CM-07 メッセージ管理機能

■ 概要

本機能は、アプリケーションで扱うメッセージを統合的に管理する機能を提供する。

- .NET 標準のリソース管理機能の利用
  .NET Framework が標準提供するリソース管理機能[1]を使用し、リソースファイル(resx ファイル)により「メッセージリソース」を管理する。

- フレームワークが扱う「メッセージリソース」の管理
  フレームワークが出力するログや入力値検証エラーメッセージ、ダイアログメッセージなどのデフォルトのメッセージリソースは、フレームワークのアセンブリ内にあるリソースファイルで管理している。
  本機能は、各リソースファイルを管理する「メッセージリソース」クラスを提供する。
  「メッセージリソース」クラスが提供する API により、独自に作成した「メッセージリソース」を使ってデフォルトメッセージを差し替えることが可能である。

■ 使用方法

◆「メッセージリソース」の作成

本機能において利用するリソースは、.NET Framework が提供する ResourceManager クラスが対応するリソース型及びリソースファイルに準ずる。
リソース使用方法の詳細は、MSDN のドキュメント等を参照のこと。

メッセージを格納するリソース型を作成する場合、Visual Studio のプルダウンメニューより、[プロジェクト - 新しい項目の追加]を選択し、「新しい項目の追加」ダイアログのテンプレートの一覧から「アセンブリ リソースファイル」を追加する。
以下に、Visual Studio からプロジェクトにリソースを追加するイメージを示す。

[1] リソースについては MSDN ドキュメント「.NET Framework 開発者ガイド - リソースのパッケージ化と配置」を参照のこと。

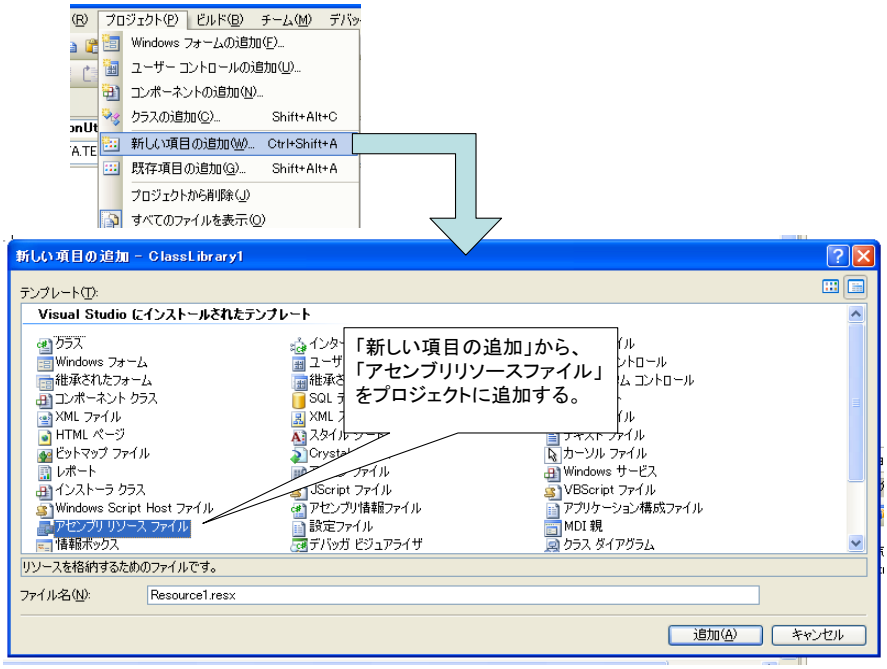

図 1 アセンブリリソースファイルの追加

◆「メッセージリソース」の編集

作成したリソース型をソリューションエクスプローラでダブルクリックするか、「ファイルを開くアプリケーションの選択」から「マネージリソースエディタ」を選択することで、リソースエディタが表示される。リソースエディタでリソースを編集すると、自動的に「[リソース名].Designers.cs」という名称でソースコードが自動生成され、リソースにアクセスするための型が作成される。リソースにアクセスするための型は、リソース名が型名となる。

