I18N.DotNet

main@d574fb

Generated by Doxygen 1.9.5

Chapter 1

I18N.DotNet

Documentation in PDF format is available here.

1.1 About

I18N.DotNet is a .NET library written in C# to enable simple internationalization (I18N) / localization (L10N) (i.e. translation to different languages) of .NET applications and libraries.

The companion utility I18N.DotNet Tool is provided to ease management of translation files.

1.2 Installation

The easiest way to install I18N.DotNet is using the NuGet package: https://www.nuget.org/packages/I18N.DotNet/

1.3 Getting Started

To use the I18N.DotNet library, three steps must be followed:

1. Write/modify the source code to internationalize strings that must be translated (see Writing/Adapting Source Code (I18N)).
2. Write translations for internationalized strings (see Writing Translations (L10N)).
3. Embed the translations file in the executable (see Embedding the Translations File).

1.3.1 Writing/Adapting Source Code (I18N)

When writing internationalized source code, the strings to be translated must be wrapped with a call to I18N.DotNet.GlobalLocalizer.Localize().

The easier and most convenient approach for writing internationalized software is to choose a language that will be used as the base language throughout the software development (e.g., English), and then write the software just as any non-internationalized source code, except that strings to be translated must be wrapped with calls to Localize(). This way the base language will act as the default language when translations are not available for the current target language.

Adapting existing non-internationalized source code is as easy as wrapping the existing strings to be translated with calls to Localize().

Example (C#)
```
using static I18N.DotNet.GlobalLocalizer;
using System;
using System.IO;
public class Program
{
  static void Main( string[] args )
  {
    int i = 0x555;
    Console.WriteLine( Localize( "Plain string to be translated" ) );
    Console.WriteLine( Localize( $"Interpolated string to be translated with value {i:X4}" ) );
  }
}
```

1.3.2 Writing Translations (L10N)

String translations must be stored in an XML file (the translations file) with root element `I18N`.

For each string than has been internationalized an `Entry` element under the root must be defined, with:

- A single `Key` child element which value is the internationalized string defined in the code (replacing for interpolated strings the interpolated expressions with their positional index).
- Several `Value` child elements with their attribute `lang` set to the target language of the translation and which value is the translated string.

It is not necessary to add translations (i.e., `Value` elements) for the development base language, since the value of the `Key` element will be used as the default translation when a translation for a specific language is not found.

Example
```
<?xml version="1.0" encoding="utf-8"?>
<I18N>
  <Entry>
    <Key>Plain string to be translated</Key>
    <Value lang="es">String simple a traducir</Value>
    <Value lang="fr">String simple à traduire</Value>
  </Entry>
  <Entry>
    <Key>Interpolated string to be translated with value {0:d}</Key>
    <Value lang="es">String interpolado a traducir con valor {0:dd/MM/yy}</Value>
    <Value lang="fr">String interpolé à traduire avec valeur {0:d}</Value>
  </Entry>
</I18N>
```

NOTE: The companion utility `I18N.DotNet Tool` can be used to ease the creation of the translations file by scanning source files and automatically generating entries for discovered internationalized strings.

1.3.3 Embedding the Translations File

A very convenient way of distributing the translations for an application is to embedded the translations file in the executable assembly as an embedded resource identified by *Resources.I18N.xml*.

Using Visual Studio, the easiest way to achieve this is to deploy the translations file as a file named _"I18N.xml"_ in a directory named _"Resources"_ inside the VS project directory, and then configure the file in the VS project as an embedded resource (i.e., set its Build Action to "Embedded resource" in the IDE, or add `<EmbeddedResource Include="Resources\I18N.xml" />` to an `ItemGroup` in the project file).

Example (.csproj)
```
<Project Sdk="Microsoft.NET.Sdk">
  ...
  <ItemGroup>
    <EmbeddedResource Include="Resources\I18N.xml" />
  </ItemGroup>
  ...
</Project>
```

> **NOTE**: The companion utility `I18N.DotNet Tool` can be used to generate translations files optimized for deployment from the separate translations files used during development and during the translation process.

1.4 Advanced Usage (Internationalizing Applications)

1.4.1 Global Localizer

The static class `GlobalLocalizer` has the property `Localizer` which contains the global localizer. This instance is shared and can be conveniently used by all software components. In fact all the methods exposed by the `Global↩ Localizer` class are just convenience wrappers that call the global localizer.

The property `GlobalLocalizer.Localizer` is an instance of `AutoLoadLocalizer` that on first usage (if translations have not been previously loaded) tries to load the translations from an embedded resource identified by _"Resources.I18N.xml"_ inside the entry (application) assembly using the current UI language as the target language.

The default behavior is just right for most use cases, but if the translations file is stored in an embedded resource with a different identifier, or in a separate file (e.g., installed alongside the application executable), one of the `LoadXML` methods can be invoked on the global localizer to load it (see Loading Translations).

Non-Default usage Example (C#)
```
void SetupI18N( string language, string directoryPath )
{
  GlobalLocalizer.Localizer.LoadXML( directoryPath + "/I18N.xml", language );
}
```

1.4.2 Local Localizers

Instances of `Localizer` can be created (local localizers), loaded with string translations, and then passed to software components for being used instead of the global localizer.

For most cases using the global localizer (and optionally contexts) is just enough, but local localizers can be useful for example to implement report generation in different languages than the application UI language (see Loading Translations and Specifying the Translation Target Language).

Example (C#)
```
Report GenerateReport( string language )
{
  var reportLocalizer = new Localizer();
  reportLocalizer.LoadXML( Assembly.GetExecutingAssembly(), "Reports.I18N.xml", language )
  return GenerateReport( reportLocalizer );
}
Report GenerateReport( ILocalizer localizer )
{
  var report = new Report();
  report.AddEntry( localizer.Localize( $"Date: {DateTime.Now:d}" ) );
  ...
  return report;
}
```

1.4.3 String Format

Calls to `String.Format()` where the format string has to be internationalized should be replaced by a call to `GlobalLocalizer.LocalizeFormat()` / `ILocalizer.LocalizeFormat()` instead of just internationalizing the format string.

Strings formatted using `LocalizeFormat()` and interpolated strings localized using `Localize()` are conveniently formatted using the formatting conventions of the localizer's target language/culture.

> Strings formatted using `String.Format()`, when no explicit `format provider` is used, will use the format conventions of the `default culture`, which may be different from the localizer's target culture, and can therefore produce unexpected localization results.

Example (C#)
```
String.Format( Localize( "Format string to be translated with value {0}" ), myVar );
// should be replaced by
LocalizeFormat( "Format string to be translated with value {0}", myVar );
// which is equivalent to
Localize( $"Format string to be translated with value {myVar}" );
```

1.4.4 Language Identifiers & Variants

Any arbitrary string can be used for identifying languages, although it is recommended to use identifiers formed by a ISO 639-1 alpha-2 language name (2-letter language codes, e.g., _"en"_, _"es"_), additionally followed by an hyphen and a ISO 3166-1 alpha-2 country/region name (e.g., _"en-US"_, _"es-ES"_). Specifically, it is recommended to use one of the `language/region names supported by Windows`.

Language identifiers are processed as case-insensitive (i.e., _"fr-FR"_ is equivalent to _"fr-fr"_).

When using language identifiers formed by a primary code and a variant code separated by an hyphen (e.g., _"en-us"_, _"es-es"_), if a localized conversion for the language variant is not found then a conversion for the primary (base) language is tried too.

For example, when loading the translations on a `Localizer` created for the _"en-gb"_ language, for each string to be translated a translation for the language _"en-gb"_ will be searched first, and if not found then a translation for the language _"en"_ will be searched next.

It is therefore recommended to:

- In source code:
  - Use primary-variant code (e.g., _"en-us"_, _"es-es"_) as target language identifiers (e.g., as `string` arguments to the `LoadXML` methods) or when `obtaining target cultures` (e.g., to use as `CultureInfo` arguments to the `LoadXML` methods).
- In translation files:
  - Use primary code (e.g., _"en"_, _"fr"_) as translation language identifiers (i.e, as the `lang` attribute values of XML `I18N.Entry.Value` entries) for generic (non variant-specific) translations.
  - Use primary code-variant (e.g., _"en-gb"_, _"es-ar"_) as translation language identifiers (i.e, as the `lang` attribute values of XML `I18N.Entry.Value` entries) for variant-specific translations.

1.4.5 Contexts

Sometimes the same source language string has different translations in different contexts (e.g., English _"OK"↩_ should be translated in Spanish to _"Aceptar"_ for a button label but to _"Correcto"_ for a successful outcome indication).

Since the source language key is the same in both cases, context partitioning must be used, which affects the source code side and the translations file side.

1.4.5.1 Context Partitioning in Source Code (I18N)

In source code, the context of the key can be explicitly indicated when the string is being internationalized by calling `GlobalLocalizer.Context()` / `ILocalizer.Context()` and passing it the context identifier, and then calling the localization methods on the returned context (which is an `ILocalizer`).

