソッドの種類	第 1 引数	第 2 引数	説明
1	protected void RegisterFromEnum(string categoryName, Type enumType)	カテゴリ名	Enum の型	指定した Enum の列挙値を、コードリストとして登録する。
2	protected void Register(string categoryName, IList<SelectableValue> selectableValues)	カテゴリ名	SelectableValue のリスト	指定した SelectableValue のリストの値を、コードリストとして登録する
3	protected void RegisterFromDatabase(string categoryName, string sql)	カテゴリ名	SQL 文	指定した SQL を実行し DB から取得したレコード群を、コードリストとして登録する。 ※表示名のカラム名を「DisplayName」、コード値のカラム名を「Value」とする SQL を指定すること。
4	protected void RegisterFromBizLogic(string categoryName, BizLogicInfo info)	カテゴリ名	ビジネスロジックの実行情報	ビジネスロジックの実行結果を、コードリストとして登録する。 ※ ビジネスロジック情報に指定する、ビジネスロジックのメソッドは、引数なしで、戻り値が IList<SelectableValue> であること。

RegisterFromDatabase メソッドの第2引数に渡す SQL により取得するテーブルのカラム名が「DiplayName」、「Value」でない場合は、前述の通り、SQL の SELECT 文の AS 句を使用して別名を「DiplayName」、「Value」とする。
RegisterFromBizLogic メソッドの第2引数に渡す BizLogicInfo は、ビジネスロジックの実行にかかわる情報を格納する。以下に、BizLogicInfo のコンストラクタを示す。
BizLogicInfo の設定については、「CL-03 イベント処理実行機能」の機能説明書の「ビジネスロジック実行の設定」の項を併せて参照のこと。

表 2 BizLogicInfo のコンストラクタ

項番	Register メソッドの種類	引数
1	public BizLogicInfo(string executorName, Type bizLogicType, string bizLogicName, string methodName)	第1引数: ビジネスロジックの実行クラス名を指定する。 標準では、以下の文字列が指定可能である。 “ClientBizLogic”: クライアントビジネスロジッククラスを実行する場合に使用 “WcfProxy”: WCF クライアント(ClientBase 継承クラス)を使って直接サーバビジネスロジックを実行する場合に使用 第2引数: 実行するビジネスロジックのクラスの型を指定する。 第1引数が、“ClientBizLogic”の場合は、クライアントビジネスロジックの型名を指定する。”WcfProxy”の場合は、WCF クライアントクラスを指定する。 第3引数: 構成ファイルに DI 設定を記述している場合に、本引数と一致する name 属性を持つ設定に基づき UnityContainer よりインスタンス生成する。 DI 設定しない場合には、null を指定することでよい。 第4引数: 実行するビジネスロジッククラスのメソッド名を指定する。

以下に、SelectableValuesExtension の記述例を示す。

```
public class SampleSelectableValuesExtension : SelectableValuesExtension
{
    protected override void Setup()
    {
        base.Setup();
        //Enumによるコードリストの例
        RegisterFromEnum("交通機関", typeof(TransportationEnum));

        //DBによるコードリストの例
        RegisterFromDatabase("ツアーコード",
            @"SELECT
                CategoryName AS DisplayName,
                CategoryCode AS Value
              FROM Table1");

        //ビジネスロジックによるコードリストの例
        RegisterFromBizLogic("役割",
            new BizLogicInfo("ClientBizLogic", typeof(BizLogicA), null, "Execute"));

    }
}
```

リスト 5 SelectableValuesExtension 継承クラスの実装例

◆ 構成ファイルの設定

本機能は、「CM-02 インスタンス管理機能」を利用しており、SelectableValueExtension クラスは、UnityContainerExtension 継承クラスである。定義したコードリストを登録するには、表 3 のいずれかの構成ファイルで、/configuration/unity/containers/container/extension/add タグにより設定する。このとき、複数回 add タグを使って複数個の SelectableValuesExtensions 継承クラスを追加することもできる。

表 3 コードリストの設定可能な構成ファイル

項番	登録先構成ファイル	説明
1	AP 共通構成ファイル (TerasolunaApplication.config)	ソリューション全体で共通的に利用するコードリストを登録する。ここに登録したコードリストは、全ての業務プロジェクトにおいて利用可能である
2	業務構成ファイル (“アセンブリ名”.config)	業務プロジェクトで利用するコードリストを登録する。登録したコードリストは、当該業務プロジェクトにおいてのみ、利用可能である