以下に、「メッセージリソース」の編集イメージを示す。

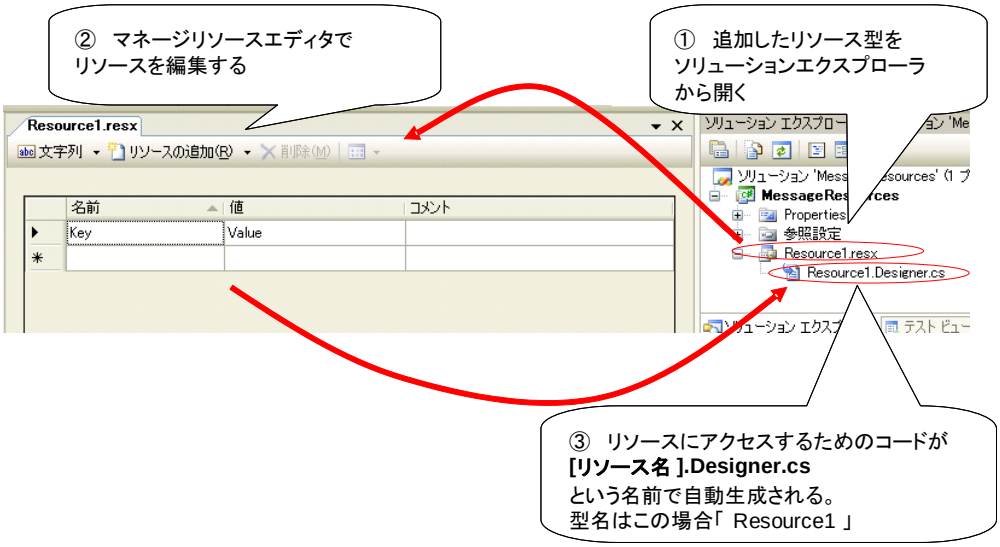

図 2 Visual Studio による「メッセージリソース」の編集イメージ

◆ メッセージ文字列の取得

リソースファイルで定義したメッセージ文字列の取得例と実行結果を以下に示す。
この例では、既定のメッセージと、”en-US”（英語-米国）カルチャ固有のメッセージを切り替えて表示する。

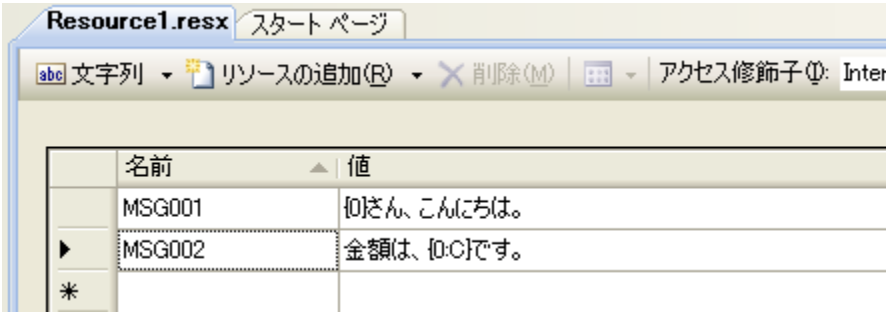

図 3 既定（インバリアント・カルチャ）のリソース(Resource1.resx)