Contexts can be nested. A chain of successively nested contexts can be identified by joining their identifiers using the dot character ('.') as a composite context identifier.

Translations in a context are searched hierarchically: if a translation is not found for the target language in a context (neither for the language variant nor the primary language), then a translation is searched again on its parent context (if it exists). Finally, if no translation is found in the context hierarchy, then the base language translation is used (i.e., the value of the argument passed to the `Localize` method).

Example (C#)
```
Button.Label = Context( "GUI.Button" ).Localize( "OK" );
// ...
TextBox.Text = Context( "GUI" ).Context( "Status" ).Localize( "OK" );
```

1.4.5.2 Context Partitioning in the Translation File (L10N)

Context partitioning is performed in the translations XML file using `Context` elements as children of the root element or nested within other `Context` elements. These elements must have an `id` attribute to indicate the context identifier (which can be a composite context identifier), and are containers for the `Entry` elements that define the translations for that context.

Example
```
<?xml version="1.0" encoding="utf-8"?>
<I18N>
  <Entry>
    <Key>OK</Key>
    <Value lang="fr">O.K.</Value>
  </Entry>
  <Context id="GUI">
    <Context id="Button">
      <Entry>
        <Key>OK</Key>
        <Value lang="es">Aceptar</Value>
      </Entry>
    </Context>
    <Context id="Status">
      <Entry>
        <Key>OK</Key>
        <Value lang="es">Correcto</Value>
      </Entry>
    </Context>
  </Context>
</I18N>
```

1.4.6 Loading Translations

The translations can be (re)loaded into a localizer implementing `ILoadableLocalizer` by different ways:

1.4.6.0.1 Automatically The global localizer and `AutoLoadLocalizer` instances load their translations automatically from an embedded resource when any of their localization methods (those defined in `ILocalizer`) is called (only if translations have not been previously loaded explicitly).

1.4.6.0.2 From an Embedded Resource A very convenient way of using translation files is to embed them into an executable assembly (application or library), then load them into a localizer implementing `ILoadableLocalizer` using a `LoadXML` method passing as arguments the assembly to load the embedded resource from and its identifier (and optionally the target language for translations).

Example (C#)
```
void SetupI18N()
{
  GlobalLocalizer.Localizer.LoadXML( Assembly.GetExecutingAssembly(), "I18N.Translations.xml" );
}
```

1.4.6.0.3 From a Standalone File If the translations file is stored as a separate file (e.g., installed alongside the application executable), a `LoadXML` method can be invoked on a localizer implementing `ILoadableLocalizer` passing the path to the file as an argument (and passing optionally the target language for translations).

Example (C#)
```
void SetupI18N()
{
  var programPath = Path.GetDirectoryName( Assembly.GetExecutingAssembly().Location );
  GlobalLocalizer.Localizer.LoadXML( programPath + "/I18N.xml" );
}
```

1.4.6.0.4 From a Stream When the translations file is neither stored as a file nor as an embedded resource (e.g., downloading the translations from a remote server to local memory, obtaining the translations from a database), a `LoadXML` method can be invoked on an `ILoadableLocalizer` instance passing as an argument a `System.IO.Stream` object that must provide the file contents (and passing optionally the target language for translations).

1.4.6.0.5 From an XML Document Additionally, translations can be loaded from an XML document by invoking a `LoadXML` method on an `ILoadableLocalizer` instance passing as an argument a `System.Xml.Linq.XDocument` object loaded with the translations (and passing optionally the target language for translations).

1.4.7 Specifying the Translation Target Language

The target language and associated culture for formatting operations used by localizers (e.g., the global localizer or a local localizer) is selected only when translations are loaded on the localizer.

When translations are loaded (automatically or by means of explicit calls to `LoadXML` methods), if an explicit target language (or culture) is not indicated, then the current UI language (obtained from `System.Globalization.CultureInfo.CurrentUICulture`) is used by default as the target language (and culture).

Selecting a different language (or culture) than the default can be achieved by different ways:

1.4.7.0.1 Changing the UI Culture During Startup An easy way of changing the target language is, during application startup, before any localization method is called, to set `System.Globalization.CultureInfo.CurrentUICulture` to the culture corresponding to the desired target language.

> Automatically-loading localizers do not select their default language when they are created, but they instead delay this selection until they load their translations, which will not happen automatically until a localization method is called. Therefore the default UI culture can be changed after these localizers have been created, and the new culture will still be used properly when localization methods are called for the first time.

This approach to make the global localizer use a specific language (e.g., use a language configured by the user) is very simple, and it has the advantage that resources localized by other means may probably also use the same target language and culture.

Example (C#)
```
using System.Globalization;
public class Program
{
  static void Main( string[] args )
  {
    if( args.Length >= 1 )
    {
      CultureInfo.CurrentUICulture = CultureInfo.GetCultureInfo( args[0] );
    }
    ...
  }
}
```

> **NOTE**: It may also be useful to set `System.Globalization.CultureInfo CurrentCulture`, `System.Globalization.CultureInfo DefaultThreadCurrentUICulture`, and/or `System.Globalization.CultureInfo DefaultThreadCurrentCulture`.

1.4.7.0.2 Changing the UI Culture Dynamically

When the application is already running, changing the UI culture will have no immediate effect on the localizers which translations have already been loaded.

Automatically-loading localizers (i.e., instances of `AutoLoadLocalizer`, like the global localizer) can be manually forced to reload its translations to enforce dynamic changes of the UI culture to take effect.

Example (C#)
```
void SetupI18N( string language )
{
  CultureInfo.CurrentUICulture = CultureInfo.GetCultureInfo( language );
  GlobalLocalizer.Localizer.Load( null );
}
```

1.4.7.0.3 Automatically-Loading Localizers Translations (Re)Load

The `AutoLoadLocalizer` class provides `Load` methods that accept the target language as a language identifier or culture (see TargetCulture) parameter.

The `AutoLoadLocalizer.Load` methods can be called during application startup or during runtime to (re)load the translations from the embedded resource for a specific language.

Example (C#)
```
void SetupI18N( CultureInfo culture )
{
  GlobalLocalizer.Localizer.Load( culture );
}
```

1.4.7.0.4 Loadable Localizers Translations (Re)Load

The `ILoadableLocalizer` interface defines `LoadXML` methods that accept the target language as an optional language identifier or culture (see TargetCulture) parameter.

The `ILoadableLocalizer.LoadXML` methods can be called during application startup or during runtime to (re)load the translations for a specific language.

Example (C#)
```
void SetupI18N( string language )
{
  var programPath = Path.GetDirectoryName( Assembly.GetExecutingAssembly().Location );
  GlobalLocalizer.Localizer.LoadXML( programPath + "/I18N.xml", language );
}
```

1.4.7.1 Target Culture

When selecting the target language for translations, it can be specified either with a language identifier (using a `string`) or with a culture (using a `CultureInfo`).

When specifying the target language with a language identifier, the target culture is automatically set to the culture associated in the system with the language identifier (obtained using `System.Globalization.CultureInfo.GetCultureInfo`). If the system does not support a culture for the target language, then the `invariant culture` will be used.

The localizer's target culture is used during formatting operations.

1.5 Advanced Usage (Internationalizing Libraries)

1.5.1 Library Localizers

The global localizer is convenient for usage in applications (i.e., which are implemented in the entry assembly), but libraries should not use the global localizer because they would depend on the application to load the translations for its internationalized strings, or risk the application discarding the translations after the library has merged its own translations automatically during library initialization.

For libraries the easiest solution is to define their own "global" localizer as a static property inside a static class, similar to the `GlobalLocalizer` class but only intended for the scope of the library.

This library localizer can be initialized using an instance of `AutoLoadLocalizer`, which is a special localizer that automatically loads the translations file from an embedded resource.

The static class can be declared with `internal` scope, or better with `public` scope to allow applications to access the library localizer (e.g., to add more translations, change them, reload them, etc.).

Finally, the translations file for the library must be embedded in the library assembly as an embedded resource identified by *Resources.I18N.xml* (just like with an application), which the `AutoLoadLocalizer` instance will try to load by default.

Library Localizer Implementation Example (C#) `using I18N.DotNet;`

```
using System;
namespace ExampleLibrary
{
  public static class LibraryLocalizer
  {
    public static ILocalizer Localizer { get; } = new AutoLoadLocalizer();
    internal static string Localize( PlainString text ) => Localizer.Localize( text );
    internal static string Localize( FormattableString text ) => Localizer.Localize( text );
  }
}
```

Library Localizer Usage Example (C#) `using static ExampleLibrary.LibraryLocalizer;`

```
using System;
namespace ExampleLibrary
{
  public class ExampleClass
  {
    public void SomeMethod()
    {
      Console.WriteLine( Localize( "Plain string to be translated" ) );
      Console.WriteLine( Localize( $"Interpolated string to be translated with value {i:X4}" ) );
    }
  }
}
```

1.6 API Documentation

1.6.0.1 ILocalizer Interface

The `ILocalizer` interface represents classes which provide localization functionality to software components (i.e. perform string translations) for a single target language:

- `Localize` methods to translate strings, interpolated strings and collections of strings.
- `LocalizeFormat` method to format and translate strings.
- `Context` methods to access contexts and subcontexts (see Contexts).