以下に、構成ファイルの記述例を示す。

```
<unity>
  <containers>
    <container>
      <types>
      </types>
      <extensions>
        <add type ="Terasoluna.Windows.Laboratories.SelectableValues.SampleSelectableValuesExtension,
                    Terasoluna.Windows.Laboratories"/>
      </extensions>
    </container>
  </containers>
</unity>
```

リスト 6 構成ファイルの記述例

また、「ビジネスロジックの処理結果によるコードリスト」を利用する場合は、AP 共通構成ファイルに /configuration/unity/containers/container/extension/add タグにより、以下の UnityCotainerExtension 継承クラスを追加する。

- Terasoluna.Windows.Forms.BizLogic.ClientBizLogicExtension
  - ビジネスロジッククラスの実行に必要な UnityContainerExtension。「CL-03 イベント処理実行機能」で利用する Extension クラス。
- Terasoluna.ServiceModel.CommunicationExtension
  - 「CL-04 サーバ通信機能」の Extension クラス。サーバビジネスロジックの処理結果をもとにコードリストを生成する場合に必要。

以下に、TerasolunaApplication.config の記述例を示す。

```
        <extensions>
・・・
          <!-- ビジネスロジック実行の設定 -->
          <add type="Terasoluna.Windows.Forms.BizLogic.ClientBizLogicExtension,
                    Terasoluna.Windows.Forms" />
          <!-- サーバ通信機能のデフォルト設定 -->
          <add type="Terasoluna.ServiceModel.CommunicationExtension,
                    Terasoluna" />
        </extensions>
```

リスト 7 TerasolunaApplication.config の設定例

◆ SelectableValuesProvider の追加とバインド設定

業務開発者は、コードリストを表示したい画面または部品に SelectableValuesProvider を追加した上で、UI コントロールとコードリストの紐づけを設定する。
SelectableValuesProvider は Component 継承クラスであり、画面に貼り付け可能な部品である。SelectableValuesProvider を Visual Studio のデザイナで対象画面へ張り付けると、張り付けた画面上にある全ての System.Windows.Forms.ListControl 継承クラス(ComboBox や ListBox)のプロパティに SelectableCategoryName プロパティが追加表示される。
SelectableCategoryName プロパティには、前述の SelectableValuesExtension 継承クラスの Register メソッドで事前に登録したカテゴリ名の一覧が表示されるので、開発者は、一覧の中から表示したいコードリストを表すカテゴリ名を設定する。
通常、.NET をそのまま使って、ListBox や ComboBox にデータバインド設定をする場合は、DataSource プロパティ、DisplayMember プロパティ、ValueMember プロパティの設定が必要である。SelectableCategoryName プロパティの設定は、この3つのプロパティの設定と等価であり、SelectableCategoryName プロパティを設定すれば、DataSource プロパティ、DisplayMember プロパティ、ValueMember プロパティの設定は不要である。[1]
DataSource プロパティには、コードリストとして IList<SelectableValue>オブジェクトが自動的に設定される。また、DisplayMember プロパティには SelecatableValue クラスの「DisplayName」プロパティが自動的に設定され、ValueMember プロパティには SelectableValue クラスの「StringValue」プロパティ(コードの値を文字列化したものを返却する)が自動的に設定される。
例えば、データソースが Enum の場合、EnumDisplayName 属性に指定した値が DisplayMember、Enum 値を文字列化した値が ValueMember になる。

[1] SelectableValue クラスを SelectableValuesProvider と併用せず単独で利用する場合、ListControl(ListBox や ComboBox 等)のプロパティエディタ上で、DisplayMember プロパティに SelectableValue クラスの「DisplayName」フィールド、ValueMember プロパティに、「StringValue」フィールドを、それぞれ設定すること。