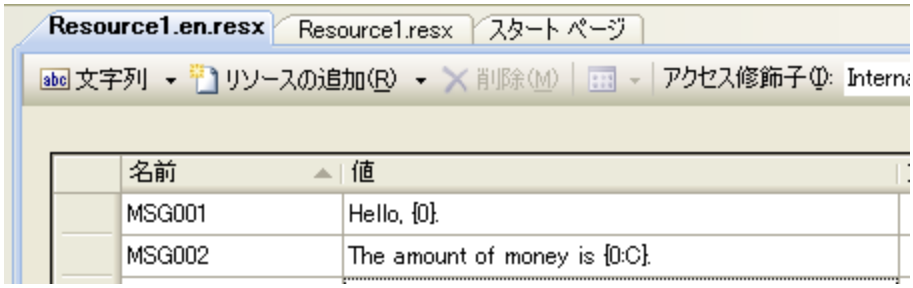

図 4 “en”（英語）カルチャ固有のリソース(Resource1.en.resx)

```
class Program
{
    static void Main(string[] args)
    {
        //置換文字列
        string user = "Terasoluna";
        int money = 10;

        //既定のメッセージの出力
        string msg001 = string.Format(Resource1.MSG001, user);
        string msg002 = string.Format(Resource1.MSG002, money);
        Console.WriteLine(msg001);
        Console.WriteLine(msg002);

        //カルチャを切り替えメッセージ出力
        Thread.CurrentThread.CurrentUICulture = new CultureInfo("en-US");
        Thread.CurrentThread.CurrentCulture = new CultureInfo("en-US");
        msg001 = string.Format(Resource1.MSG001, user);
        msg002 = string.Format(Resource1.MSG002, money);
        Console.WriteLine(msg001);
        Console.WriteLine(msg002);
    }
}
```

リスト 1 メッセージ管理機能の利用の例

```
Terasoluna さん、こんにちは。
金額は、¥10 です。
Hello, Terasoluna.
The amount of money is $10.00.
```

リスト 2 実行結果

◆ フレームワークが管理するデフォルトメッセージの上書き

TERASOLUNA フレームワークが管理するメッセージリソース用のリソースファイルは表 1 のとおりであり、フレームワークを構成するアセンブリのうち、以下のアセンブリ内に複数のリソースファイルを保持している。
リソースファイルおよび管理されるメッセージリソースの定義内容については、フレームワークのソースコードよりリソースファイルを参照すること。

表 1 フレームワークが管理するリソースファイルと対応する「メッセージリソース」クラス

項番	アセンブリ	リソースファイル	対応する「メッセージリソース」クラス
1	Terasoluna.dll	Resources.resx	Terasoluna.Properties.MessageResources
2	Terasoluna.Validation.dll	Resources.resx	Terasoluna.Validation.Properties.MessageResources
3		ValidationResources.resx	Terasoluna.Validation.Properties.ValidationMessageResources
4	Terasoluna.Windows.Forms.dll	DisplayResources.resx	Terasoluna.Windows.Forms.Properties.DisplayMessageResources
5		Resources.resx	Terasoluna.Windows.Forms.Properties.MessageResources
6	Terasoluna.Windows.ViewModel.Validation.dll	DisplayResources.resx	Terasoluna.Windows.ViewModel.Validation.Properties.DisplayMessageResources
7		Resources.resx	Terasoluna.Windows.ViewModel.Validation.Properties.MessageResources
8	Terasoluna.Server.ServiceModel.dll	DisplayResources.resx	Terasoluna.Server.ServiceModel.Properties.DisplayMessageResources
9		Resources.resx	Terasoluna.Server.ServiceModel.Properties.MessageResources

フレームワークが保持するリソースファイルの名称は、管理するメッセージの種類によって、以下の3つに分類される。

- Resources.resx
  - フレームワークが出力するログ等、開発者向けのメッセージを定義する。
- DisplayResources.resx
  - ダイアログ表示用のデフォルトメッセージなど、画面表示に利用されるメッセージを定義する。
- ValidationResources.resx
  - 入力値検証エラー時のメッセージを定義する。