1.6.0.2 ILoadableLocalizer Interface

The `ILoadableLocalizer` interface is an extension of `ILocalizer` that represents localizer classes which provide functionality to load translations for a single target language from different sources:

- `LoadXML` method to load translations from a file in the filesystem.
- `LoadXML` method to load translations from a `Stream`.
- `LoadXML` method to load translations from an XML document (`XDocument`).
- `LoadXML` method to load translations from an embedded resource in an assembly.

1.6.0.3 Localizer Class

The `Localizer` class is a simple implementation of `ILoadableLocalizer` which is capable of loading string translations for a single target language and then providing localization functionality.

1.6.0.4 AutoLoadLocalizer Class

The `AutoLoadLocalizer` class is an implementation of `ILoadableLocalizer` that on first call of any of its localization methods (i.e., those specified by `ILocalizer`), loads automatically the translations from an embedded resource in an assembly using the current UI language as the target language (if translations have not been previously loaded).

The default parameters for the `AutoLoadLocalizer` constructor make the created instance load the translations file from an embedded resource identified by *Resources.I18N.xml* in the calling assembly (i.e., in the assembly that creates the instance).

A different resource identifier or assembly can be passed as parameters to the `AutoLoadLocalizer` constructor if necessary.

Additionally, this class provides:

- `Load` method to load/reload translations from the configured embedded resource for a given language.

1.6.0.5 GlobalLocalizer Class

The `GlobalLocalizer` static class provides access to the global localizer:

- `Localizer` static property that provides the global localizer as a localizer instance.
- `Localize` static methods to translate strings, interpolated strings and collections of strings.
- `LocalizeFormat` static method to format and translate strings.
- `Context` static methods to access contexts and subcontexts (see Contexts).

1.6.1 Full API Documentation

You can browse the full API documentation for:

- The last release (stable)
- Main branch (unstable)

Chapter 2

Namespace Index

2.1 Package List

Here are the packages with brief descriptions (if available):

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

Namespace Documentation

5.1 I18N Namespace Reference

Namespaces

- namespace DotNet

5.2 I18N.DotNet Namespace Reference

Classes

- class AutoLoadLocalizer

  Implementation of a localizer which configuration is automatically loaded from an embedded resource.
- class ContextLocalizer

  Localizer that can provide translations and can store nested contexts.
- class GlobalLocalizer

  Utility class for convenient access to localization functions.
- interface ILoadableLocalizer

  Localizer which translations can be loaded from different sources.
- interface ILocalizer

  Converter of strings from a base-language value to its corresponding language-specific localization.
- class **Language**

  Represents a language for localization purposes.
- class Localizer

  Simple loadable localizer.
- class PlainString

  Represents just a string. This class is used to allow interpolated strings to preferably be passed as FormattableString instead of string to methods that overload both types.

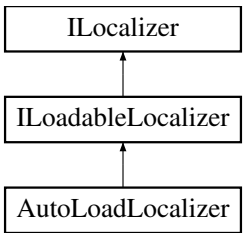
Chapter 6

Class Documentation

6.1 AutoLoadLocalizer Class Reference

Implementation of a localizer which configuration is automatically loaded from an embedded resource.

Inheritance diagram for AutoLoadLocalizer:



Public Member Functions

- AutoLoadLocalizer (string resourceName=DEFAULT_RESOURCE_NAME, Assembly? assembly=null)

  Constructor.
- string Localize (PlainString text)

  Localizes a string.
  Converts the base-language string text to its corresponding language-specific localized value.
- string Localize (FormattableString frmtText)

  Localizes an interpolated string.
  Converts the composite format string of the base-language formattable string frmtText (e.g. an interpolated string) to its corresponding language-specific localized composite format value, and then generates the result by formatting the localized composite format value along with thefrmtText arguments by using the formatting conventions of the localizer culture.
- IEnumerable< string > Localize (IEnumerable< string > texts)

  Localizes multiple strings.
  Converts the base-language strings in texts to their corresponding language-specific localized values.
- string LocalizeFormat (string format, params object[ ] args)

  Localizes and then formats a string.
  Converts the base-language format string format to its corresponding language-specific localized format value, and then generates the result by formatting the localized format value along with theargs arguments by using the formatting conventions of the localizer culture.
- ILocalizer Context (string contextId)

Gets the localizer for a context in the current localizer.
Contexts are used to disambiguate the conversion of the same base-language string to different language-specific strings depending on the context where the conversion is performed.

- ILocalizer Context (IEnumerable< string > splitContextIds)

  Gets the localizer for a context in the current localizer.
  Contexts are used to disambiguate the conversion of the same base-language string to different language-specific strings depending on the context where the conversion is performed.

- void LoadXML (string filepath, CultureInfo? culture=null)

  Loads translations for the given culture from a localization configuration file in XML format.
  All the translations loaded previously in the localizer are discarded and replaced with the new ones.

- void LoadXML (string filepath, string language)

  Loads translations for the given language from a localization configuration file in XML format.
  All the translations loaded previously in the localizer are discarded and replaced with the new ones.

- void LoadXML (string filepath, bool merge)

  Loads translations for the current localizer language from a localization configuration file in XML format.

Parameters

filepath	*Path to the localization configuration file*
merge	*Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).*

Exceptions

ParseException	*Thrown when the input file cannot be parsed properly.*

- void LoadXML (Stream stream, CultureInfo? culture=null)

  Loads translations for the given culture from a localization configuration file in XML format obtained from a stream.
  All the translations loaded previously in the localizer are discarded and replaced with the new ones.

- void LoadXML (Stream stream, string language)

  Loads translations for the given language from a localization configuration file obtained in XML format from a stream.
  All the translations loaded previously in the localizer are discarded and replaced with the new ones.

- void LoadXML (Stream stream, bool merge)

  Loads translations for the current localizer language from a localization configuration file in XML format obtained from a stream.

Parameters

stream	*Stream with the localization configuration*
merge	*Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).*

Exceptions

ParseException	*Thrown when the stream contents cannot be parsed properly.*

- void LoadXML (XDocument doc, CultureInfo? culture=null)

  Loads translations for the given culture from a localization configuration in an XML document.
  All the translations loaded previously in the localizer are discarded and replaced with the new ones.

- void LoadXML (XDocument doc, string language)

  Loads translations for the given language from a localization configuration in an XML document.
  All the translations loaded previously in the localizer are discarded and replaced with the new ones.

- void LoadXML (XDocument doc, bool merge)

Loads translations for the current localizer language from a localization configuration in an XML document.

Parameters

doc	*XML document with the localization configuration*
merge	*Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).*

Exceptions

ParseException	*Thrown when the stream contents cannot be parsed properly.*

- void LoadXML (Assembly assembly, string resourceName, CultureInfo? culture=null)

  Loads translations for the given culture from a localization configuration file in XML format obtained from an embedded resource in the given assembly.
  All the translations loaded previously in the localizer are discarded and replaced with the new ones.
- void LoadXML (Assembly assembly, string resourceName, string language)

  Loads translations for the given language from a localization configuration file in XML format obtained from an embedded resource in the given assembly.
  All the translations loaded previously in the localizer are discarded and replaced with the new ones.
- void LoadXML (Assembly assembly, string resourceName, bool merge)

  Loads translations for the current localizer language from a localization configuration file in XML format obtained from an embedded resource in the given assembly.

Parameters

assembly	*Assembly that contains the embedded XML file*
resourceName	*Name of the embedded resource for the XML file*
merge	*Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).*

Exceptions

ParseException	*Thrown when the stream contents cannot be parsed properly.*
InvalidOperationException	*Thrown when the embedded resource could not be found in the given assembly.*

- void Load (CultureInfo? culture)

  Loads translations for the given culture from the embedded resource specified when creating the instance.
- void Load (string language)

  Loads translations for the given language from the embedded resource specified when creating the instance.

Static Public Attributes

- const string DEFAULT_RESOURCE_NAME = "Resources.I18N.xml"

  Default identifier for the embedded resource containing the translations.

Properties

- string TargetLanguage `[get]`

  Target language of the localizer.
- CultureInfo TargetCulture `[get]`

  Target culture of the localizer.

6.1.1 Detailed Description

Implementation of a localizer which configuration is automatically loaded from an embedded resource.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 AutoLoadLocalizer()

```
AutoLoadLocalizer (
            string resourceName = DEFAULT_RESOURCE_NAME,
            Assembly?  assembly = null )
```

Constructor.

When the localization methods are called for the first time, the translations are automatically loaded from the embedded resource identified by *resourceName* inside the given *assembly* (if translations have not been previously loaded explicitly).

Parameters

resourceName	Name of the embedded resource for the XML file
assembly	Assembly that contains the embedded XML file (the calling assembly will be used if `null`)

6.1.3 Member Function Documentation

6.1.3.1 Context() [1/2]

```
ILocalizer Context (
            IEnumerable< string > splitContextIds )
```

Gets the localizer for a context in the current localizer.