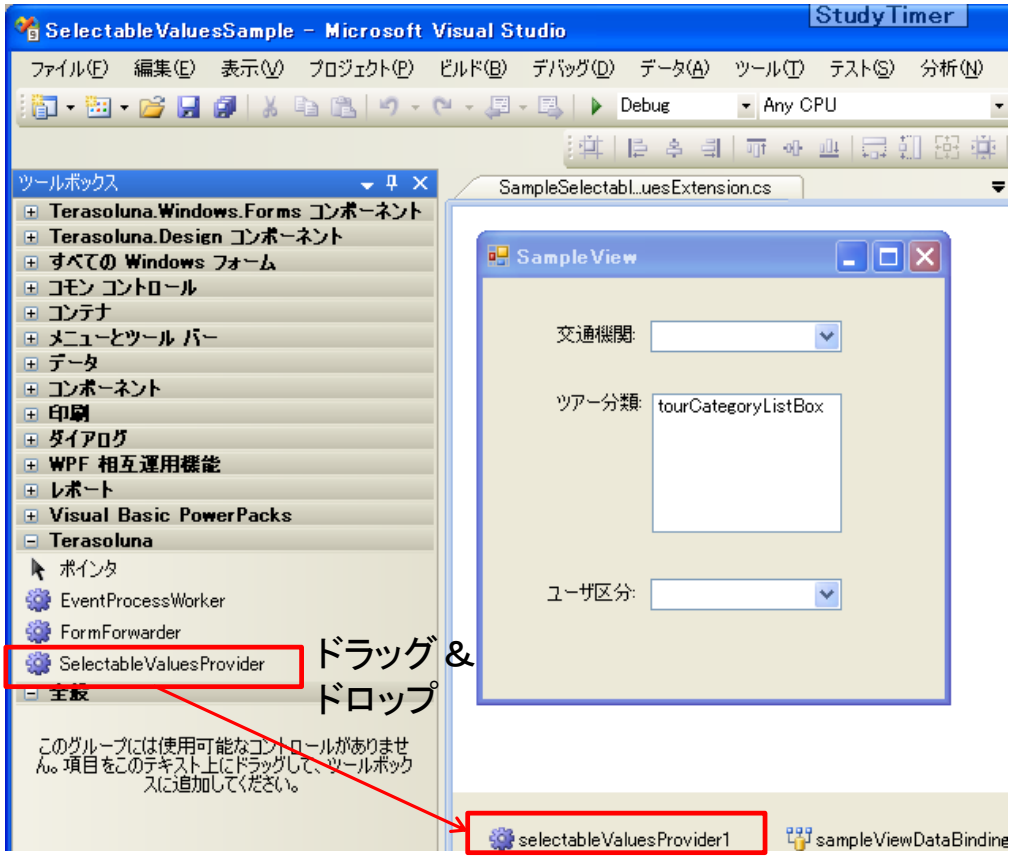

図 4 SelectableValuesProvider の追加

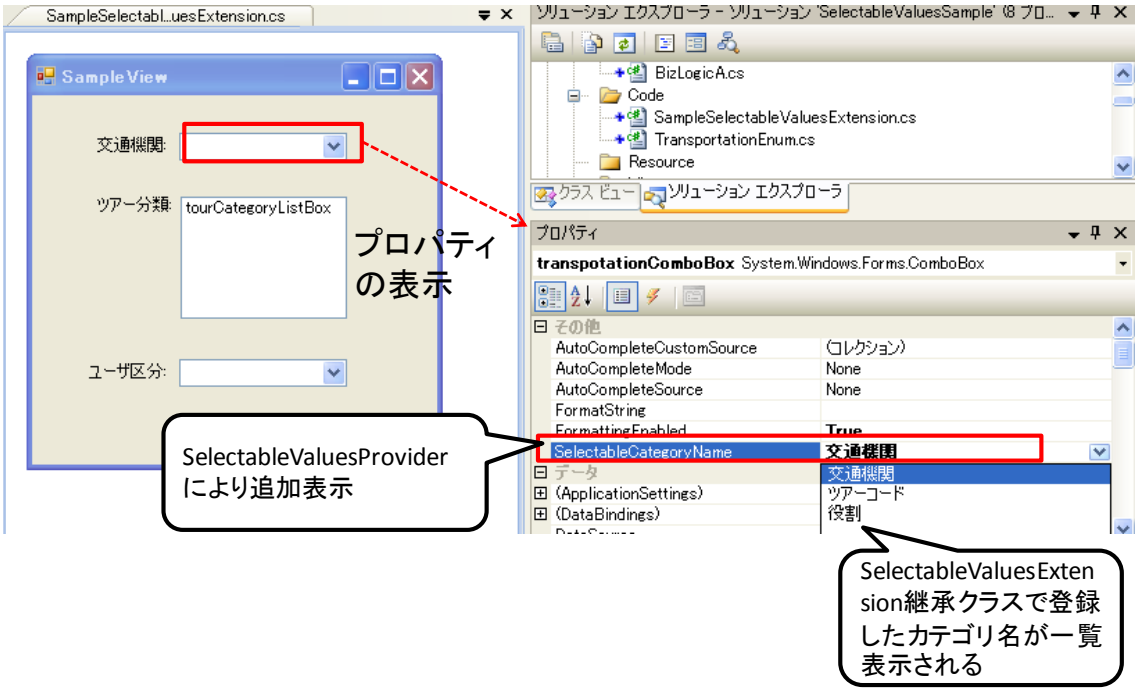

図 5 ListControl に追加された SelectableCategoryName プロパティの設定

◆ 実行結果

リスト 5 のようにコードリストを定義し SelectableValuesProvider を利用することで、画面にリストデータを表示することができる。

図 6 実行結果

◆ コードリストのリロード

一旦コードリストを取得すると、アプリケーション起動中は、次回以降はメモリ上にキャッシュした同コードリストを使用し、コードリストを何度も取得しない仕組みになっている。
データソースが保持するコードリストに変更があった場合には、リロードすることで最新のコードリストを再度取得することができる。コードリストのリロードには、SelectableValuesProvider クラスの提供する以下のメソッドを使用する。
注意点として、リロード前に生成した画面とバインドしているコードリストは、リロード後もリロード前のコードリストのままであるため変更は反映されない。最新のコードリストを画面に表示するためには、リロード後に画面を再作成する。画面の非表示(Hide)により画面遷移を実施する場合には注意すること。

表 4 SelectableValueManager のリロード用メソッド

項番	メソッドの種類	第 1 引数	説明
1	public static void ReloadAll()	-	アプリケーションで使用している全てのコードリストをリロードする。
2	public static void Reload(string categoryName)	カテゴリ名	指定したカテゴリ名のコードリストをリロードする。

◆ UI コントロールと画面データのバインド

本機能を使って表示した ComboBox や ListBox について、ユーザがリストから選択したコードの値(Value)と画面データクラスのプロパティを双方向バインドするには、ListControl.Text プロパティではなく ListControl.SelectedValue プロパティとすること。