このうち、DisplayResources.resx と ValidationResources.resx は、アプリケーションの利用者が画面を通して見るメッセージであるため、プロジェクトの UI 規約等に基づき適切なメッセージに差し替えたいというケースが考えられる。メッセージの差し替えを実施するには、表 1 に記述した各リソースファイルに対応する「メッセージリソース」クラスを使用する。
各「メッセージリソース」クラスには static なプロパティとして、MessageResourceManager 型の Manager プロパティが定義されている。
MessageResourceManager クラスは、フレームワークが管理する「メッセージリソース(旧)」と、差し替え対象の「メッセージリソース(新)」を管理し、差し替え対象の「メッセージリソース(新)」で定義されているメッセージを優先的に使用する。以下に、MessageResourceManager クラスの主要なプロパティを示す。

表 2　MessageResourceManager の主要なプロパティ

項番	プロパティ	説明
1	public Type CustomResourceType { get; set; }	差し替え対象の「メッセージリソース」のリソース型を指定する。指定されたリソース型を引数に持つ System.Resources.ResourceManager クラスが生成される。 未指定時は、フレームワークが提供するデフォルト「メッセージリソース」のリソース型がそのまま利用される。
2	public virtual CultureInfo UICulture { get; set; }	UI 用のカルチャ(System.Resources.ResourceManager クラスが、カルチャ固有のリソースを検索するために使用)を設定する。 未指定時は、現在のスレッドに設定されている UI カルチャ(CultureInfo.CurrentUICulture)を参照する。
3	public virtual CultureInfo Culture { get; set; }	日付、時刻、通貨、および数値の各既定形式と、テキスト並べ替え順序、文字列比較、大文字小文字の区別の決定等において使用されるカルチャ情報を設定する。 未指定時は、現在のスレッドに設定されているカルチャ(CultureInfo.CurrentCulture)を参照する。

CustomResourceType プロパティを、開発者が作成した「メッセージリソース」のリソース型に設定することで、デフォルトのメッセージを差し替えることができる。差し替えた「メッセージリソース」で変更の定義をしていないメッセージに関しては、フレームワークが提供するデフォルトのメッセージが使用されるため、開発者は修正対象のメッセージのみを新たに定義すればよい。
UICulture プロパティや Culture プロパティは、対象「メッセージリソース」のカルチャだけを変更

したい場合に使用する。通常は、アプリケーション全体でカルチャを切り替えるので、前述の「Thread.CurrentThread.CurrentUICulture」、「Thread.CurrentThread.CurrentCulture」でカルチャを設定する。
以下に、フレームワークが管理するメッセージを上書きする例を示す。
フレームワークの標準機能では、「CL-03 イベント処理実行機能」の入力値検証エラー発生時に、エラーダイアログに「入力に誤りがあります。」というメッセージが表示される。
また、エラー対象のコントロールには、ErrorProvider により、「CM-05 入力値検証機能」の検証エラーメッセージが表示される。この例では、必須入力エラーに対応するデフォルトエラーメッセージ、「"{2}"は、入力必須項目です。」が表示されている。

図 5 入力値検証エラー時に表示されるエラーダイアログのデフォルトメッセージ

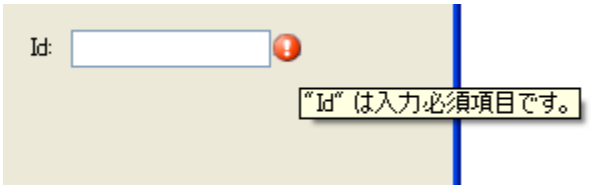

図 6 必須入力チェックエラーのデフォルトメッセージ

これらのメッセージは、以下のリソースファイルで管理されている。

- 入力値検証エラー時に表示されるエラーダイアログのデフォルトメッセージ
  - Terasoluna.Windows.ViewModel.Validation.dll の DisplayResources.resx
  - メッセージの名前は、「ValidationErrorNotificationMessage」
  - メッセージの値は、「入力に誤りがあります。」
- 必須入力チェックエラーのデフォルトメッセージ
  - Terasoluna.Validation.dll の ValidationResources.resx
  - メッセージの名前は、「ErrorRequiredValidatorIsRequired」
  - メッセージの値、「"{2}"は入力必須項目です。」