Contexts are used to disambiguate the conversion of the same base-language string to different language-specific strings depending on the context where the conversion is performed.

Implements ILocalizer.

6.1.3.2 Context() [2/2]

```
ILocalizer Context (
            string contextId )
```

Gets the localizer for a context in the current localizer.

Contexts are used to disambiguate the conversion of the same base-language string to different language-specific strings depending on the context where the conversion is performed.

Implements ILocalizer.

6.1.3.3 Load() [1/2]

```
void Load (
            CultureInfo?  culture )
```

Loads translations for the given *culture* from the embedded resource specified when creating the instance.

Parameters

culture	Culture for the target language of translations, or `null` to use the current UI culture (obtained from System.Globalization.CultureInfo.CurrentUICulture)

Exceptions

ILoadableLocalizer.ParseException	Thrown when the embedded resource contents cannot be parsed properly.
InvalidOperationException	Thrown when the embedded resource could not be found.

6.1.3.4 Load() [2/2]

```
void Load (
            string language )
```

Loads translations for the given *language* from the embedded resource specified when creating the instance.

Parameters

language	Name, code or identifier for the target language of translations

Exceptions

ILoadableLocalizer.ParseException	Thrown when the embedded resource contents cannot be parsed properly.
InvalidOperationException	Thrown when the embedded resource could not be found.

6.1.3.5 LoadXML() [1/12]

```
void LoadXML (
            Assembly assembly,
            string resourceName,
            bool merge )
```

Loads translations for the current localizer language from a localization configuration file in XML format obtained from an embedded resource in the given assembly.

Parameters

assembly	Assembly that contains the embedded XML file
resourceName	Name of the embedded resource for the XML file
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	Thrown when the stream contents cannot be parsed properly.
InvalidOperationException	Thrown when the embedded resource could not be found in the given assembly.

Implements ILoadableLocalizer.

6.1.3.6 LoadXML() [2/12]

```
void LoadXML (
            Assembly assembly,
            string resourceName,
            CultureInfo?  culture = null )
```

Loads translations for the given *culture* from a localization configuration file in XML format obtained from an embedded resource in the given assembly.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

Implements ILoadableLocalizer.

6.1.3.7 LoadXML() [3/12]

```
void LoadXML (
            Assembly assembly,
            string resourceName,
            string language )
```

Loads translations for the given *language* from a localization configuration file in XML format obtained from an embedded resource in the given assembly.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

Implements ILoadableLocalizer.

6.1.3.8 LoadXML() [4/12]

```
void LoadXML (
            Stream stream,
            bool merge )
```

Loads translations for the current localizer language from a localization configuration file in XML format obtained from a stream.

Parameters

stream	Stream with the localization configuration
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	Thrown when the stream contents cannot be parsed properly.

Implements ILoadableLocalizer.

6.1.3.9 LoadXML() [5/12]

```
void LoadXML (
            Stream stream,
            CultureInfo?  culture = null )
```

Loads translations for the given *culture* from a localization configuration file in XML format obtained from a stream.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

Implements ILoadableLocalizer.

6.1.3.10 LoadXML() [6/12]

```
void LoadXML (
            Stream stream,
            string language )
```

Loads translations for the given *language* from a localization configuration file obtained in XML format from a stream.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

Implements ILoadableLocalizer.

6.1.3.11 LoadXML() [7/12]

```
void LoadXML (
            string filepath,
            bool merge )
```

Loads translations for the current localizer language from a localization configuration file in XML format.

Parameters

filepath	Path to the localization configuration file
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	Thrown when the input file cannot be parsed properly.

Implements ILoadableLocalizer.

6.1.3.12 LoadXML() [8/12]

```
void LoadXML (
            string filepath,
            CultureInfo?  culture = null )
```

Loads translations for the given *culture* from a localization configuration file in XML format.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

Implements ILoadableLocalizer.

6.1.3.13 LoadXML() [9/12]

```
void LoadXML (
            string filepath,
            string language )
```

Loads translations for the given *language* from a localization configuration file in XML format.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

Implements ILoadableLocalizer.

6.1.3.14 LoadXML() [10/12]

```
void LoadXML (
            XDocument doc,
            bool merge )
```

Loads translations for the current localizer language from a localization configuration in an XML document.

Parameters

doc	XML document with the localization configuration
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	Thrown when the stream contents cannot be parsed properly.

Implements ILoadableLocalizer.

6.1.3.15 LoadXML() [11/12]

```
void LoadXML (
            XDocument doc,
            CultureInfo?  culture = null )
```

Loads translations for the given *culture* from a localization configuration in an XML document.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

Implements ILoadableLocalizer.

6.1.3.16 LoadXML() [12/12]

```
void LoadXML (
            XDocument doc,
            string language )
```

Loads translations for the given *language* from a localization configuration in an XML document.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

Implements ILoadableLocalizer.

6.1.3.17 Localize() [1/3]

```
string Localize (
            FormattableString frmtText )
```

Localizes an interpolated string.

Converts the composite format string of the base-language formattable string *frmtText* (e.g. an interpolated string) to its corresponding language-specific localized composite format value, and then generates the result by formatting the localized composite format value along with the*frmtText* arguments by using the formatting conventions of the localizer culture.

Implements ILocalizer.

6.1.3.18 Localize() [2/3]

```
IEnumerable< string > Localize (
            IEnumerable< string > texts )
```

Localizes multiple strings.

Converts the base-language strings in *texts* to their corresponding language-specific localized values.

Implements ILocalizer.

6.1.3.19 Localize() [3/3]

```
string Localize (
            PlainString text )
```

Localizes a string.

Converts the base-language string *text* to its corresponding language-specific localized value.

Implements ILocalizer.

6.1.3.20 LocalizeFormat()

```
string LocalizeFormat (
            string format,
            params object[] args )
```

Localizes and then formats a string.

Converts the base-language format string *format* to its corresponding language-specific localized format value, and then generates the result by formatting the localized format value along with the*args* arguments by using the formatting conventions of the localizer culture.

Implements ILocalizer.

6.1.4 Member Data Documentation

6.1.4.1 DEFAULT_RESOURCE_NAME

```
const string DEFAULT_RESOURCE_NAME = "Resources.I18N.xml"  [static]
```

Default identifier for the embedded resource containing the translations.

6.1.5 Property Documentation

6.1.5.1 TargetCulture

```
CultureInfo TargetCulture  [get]
```

Target culture of the localizer.

Implements ILocalizer.

6.1.5.2 TargetLanguage

```
string TargetLanguage  [get]
```

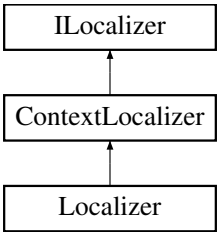
Target language of the localizer.

Implements ILocalizer.

6.2 ContextLocalizer Class Reference

Localizer that can provide translations and can store nested contexts.

Inheritance diagram for ContextLocalizer:



Public Member Functions

- string Localize (PlainString text)

  Localizes a string.
  Converts the base-language string text to its corresponding language-specific localized value.
- string Localize (FormattableString frmtText)

  Localizes an interpolated string.
  Converts the composite format string of the base-language formattable string frmtText (e.g. an interpolated string) to its corresponding language-specific localized composite format value, and then generates the result by formatting the localized composite format value along with thefrmtText arguments by using the formatting conventions of the localizer culture.
- string LocalizeFormat (string format, params object?[ ] args)
- IEnumerable< string > Localize (IEnumerable< string > texts)

  Localizes multiple strings.
  Converts the base-language strings in texts to their corresponding language-specific localized values.
- ContextLocalizer Context (string contextId)

  Gets the localizer for a context in the current localizer.
  Contexts are used to disambiguate the conversion of the same base-language string to different language-specific strings depending on the context where the conversion is performed.
- ContextLocalizer Context (IEnumerable< string > splitContextIds)

  Gets the localizer for a context in the current localizer.
  Contexts are used to disambiguate the conversion of the same base-language string to different language-specific strings depending on the context where the conversion is performed.
- string LocalizeFormat (string format, params object[ ] args)

  Localizes and then formats a string.

Protected Member Functions

- ContextLocalizer ()
- void Clear ()
- void Load (XElement element)

Properties

- string TargetLanguage `[get]`

  Target language of the localizer.
- CultureInfo TargetCulture `[get]`

  Target culture of the localizer.
- Language Language `[get, set]`

6.2.1 Detailed Description

Localizer that can provide translations and can store nested contexts.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 ContextLocalizer()

```
ContextLocalizer ( )  [protected]
```

6.2.3 Member Function Documentation

6.2.3.1 Clear()

```
void Clear ( )  [protected]
```

6.2.3.2 Context() [1/2]

```
ContextLocalizer Context (
            IEnumerable< string > splitContextIds )
```

Gets the localizer for a context in the current localizer.

Contexts are used to disambiguate the conversion of the same base-language string to different language-specific strings depending on the context where the conversion is performed.