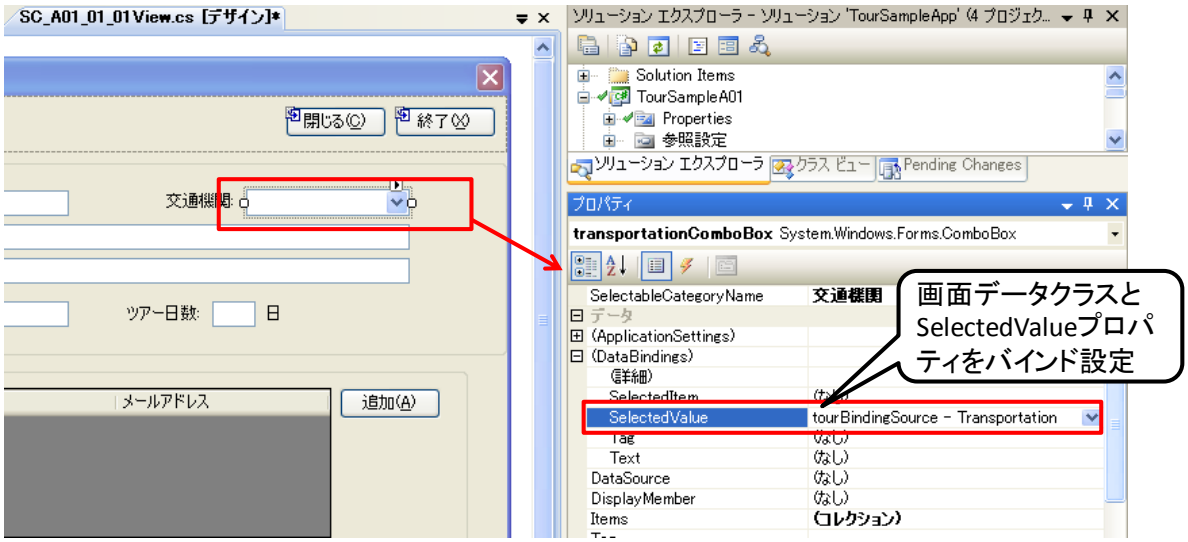

図 7 画面データとのコード値のバインド設定

■ TIPS

以下では、本機能を利用する際、開発者がよく直面する問題について対処方法をまとめている。開発時に実装方法に困った場合に参考にするとよい。

◆ コードリストの取得タイミング

アプリケーションの性能や保守性等を考慮して、アプリケーション起動時や初期画面の Load イベント等で使用するコードリストを一括で取得するように実装するとよい。この時、取得したコードリストは AP 全体で共有するデータクラス(Singleton 等 static なデータ領域)に格納しておき、本機能の Extension クラスに登録する、コードリストを取得するビジネスロジックはこのデータクラスから取得するようにする。なお、SelectableValueManager のリロードメソッドを使用する場合は、リロードメソッドを呼ぶ前にコードリストを一括取得する処理を再実行して AP 全体で共有するデータクラスに格納し直してからリロードすること。
また、アプリケーション起動時のコードリストの取得処理の実装方法の1つとして、「CM-01 アプリケーション起動・終了機能」が提供する IInitializer インタフェースを実装する方法がある。IInitializer インタフェースの実装方法については、「CM-01 アプリケーション起動・終了機能」の機能説明書を参照すること。

■ 内部構成

◆ 構成クラス

本機能を構成するクラスを以下に示す。表 5 構成クラス一覧

項番	クラス名	説明
Terasoluna.SelectableValue 名前空間		
1	SelectableValue	コードリストとして、表示名とコード値を保持する汎用的なデータクラス
2	SelectableValuesManager	本機能のエントリークラス
3	DefaultSelectableValuesFactory	ISelectableValuesFactory のデフォルト実装クラス
4	ISelectableValuesBuilder	SelectableValue のリストを作成するインタフェース
5	DefinedSelectableValuesBuilder	定義した SelectableValue のリストをもとにコードリストを作成するクラス
6	EnumSelectableValuesBuilder	Enum をもとに選択可能な値のリストを作成するクラス
7	DatabaseSelectableValuesBuilder	データベースから取得した値をもとに選択可能な値のリストを作成するクラス
8	BizLogicSelectableValuesBuilder	ビジネスロジックの処理結果をもとに選択可能な値のリストを作成するクラス
9	SelectableValuesInfo	SelectableValue を保持するクラス
10	SelectableValuesInfoManager	SelectableValue を管理するクラス
11	SelectableValuesExtension	本機能を使ってコードリストを登録するための基底 Extension クラス。
Terasoluna.Data 名前空間		
11	DbAccessManager	DB アクセスを実施するクラス
12	DbColumnMapping	SQL の結果に対するカラム情報を保持するクラス
13	DbCommandExecutor	DB コマンドを実行するクラス
14	IDbProviderManager	DB プロバイダに関する情報を管理するインタフェース
15	DbProviderManager	IDbProvider のデフォルト実装クラス