このサンプルでは、上記メッセージを、下記カスタムメッセージに差し替える。

- 入力値検証エラー時に表示されるエラーダイアログのデフォルトメッセージ
  - 開発者が作成したリソースファイル CustomDisplayResources.resx
  - メッセージの名前は、「ValidationErrorNotificationMessage」
  - メッセージの値は、「入力チェックエラーです。」

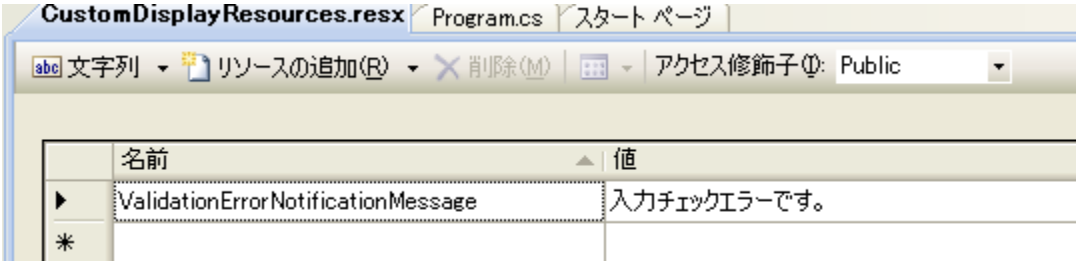

図 7　リソースファイルの作成例（CustomDisplayResources.resx）

- 必須入力チェックエラーのデフォルトメッセージ
  - 開発者が作成したリソースファイル CustomValidationResources.resx
  - メッセージの名前は、「ErrorRequiredValidatorIsRequired」
  - メッセージの値、「"{2}"は必ず入力してください。」

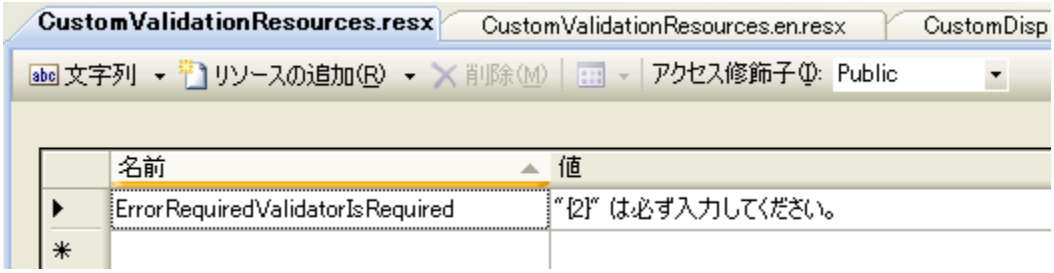

図 8 リソースファイルの作成例（CustomValidationResources.resx）

以下に、メッセージの差し替えの実装例を示す。
通常、アプリケーション起動直後のタイミングで、差し替え処理を実施する。

```
static class Program
{
    [STAThread]
    static void Main()
    {
        ///ValidationResources.resxのメッセージを上書き
        Terasoluna.Validation.Properties.ValidationMessageResources.
            Manager.CustomResourceType = typeof(CustomValidationResources);
        ///DisplayResources.resxのメッセージを上書き
        Terasoluna.Windows.ViewModel.Validation.Properties.
            DisplayMessageResources.Manager.CustomResourceType =
                typeof(CustomDisplayResources);

        Application.Run(new StartupForm());
    }
}
```