Implements ILocalizer.

6.2.3.3 Context() [2/2]

```
ContextLocalizer Context (
            string contextId )
```

Gets the localizer for a context in the current localizer.

Contexts are used to disambiguate the conversion of the same base-language string to different language-specific strings depending on the context where the conversion is performed.

Implements ILocalizer.

6.2.3.4 Load()

```
void Load (
            XElement element )  [protected]
```

6.2.3.5 Localize() [1/3]

```
string Localize (
            FormattableString frmtText )
```

Localizes an interpolated string.

Converts the composite format string of the base-language formattable string *frmtText* (e.g. an interpolated string) to its corresponding language-specific localized composite format value, and then generates the result by formatting the localized composite format value along with the*frmtText* arguments by using the formatting conventions of the localizer culture.

Implements ILocalizer.

6.2.3.6 Localize() [2/3]

```
IEnumerable< string > Localize (
            IEnumerable< string > texts )
```

Localizes multiple strings.

Converts the base-language strings in *texts* to their corresponding language-specific localized values.

Implements ILocalizer.

6.2.3.7 Localize() [3/3]

```
string Localize (
            PlainString text )
```

Localizes a string.

Converts the base-language string *text* to its corresponding language-specific localized value.

Implements ILocalizer.

6.2.3.8 LocalizeFormat() [1/2]

```
string LocalizeFormat (
            string format,
            params object?[] args )
```

6.2.3.9 LocalizeFormat() [2/2]

```
string LocalizeFormat (
            string format,
            params object[] args )  [inherited]
```

Localizes and then formats a string.

Converts the base-language format string *format* to its corresponding language-specific localized format value, and then generates the result by formatting the localized format value along with the*args* arguments by using the formatting conventions of the localizer culture.

Parameters

format	Base-language format string
args	Arguments for the format string

Returns

Formatted string generated from the language-specific localized format string if found, or generated from *format* otherwise

Exceptions

FormatException	Thrown when *format* or its localized format value is invalid.

Implemented in AutoLoadLocalizer.

6.2.4 Property Documentation

6.2.4.1 Language

```
Language Language  [get], [set], [protected]
```

6.2.4.2 TargetCulture

```
CultureInfo TargetCulture  [get]
```

Target culture of the localizer.

Implements ILocalizer.

6.2.4.3 TargetLanguage

```
string TargetLanguage  [get]
```

Target language of the localizer.

Implements ILocalizer.

6.3 GlobalLocalizer Class Reference

Utility class for convenient access to localization functions.

Static Public Member Functions

- static string Localize (PlainString text)

  Localizes a string using the global localizer.
- static string Localize (FormattableString frmtText)

  Localizes an interpolated string using the global localizer.
- static IEnumerable< string > Localize (IEnumerable< string > texts)

  Localizes multiple strings.
- static string LocalizeFormat (string format, params object[] args)

  Localizes and then formats a string using the global localizer.
- static ILocalizer Context (string contextId)

  Gets a context in the global localizer.
- static ILocalizer Context (IEnumerable< string > splitContextIds)

  Gets a context in the global localizer.

Properties

- static AutoLoadLocalizer Localizer = new AutoLoadLocalizer() `[get]`

6.3.1 Detailed Description

Utility class for convenient access to localization functions.

6.3.2 Member Function Documentation

6.3.2.1 Context() `[1/2]`

```
static ILocalizer Context (
            IEnumerable< string > splitContextIds )  [static]
```

Gets a context in the global localizer.

See also

ILocalizer.Context(IEnumerable<string>)

Parameters

splitContextIds	Chain of context identifiers in split form

Returns

Localizer for the given context

6.3.2.2 Context() [2/2]

```
static ILocalizer Context (
            string contextId )  [static]
```

Gets a context in the global localizer.

See also

ILocalizer.Context(string)

Parameters

contextId	Identifier of the context

Returns

Localizer for the given context

6.3.2.3 Localize() [1/3]

```
static string Localize (
            FormattableString frmtText )  [static]
```

Localizes an interpolated string using the global localizer.

See also

ILocalizer.Localize(FormattableString)

Parameters

frmtText	Language-neutral formattable string

Returns

Formatted string generated from the language-specific localized format string if found, or generated from *frmtText* otherwise

6.3.2.4 Localize() [2/3]

```
static IEnumerable< string > Localize (
            IEnumerable< string > texts )  [static]
```

Localizes multiple strings.

See also

ILocalizer.Localize(IEnumerable<string>)

Parameters

texts	Array of language-neutral strings

Returns

Array with the language-specific localized strings if found, or the language-neutral string otherwise

6.3.2.5 Localize() [3/3]

```
static string Localize (
            PlainString text )  [static]
```

Localizes a string using the global localizer.

See also

ILocalizer.Localize(PlainString)

Parameters

text	Language-neutral string

Returns

Language-specific localized string if found, or *text* otherwise

6.3.2.6 LocalizeFormat()

```
static string LocalizeFormat (
            string format,
            params object[] args )  [static]
```

Localizes and then formats a string using the global localizer.

See also

ILocalizer.LocalizeFormat(string, object[ ])

Parameters

format	Language-neutral format string
args	Arguments for the format string

Returns

Formatted string generated from the language-specific localized format string if found, or generated from *format* otherwise

6.3.3 Property Documentation

6.3.3.1 Localizer

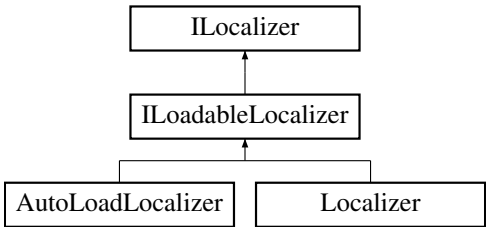
```
AutoLoadLocalizer Localizer = new AutoLoadLocalizer()  [static], [get]
```

Global localizer.

6.4 ILoadableLocalizer Interface Reference

Localizer which translations can be loaded from different sources.

Inheritance diagram for ILoadableLocalizer:



Classes

- class ParseException

  Exception thrown when a localization file cannot be parsed properly.

Public Member Functions

- void LoadXML (string filepath, CultureInfo? culture=null)

  Loads translations for the given culture from a localization configuration file in XML format.
- void LoadXML (string filepath, string language)

  Loads translations for the given language from a localization configuration file in XML format.
- void LoadXML (string filepath, bool merge)

  Loads translations for the current localizer language from a localization configuration file in XML format.
- void LoadXML (Stream stream, CultureInfo? culture=null)

  Loads translations for the given culture from a localization configuration file in XML format obtained from a stream.
- void LoadXML (Stream stream, string language)

  Loads translations for the given language from a localization configuration file obtained in XML format from a stream.
- void LoadXML (Stream stream, bool merge)

  Loads translations for the current localizer language from a localization configuration file in XML format obtained from a stream.
- void LoadXML (XDocument doc, CultureInfo? culture=null)

  Loads translations for the given culture from a localization configuration in an XML document.
- void LoadXML (XDocument doc, string language)

  Loads translations for the given language from a localization configuration in an XML document.
- void LoadXML (XDocument doc, bool merge)

  Loads translations for the current localizer language from a localization configuration in an XML document.
- void LoadXML (Assembly assembly, string resourceName, CultureInfo? culture=null)

  Loads translations for the given culture from a localization configuration file in XML format obtained from an embedded resource in the given assembly.
- void LoadXML (Assembly assembly, string resourceName, string language)

  Loads translations for the given language from a localization configuration file in XML format obtained from an embedded resource in the given assembly.
- void LoadXML (Assembly assembly, string resourceName, bool merge)

  Loads translations for the current localizer language from a localization configuration file in XML format obtained from an embedded resource in the given assembly.
- string Localize (PlainString text)

  Localizes a string.
- string Localize (FormattableString frmtText)

  Localizes an interpolated string.
- IEnumerable< string > Localize (IEnumerable< string > texts)

  Localizes multiple strings.
- string LocalizeFormat (string format, params object[] args)

  Localizes and then formats a string.
- ILocalizer Context (string contextId)

  Gets the localizer for a context in the current localizer.
- ILocalizer Context (IEnumerable< string > splitContextIds)

  Gets the localizer for a context in the current localizer.

Properties

- string TargetLanguage `[get]`

  Target language of the localizer.
- CultureInfo TargetCulture `[get]`

  Target culture of the localizer.

6.4.1 Detailed Description

Localizer which translations can be loaded from different sources.

6.4.2 Member Function Documentation

6.4.2.1 Context() [1/2]

```
ILocalizer Context (
            IEnumerable< string > splitContextIds )  [inherited]
```

Gets the localizer for a context in the current localizer.

Contexts are used to disambiguate the conversion of the same base-language string to different language-specific strings depending on the context where the conversion is performed.

Parameters

splitContextIds	Chain of context identifiers in split form

Returns

Localizer for the given context

Implemented in AutoLoadLocalizer, and ContextLocalizer.