16	CommandSetter	DB コマンドをセットするためのデリゲート

17	RowCreator	SQL の結果をもとにオブジェクトを生成するためのデリゲート
Terasoluna.Windows.Forms.Contorols 名前空間		
11	SelectableValuesProvider	UI コントロールに Visual Studio のエディタで本機能の設定を追加するコンポーネント

■ 拡張ポイント

コードリストの取得元となるデータソースを追加する場合は、ISelectableValuesBuilder インタフェースを実装し、対象のデータソースからコードリストを生成する処理を実装する。
また、ISelectableValuesBuidler 実装クラスを使用してコードリストを登録するように SelectableValuesExtension 継承クラスして Extension クラスを実装し、構成ファイルに作成した Extension クラスを設定する。
以下に、実装例を示す。

```
public class CsvFileSelectableValuesBuilder : ISelectableValuesBuilder
{
    public string FileName { get; set; }

    public IList<SelectableValue>  CreateSelectableValues()
    {
        // CSVファイルからコードを取得するように実装
    }
}
```

リスト 8 ISelectableValuesBuilder の実装

```
public class SampleSelectableValuesExtension : SelectableValuesExtension
{
    protected override void Setup()
    {
        base.Setup();
        ・・・
        //CSVファイルによるコードリストの登録
        RegisterFromCsvFile("カテゴリ", "testdata.csv");
    }

    protected void RegisterFromCsvFile(string categoryName, string fileName)
    {
        RegisterBuilder<CsvFileSelectableValuesBuilder>(
            categoryName,
            new InjectionProperty("FileName", fileName));
    }
}
```

リスト 9 SelectableValuesExtension 継承クラスの実装

```
<unity>
  <containers>
    <container>
      <types>
      </types>
      <extensions>
        <add type ="Terasoluna.Windows.Laboratories.SelectableValues.SampleSelectableValuesExtension,
                    Terasoluna.Windows.Laboratories"/>
      </extensions>
    </container>
  </containers>
</unity>
```

リスト 10　構成ファイルの記述例

■ 関連機能

- CM-02 インスタンス管理機能
- CL-04 サーバ通信機能