リスト 3 メッセージの差し替え処理の実装例

以下に、サンプルの実行結果を示す。
変更後のメッセージが画面に反映されているのが分かる。

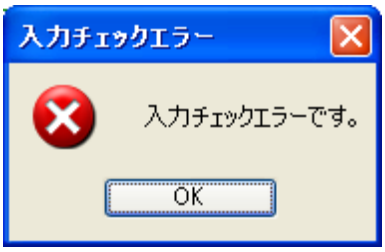

図 9 入力値検証エラー時に表示されるエラーダイアログの変更後のメッセージ

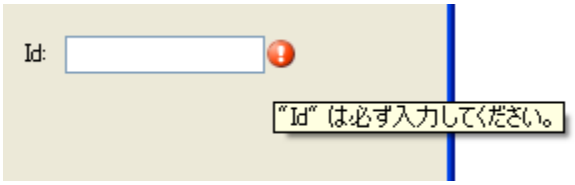

図 10 必須入力チェックエラーの変更後のメッセージ

次に、ユーザインタフェースカルチャが”en”(英語)、カルチャ情報が、”en-US”(英語-米国)のカスタムメッセージに切り替える例を示す。

- 入力値検証エラー時に表示されるエラーダイアログのデフォルトメッセージ
  - 開発者が作成したリソースファイル CustomDisplayResources.en.resx
  - メッセージの名前は、「ValidationErrorNotificationMessage」
  - メッセージの値は、「Validation Error!」

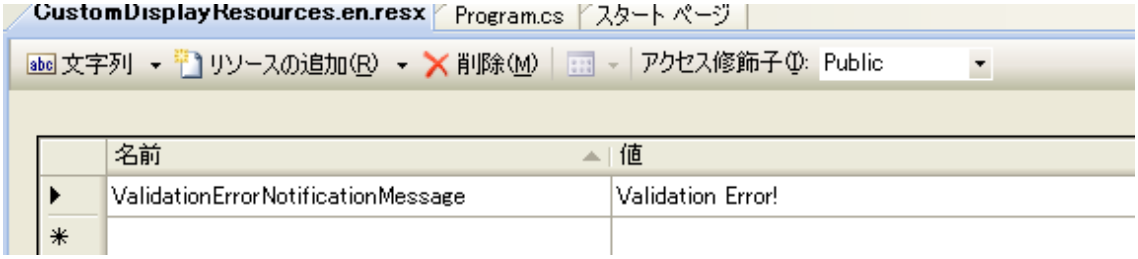

図 11 リソースファイルの作成例(CustomDisplayResources.en.resx)

- 必須入力チェックエラーのデフォルトメッセージ
  - 開発者が作成したリソースファイル CustomValidationResources.en.resx
  - メッセージの名前は、「ErrorRequiredValidatorIsRequired」
  - メッセージの値、「”{2}” is required.」