6.4.2.2 Context() [2/2]

```
ILocalizer Context (
            string contextId )  [inherited]
```

Gets the localizer for a context in the current localizer.

Contexts are used to disambiguate the conversion of the same base-language string to different language-specific strings depending on the context where the conversion is performed.

Contexts can be nested. The context identifier can identify a chain of nested contexts by separating their identifiers with the '.' character (left = outermost / right = innermost).

Parameters

contextId	Identifier of the context

Returns

Localizer for the given context

Implemented in AutoLoadLocalizer, and ContextLocalizer.

6.4.2.3 LoadXML() [1/12]

```
void LoadXML (
            Assembly assembly,
            string resourceName,
            bool merge )
```

Loads translations for the current localizer language from a localization configuration file in XML format obtained from an embedded resource in the given assembly.

Parameters

assembly	Assembly that contains the embedded XML file
resourceName	Name of the embedded resource for the XML file
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	Thrown when the stream contents cannot be parsed properly.
InvalidOperationException	Thrown when the embedded resource could not be found in the given assembly.

Implemented in AutoLoadLocalizer, and Localizer.

6.4.2.4 LoadXML() [2/12]

```
void LoadXML (
            Assembly assembly,
            string resourceName,
            CultureInfo?  culture = null )
```

Loads translations for the given *culture* from a localization configuration file in XML format obtained from an embedded resource in the given assembly.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

Parameters

assembly	Assembly that contains the embedded XML file
resourceName	Name of the embedded resource for the XML file
culture	Culture for the target language of translations, or `null` to use the current UI culture (obtained from CultureInfo.CurrentUICulture)

Exceptions

ParseException	Thrown when the embedded resource contents cannot be parsed properly.
InvalidOperationException	Thrown when the embedded resource could not be found in the given assembly.

Implemented in AutoLoadLocalizer, and Localizer.

6.4.2.5 LoadXML() [3/12]

```
void LoadXML (
            Assembly assembly,
            string resourceName,
            string language )
```

Loads translations for the given *language* from a localization configuration file in XML format obtained from an embedded resource in the given assembly.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

If the system does not support a culture for *language* , then CultureInfo.InvariantCulture will be used as the culture for formatting operations.

Parameters

assembly	Assembly that contains the embedded XML file
resourceName	Name of the embedded resource for the XML file
language	Name, code or identifier for the target language of translations

Exceptions

ParseException	Thrown when the embedded resource contents cannot be parsed properly.
InvalidOperationException	Thrown when the embedded resource could not be found in the given assembly.

Implemented in AutoLoadLocalizer, and Localizer.

6.4.2.6 LoadXML() [4/12]

```
void LoadXML (
            Stream stream,
            bool merge )
```

Loads translations for the current localizer language from a localization configuration file in XML format obtained from a stream.

Parameters

stream	Stream with the localization configuration
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	Thrown when the stream contents cannot be parsed properly.

Implemented in AutoLoadLocalizer, and Localizer.

6.4.2.7 LoadXML() [5/12]

```
void LoadXML (
            Stream stream,
            CultureInfo?  culture = null )
```

Loads translations for the given *culture* from a localization configuration file in XML format obtained from a stream.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

Parameters

stream	Stream with the localization configuration
culture	Culture for the target language of translations, or `null` to use the current UI culture (obtained from CultureInfo.CurrentUICulture)

Exceptions

ParseException	Thrown when the stream contents cannot be parsed properly.

Implemented in AutoLoadLocalizer, and Localizer.

6.4.2.8 LoadXML() [6/12]

```
void LoadXML (
            Stream stream,
            string language )
```

Loads translations for the given *language* from a localization configuration file obtained in XML format from a stream.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

If the system does not support a culture for *language* , then CultureInfo.InvariantCulture will be used as the culture for formatting operations.

Parameters

stream	Stream with the localization configuration
language	Name, code or identifier for the target language of translations

Exceptions

ParseException	Thrown when the stream contents cannot be parsed properly.

Implemented in AutoLoadLocalizer, and Localizer.

6.4.2.9 LoadXML() [7/12]

```
void LoadXML (
            string filepath,
            bool merge )
```

Loads translations for the current localizer language from a localization configuration file in XML format.

Parameters

filepath	Path to the localization configuration file
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	Thrown when the input file cannot be parsed properly.

Implemented in AutoLoadLocalizer, and Localizer.

6.4.2.10 LoadXML() [8/12]

```
void LoadXML (
            string filepath,
            CultureInfo?  culture = null )
```

Loads translations for the given *culture* from a localization configuration file in XML format.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

Parameters

filepath	Path to the localization configuration file
culture	Culture for the target language of translations, or `null` to use the current UI culture (obtained from CultureInfo.CurrentUICulture)

Exceptions

ParseException	Thrown when the input file cannot be parsed properly.

Implemented in AutoLoadLocalizer, and Localizer.

6.4.2.11 LoadXML() [9/12]

```
void LoadXML (
            string filepath,
            string language )
```

Loads translations for the given *language* from a localization configuration file in XML format.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

If the system does not support a culture for *language* , then CultureInfo.InvariantCulture will be used as the culture for formatting operations.

Parameters

filepath	Path to the localization configuration file
language	Name, code or identifier for the target language of translations

Exceptions

ParseException	Thrown when the input file cannot be parsed properly.

Implemented in AutoLoadLocalizer, and Localizer.

6.4.2.12 LoadXML() [10/12]

```
void LoadXML (
            XDocument doc,
            bool merge )
```

Loads translations for the current localizer language from a localization configuration in an XML document.

Parameters

doc	XML document with the localization configuration
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	Thrown when the stream contents cannot be parsed properly.

Implemented in AutoLoadLocalizer, and Localizer.

6.4.2.13 LoadXML() [11/12]

```
void LoadXML (
            XDocument doc,
            CultureInfo?  culture = null )
```

Loads translations for the given *culture* from a localization configuration in an XML document.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

Parameters

doc	XML document with the localization configuration
culture	Culture for the target language of translations, or `null` to use the current UI culture (obtained from CultureInfo.CurrentUICulture)

Exceptions

ParseException	Thrown when the input document cannot be parsed properly.

Implemented in AutoLoadLocalizer, and Localizer.

6.4.2.14 LoadXML() [12/12]

```
void LoadXML (
            XDocument doc,
            string language )
```

Loads translations for the given *language* from a localization configuration in an XML document.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

If the system does not support a culture for *language* , then CultureInfo.InvariantCulture will be used as the culture for formatting operations.

Parameters

doc	XML document with the localization configuration
language	Name, code or identifier for the target language of translations

Exceptions

ParseException	Thrown when the input document cannot be parsed properly.

Implemented in AutoLoadLocalizer, and Localizer.

6.4.2.15 Localize() [1/3]

```
string Localize (
            FormattableString frmtText )  [inherited]
```

Localizes an interpolated string.

Converts the composite format string of the base-language formattable string *frmtText* (e.g. an interpolated string) to its corresponding language-specific localized composite format value, and then generates the result by formatting the localized composite format value along with the*frmtText* arguments by using the formatting conventions of the localizer culture.

Parameters

frmtText	Base-language formattable string

Returns

Formatted string generated from the language-specific localized format string if found, or generated from*frmt*↩ *Text* otherwise

Exceptions

FormatException	Thrown when the localized format value of *frmtText* is invalid.

Implemented in AutoLoadLocalizer, and ContextLocalizer.

6.4.2.16 Localize() [2/3]

```
IEnumerable< string > Localize (
            IEnumerable< string > texts )  [inherited]
```

Localizes multiple strings.

Converts the base-language strings in *texts* to their corresponding language-specific localized values.

Parameters

texts	Base-language strings

Returns

Implemented in AutoLoadLocalizer, and ContextLocalizer.

6.4.2.17 Localize() [3/3]

```
string Localize (
            PlainString text )  [inherited]
```

Localizes a string.

Converts the base-language string *text* to its corresponding language-specific localized value.

Parameters

text	Base-language string

Returns

Language-specific localized string if found, or *text* otherwise

Implemented in AutoLoadLocalizer, and ContextLocalizer.

6.4.2.18 LocalizeFormat()

```
string LocalizeFormat (
            string format,
            params object[] args )  [inherited]
```

Localizes and then formats a string.

Converts the base-language format string *format* to its corresponding language-specific localized format value, and then generates the result by formatting the localized format value along with the*args* arguments by using the formatting conventions of the localizer culture.

Parameters

format	Base-language format string
args	Arguments for the format string

Returns

Formatted string generated from the language-specific localized format string if found, or generated from*format* otherwise

Exceptions

FormatException	Thrown when *format* or its localized format value is invalid.

Implemented in AutoLoadLocalizer.

6.4.3 Property Documentation

6.4.3.1 TargetCulture

```
CultureInfo TargetCulture  [get], [inherited]
```

Target culture of the localizer.

Implemented in AutoLoadLocalizer, and ContextLocalizer.

6.4.3.2 TargetLanguage

```
string TargetLanguage  [get], [inherited]
```

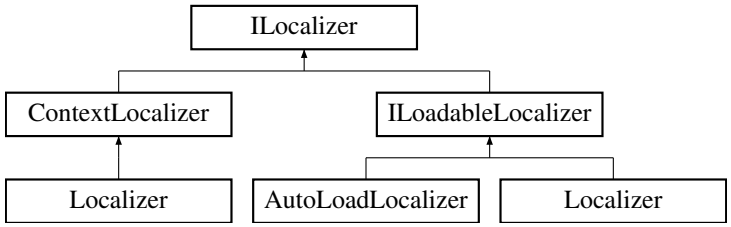
Target language of the localizer.

Implemented in AutoLoadLocalizer, and ContextLocalizer.

6.5 ILocalizer Interface Reference

Converter of strings from a base-language value to its corresponding language-specific localization.

Inheritance diagram for ILocalizer:



Public Member Functions

- string Localize (PlainString text)

  Localizes a string.
- string Localize (FormattableString frmtText)

  Localizes an interpolated string.
- string LocalizeFormat (string format, params object[] args)

  Localizes and then formats a string.
- IEnumerable< string > Localize (IEnumerable< string > texts)

  Localizes multiple strings.
- ILocalizer Context (string contextId)

  Gets the localizer for a context in the current localizer.
- ILocalizer Context (IEnumerable< string > splitContextIds)

  Gets the localizer for a context in the current localizer.

Properties

- string TargetLanguage `[get]`

  Target language of the localizer.
- CultureInfo TargetCulture `[get]`

  Target culture of the localizer.

6.5.1 Detailed Description

Converter of strings from a base-language value to its corresponding language-specific localization.

6.5.2 Member Function Documentation

6.5.2.1 Context() [1/2]

```
ILocalizer Context (
            IEnumerable< string > splitContextIds )
```

Gets the localizer for a context in the current localizer.

Contexts are used to disambiguate the conversion of the same base-language string to different language-specific strings depending on the context where the conversion is performed.

Parameters

splitContextIds	Chain of context identifiers in split form

Returns

Localizer for the given context

Implemented in AutoLoadLocalizer, and ContextLocalizer.

6.5.2.2 Context() [2/2]

```
ILocalizer Context (
            string contextId )
```

Gets the localizer for a context in the current localizer.

Contexts are used to disambiguate the conversion of the same base-language string to different language-specific strings depending on the context where the conversion is performed.

Contexts can be nested. The context identifier can identify a chain of nested contexts by separating their identifiers with the '.' character (left = outermost / right = innermost).

Parameters

contextId	Identifier of the context

Returns

Localizer for the given context

Implemented in AutoLoadLocalizer, and ContextLocalizer.

6.5.2.3 Localize() [1/3]

```
string Localize (
            FormattableString frmtText )
```

Localizes an interpolated string.

Converts the composite format string of the base-language formattable string *frmtText* (e.g. an interpolated string) to its corresponding language-specific localized composite format value, and then generates the result by formatting the localized composite format value along with the*frmtText* arguments by using the formatting conventions of the localizer culture.

Parameters

frmtText	Base-language formattable string

Returns

Formatted string generated from the language-specific localized format string if found, or generated from *frmtText* otherwise

Exceptions

FormatException	Thrown when the localized format value of *frmtText* is invalid.

Implemented in AutoLoadLocalizer, and ContextLocalizer.

6.5.2.4 Localize() [2/3]

```
IEnumerable< string > Localize (
            IEnumerable< string > texts )
```

Localizes multiple strings.

Converts the base-language strings in *texts* to their corresponding language-specific localized values.

Parameters

texts	Base-language strings

Returns

Implemented in AutoLoadLocalizer, and ContextLocalizer.

6.5.2.5 Localize() [3/3]

```
string Localize (
            PlainString text )
```

Localizes a string.

Converts the base-language string *text* to its corresponding language-specific localized value.

Parameters

text	Base-language string

Returns

Language-specific localized string if found, or *text* otherwise

Implemented in AutoLoadLocalizer, and ContextLocalizer.

6.5.2.6 LocalizeFormat()

```
string LocalizeFormat (
            string format,
            params object[] args )
```

Localizes and then formats a string.

Converts the base-language format string *format* to its corresponding language-specific localized format value, and then generates the result by formatting the localized format value along with the*args* arguments by using the formatting conventions of the localizer culture.

Parameters

format	Base-language format string
args	Arguments for the format string

Returns

Formatted string generated from the language-specific localized format string if found, or generated from*format* otherwise

Exceptions

FormatException	Thrown when *format* or its localized format value is invalid.

Implemented in AutoLoadLocalizer.

6.5.3 Property Documentation

6.5.3.1 TargetCulture

```
CultureInfo TargetCulture  [get]
```

Target culture of the localizer.

Implemented in AutoLoadLocalizer, and ContextLocalizer.

6.5.3.2 TargetLanguage

```
string TargetLanguage  [get]
```

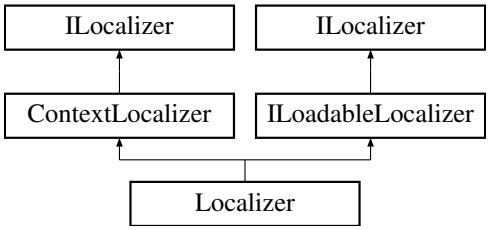
Target language of the localizer.

Implemented in AutoLoadLocalizer, and ContextLocalizer.

6.6 Localizer Class Reference

Simple loadable localizer.

Inheritance diagram for Localizer:



Public Member Functions

- Localizer ()

  Default constructor.
- void LoadXML (string filepath, CultureInfo? culture=null)

  Loads translations for the given culture from a localization configuration file in XML format.
  All the translations loaded previously in the localizer are discarded and replaced with the new ones.
- void LoadXML (string filepath, string language)

  Loads translations for the given language from a localization configuration file in XML format.
  All the translations loaded previously in the localizer are discarded and replaced with the new ones.
- void LoadXML (string filepath, bool merge)

  Loads translations for the current localizer language from a localization configuration file in XML format.

Parameters

filepath	*Path to the localization configuration file*
merge	*Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).*

Exceptions

ParseException	*Thrown when the input file cannot be parsed properly.*

- void LoadXML (Stream stream, CultureInfo? culture=null)

  Loads translations for the given culture from a localization configuration file in XML format obtained from a stream.
  All the translations loaded previously in the localizer are discarded and replaced with the new ones.
- void LoadXML (Stream stream, string language)

Loads translations for the given language from a localization configuration file obtained in XML format from a stream.
All the translations loaded previously in the localizer are discarded and replaced with the new ones.

- void LoadXML (Stream stream, bool merge)

  Loads translations for the current localizer language from a localization configuration file in XML format obtained from a stream.

Parameters

stream	*Stream with the localization configuration*
merge	*Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).*

Exceptions

ParseException	*Thrown when the stream contents cannot be parsed properly.*

- void LoadXML (XDocument doc, CultureInfo? culture=null)

  Loads translations for the given culture from a localization configuration in an XML document.
  All the translations loaded previously in the localizer are discarded and replaced with the new ones.
- void LoadXML (XDocument doc, string language)

  Loads translations for the given language from a localization configuration in an XML document.
  All the translations loaded previously in the localizer are discarded and replaced with the new ones.
- void LoadXML (XDocument doc, bool merge)

  Loads translations for the current localizer language from a localization configuration in an XML document.

Parameters

doc	*XML document with the localization configuration*
merge	*Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).*

Exceptions

ParseException	*Thrown when the stream contents cannot be parsed properly.*

- void LoadXML (Assembly assembly, string resourceName, CultureInfo? culture=null)

  Loads translations for the given culture from a localization configuration file in XML format obtained from an embedded resource in the given assembly.
  All the translations loaded previously in the localizer are discarded and replaced with the new ones.
- void LoadXML (Assembly assembly, string resourceName, string language)

  Loads translations for the given language from a localization configuration file in XML format obtained from an embedded resource in the given assembly.
  All the translations loaded previously in the localizer are discarded and replaced with the new ones.
- void LoadXML (Assembly assembly, string resourceName, bool merge)

  Loads translations for the current localizer language from a localization configuration file in XML format obtained from an embedded resource in the given assembly.

Parameters

assembly	*Assembly that contains the embedded XML file*
resourceName	*Name of the embedded resource for the XML file*
merge	*Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).*

Exceptions

ParseException	*Thrown when the stream contents cannot be parsed properly.*
InvalidOperationException	*Thrown when the embedded resource could not be found in the given assembly.*

- string Localize (PlainString text)

  Localizes a string.
  Converts the base-language string text to its corresponding language-specific localized value.
- string Localize (FormattableString frmtText)

  Localizes an interpolated string.
  Converts the composite format string of the base-language formattable string frmtText (e.g. an interpolated string) to its corresponding language-specific localized composite format value, and then generates the result by formatting the localized composite format value along with thefrmtText arguments by using the formatting conventions of the localizer culture.
- IEnumerable< string > Localize (IEnumerable< string > texts)

  Localizes multiple strings.
  Converts the base-language strings in texts to their corresponding language-specific localized values.
- string LocalizeFormat (string format, params object?[ ] args)
- string LocalizeFormat (string format, params object[ ] args)

  Localizes and then formats a string.
- ContextLocalizer Context (string contextId)

  Gets the localizer for a context in the current localizer.
  Contexts are used to disambiguate the conversion of the same base-language string to different language-specific strings depending on the context where the conversion is performed.
- ContextLocalizer Context (IEnumerable< string > splitContextIds)

  Gets the localizer for a context in the current localizer.
  Contexts are used to disambiguate the conversion of the same base-language string to different language-specific strings depending on the context where the conversion is performed.