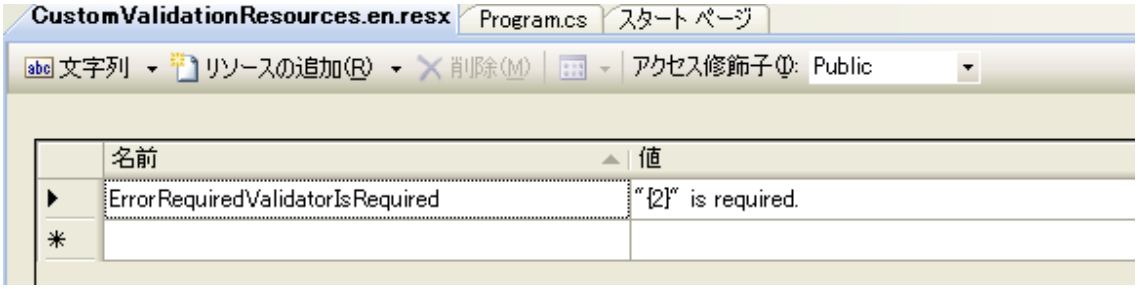

図 12 リソースファイルの作成例（CustomValidationResources.en.resx）

以下に、メッセージのカルチャの切り替えの実装例を示す。

```
static class Program
{
    [STAThread]
    static void Main()
    {
・・・
        ///デフォルトメッセージの切り替え
        Terasoluna.Validation.Properties.ValidationMessageResources.
            Manager.CustomResourceType = typeof(CustomValidationResources);
        Terasoluna.Windows.ViewModel.Validation.Properties.
            DisplayMessageResources.Manager.CustomResourceType =
                typeof(CustomDisplayResources);

        ///アプリケーション全体のカルチャの切り替え
        Thread.CurrentThread.CurrentUICulture = new CultureInfo("en-US");
        Thread.CurrentThread.CurrentCulture = new CultureInfo("en-US");

        ///各フレームワーク「メッセージリソース」を個別でカルチャ切り替える場合
        /*
        Terasoluna.Validation.Properties.ValidationMessageResources
            .Manager.UICulture = new CultureInfo("en-US");
        Terasoluna.Validation.Properties.ValidationMessageResources
            .Manager.Culture = new CultureInfo("en-US");

        Terasoluna.Windows.ViewModel.Validation.Properties.
            DisplayMessageResources.Manager.UICulture = new CultureInfo("en-US");
        Terasoluna.Windows.ViewModel.Validation.Properties.
            DisplayMessageResources.Manager.Culture = new CultureInfo("en-US");
        */
        Application.Run(new StartupForm());
    }
}
```

フレームワークメッセージのカルチャ切り替え処理の実装例

以下に、サンプルの実行結果を示す。
画面に設定したユーザインタフェースカルチャ固有のリソースでメッセージが表示されているのが分かる。
なお、今回の例では対象メッセージに日付や金額などの書式情報が含まれないため、Culture プロパティの設定は実行結果には反映されてはいない。

図 13 入力値検証エラー時に表示されるエラーダイアログのカルチャ切り替え後のメッセージ

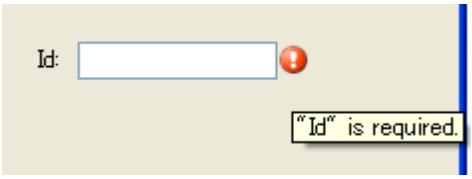

図 14 必須入力チェックエラーのカルチャ切り替え後のメッセージ

◆ 構成クラス

以下に、本機能を構成するファイルを示す。

表 3 構成クラス一覧

項番	クラス名	説明
Terasoluna.Text 名前空間		
1	MessageBuilder	フレームワーク内で使用するメッセージ文字列を作成するクラス。
2	MessageResource	1つの「メッセージリソース」にアクセスするクラス。
3	MessageResourceManager	「メッセージリソース」を管理するクラス。
Terasoluna.Properties 名前空間		
4	MessageResources	同名前空間上の Resource.resx を管理する「メッセージリソース」「メッセージリソース」クラス。
Terasoluna.Validation. Properties 名前空間.		
5	MessageResources	同名前空間上の Resource.resx を管理する「メッセージリソース」「メッセージリソース」クラス。
6	ValidationMessageResources	同名前空間上の ValidationResources.resx を管理する「メッセージリソース」「メッセージリソース」クラス。

Terasoluna.Windows.Forms.Properties 名前空間		
7	DisplayMessageResources	同名前空間上の DisplayResources.resx を管理する「メッセージリソース」「メッセージリソース」クラス。
8	MessageResources	同名前空間上の Resources.resx を管理する「メッセージリソース」「メッセージリソース」クラス。
Terasoluna.Windows.ViewModel.Validation.Properties 名前空間		
9	DisplayMessageResources	同名前空間上の DsiplayResources.resx を管理する「メッセージリソース」「メッセージリソース」クラス。
Terasoluna.Windows.ViewModel.Validation.Properties 名前空間		
10	MessageResources	同名前空間上の Resource.resx を管理する「メッセージリソース」「メッセージリソース」クラス。

■ 関連機能

特になし。