Protected Member Functions

- void Clear ()
- void Load (XElement element)

Properties

- string TargetLanguage `[get]`

  Target language of the localizer.
- CultureInfo TargetCulture `[get]`

  Target culture of the localizer.
- Language Language `[get, set]`

6.6.1 Detailed Description

Simple loadable localizer.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 Localizer()

```
Localizer ( )
```

Default constructor.

6.6.3 Member Function Documentation

6.6.3.1 Clear()

```
void Clear ( )  [protected], [inherited]
```

6.6.3.2 Context() [1/2]

```
ContextLocalizer Context (
            IEnumerable< string > splitContextIds )  [inherited]
```

Gets the localizer for a context in the current localizer.

Contexts are used to disambiguate the conversion of the same base-language string to different language-specific strings depending on the context where the conversion is performed.

Implements ILocalizer.

6.6.3.3 Context() [2/2]

```
ContextLocalizer Context (
            string contextId )  [inherited]
```

Gets the localizer for a context in the current localizer.

Contexts are used to disambiguate the conversion of the same base-language string to different language-specific strings depending on the context where the conversion is performed.

Implements ILocalizer.

6.6.3.4 Load()

```
void Load (
            XElement element )  [protected], [inherited]
```

6.6.3.5 LoadXML() [1/12]

```
void LoadXML (
            Assembly assembly,
            string resourceName,
            bool merge )
```

Loads translations for the current localizer language from a localization configuration file in XML format obtained from an embedded resource in the given assembly.

Parameters

assembly	Assembly that contains the embedded XML file
resourceName	Name of the embedded resource for the XML file
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	Thrown when the stream contents cannot be parsed properly.
InvalidOperationException	Thrown when the embedded resource could not be found in the given assembly.

Implements ILoadableLocalizer.

6.6.3.6 LoadXML() [2/12]

```
void LoadXML (
            Assembly assembly,
            string resourceName,
            CultureInfo?  culture = null )
```

Loads translations for the given *culture* from a localization configuration file in XML format obtained from an embedded resource in the given assembly.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

Implements ILoadableLocalizer.

6.6.3.7 LoadXML() [3/12]

```
void LoadXML (
            Assembly assembly,
            string resourceName,
            string language )
```

Loads translations for the given *language* from a localization configuration file in XML format obtained from an embedded resource in the given assembly.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

Implements ILoadableLocalizer.

6.6.3.8 LoadXML() [4/12]

```
void LoadXML (
            Stream stream,
            bool merge )
```

Loads translations for the current localizer language from a localization configuration file in XML format obtained from a stream.

Parameters

stream	Stream with the localization configuration
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	Thrown when the stream contents cannot be parsed properly.

Implements ILoadableLocalizer.

6.6.3.9 LoadXML() [5/12]

```
void LoadXML (
            Stream stream,
            CultureInfo?  culture = null )
```

Loads translations for the given *culture* from a localization configuration file in XML format obtained from a stream.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

Implements ILoadableLocalizer.

6.6.3.10 LoadXML() [6/12]

```
void LoadXML (
            Stream stream,
            string language )
```

Loads translations for the given *language* from a localization configuration file obtained in XML format from a stream.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

Implements ILoadableLocalizer.

6.6.3.11 LoadXML() [7/12]

```
void LoadXML (
            string filepath,
            bool merge )
```

Loads translations for the current localizer language from a localization configuration file in XML format.

Parameters

filepath	Path to the localization configuration file
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	Thrown when the input file cannot be parsed properly.

Implements ILoadableLocalizer.

6.6.3.12 LoadXML() [8/12]

```
void LoadXML (
            string filepath,
            CultureInfo?  culture = null )
```

Loads translations for the given *culture* from a localization configuration file in XML format.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

Implements ILoadableLocalizer.

6.6.3.13 LoadXML() [9/12]

```
void LoadXML (
            string filepath,
            string language )
```

Loads translations for the given *language* from a localization configuration file in XML format.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

Implements ILoadableLocalizer.

6.6.3.14 LoadXML() [10/12]

```
void LoadXML (
            XDocument doc,
            bool merge )
```

Loads translations for the current localizer language from a localization configuration in an XML document.

Parameters

doc	XML document with the localization configuration
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	Thrown when the stream contents cannot be parsed properly.

Implements ILoadableLocalizer.

6.6.3.15 LoadXML() [11/12]

```
void LoadXML (
            XDocument doc,
            CultureInfo?  culture = null )
```

Loads translations for the given *culture* from a localization configuration in an XML document.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

Implements ILoadableLocalizer.

6.6.3.16 LoadXML() [12/12]

```
void LoadXML (
            XDocument doc,
            string language )
```

Loads translations for the given *language* from a localization configuration in an XML document.

All the translations loaded previously in the localizer are discarded and replaced with the new ones.

Implements ILoadableLocalizer.

6.6.3.17 Localize() [1/3]

```
string Localize (
            FormattableString frmtText )  [inherited]
```

Localizes an interpolated string.

Converts the composite format string of the base-language formattable string *frmtText* (e.g. an interpolated string) to its corresponding language-specific localized composite format value, and then generates the result by formatting the localized composite format value along with the*frmtText* arguments by using the formatting conventions of the localizer culture.

Implements ILocalizer.

6.6.3.18 Localize() [2/3]

```
IEnumerable< string > Localize (
            IEnumerable< string > texts )  [inherited]
```

Localizes multiple strings.

Converts the base-language strings in *texts* to their corresponding language-specific localized values.

Implements ILocalizer.

6.6.3.19 Localize() [3/3]

```
string Localize (
            PlainString text )  [inherited]
```

Localizes a string.

Converts the base-language string *text* to its corresponding language-specific localized value.

Implements ILocalizer.

6.6.3.20 LocalizeFormat() [1/2]

```
string LocalizeFormat (
            string format,
            params object?[] args )  [inherited]
```

6.6.3.21 LocalizeFormat() [2/2]

```
string LocalizeFormat (
            string format,
            params object[] args )  [inherited]
```

Localizes and then formats a string.

Converts the base-language format string *format* to its corresponding language-specific localized format value, and then generates the result by formatting the localized format value along with the*args* arguments by using the formatting conventions of the localizer culture.

Parameters

format	Base-language format string
args	Arguments for the format string

Returns

Formatted string generated from the language-specific localized format string if found, or generated from *format* otherwise

Exceptions

FormatException	Thrown when *format* or its localized format value is invalid.

Implemented in AutoLoadLocalizer.

6.6.4 Property Documentation

6.6.4.1 Language

```
Language Language  [get], [set], [protected], [inherited]
```

6.6.4.2 TargetCulture

```
CultureInfo TargetCulture  [get], [inherited]
```

Target culture of the localizer.

Implements ILocalizer.

6.6.4.3 TargetLanguage

```
string TargetLanguage  [get], [inherited]
```

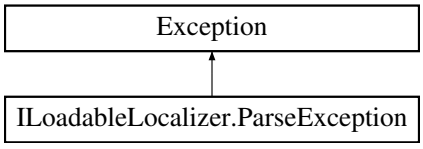
Target language of the localizer.

Implements ILocalizer.

6.7 ILoadableLocalizer.ParseException Class Reference

Exception thrown when a localization file cannot be parsed properly.

Inheritance diagram for ILoadableLocalizer.ParseException:



Public Member Functions

- ParseException (string message)

  Constructor.

6.7.1 Detailed Description

Exception thrown when a localization file cannot be parsed properly.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 ParseException()

```
ParseException (
            string message )
```

Constructor.

Parameters

message	A message that describes the error.

6.8 PlainString Class Reference

Represents just a string. This class is used to allow interpolated strings to preferably be passed as Formattable↩String instead of string to methods that overload both types.

Public Member Functions

- PlainString (string value)

  Default constructor.

Static Public Member Functions

- static implicit operator PlainString (string value)

  Converts a string value to a PlainString.
- static implicit operator PlainString (FormattableString arg)

  Converts a FormattableString value to a PlainString.

Properties

- string Value `[get]`

6.8.1 Detailed Description

Represents just a string. This class is used to allow interpolated strings to preferably be passed as FormattableString instead of string to methods that overload both types.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 PlainString()

```
PlainString (
            string value )
```

Default constructor.

6.8.3 Member Function Documentation

6.8.3.1 operator PlainString() [1/2]

```
static implicit operator PlainString (
            FormattableString arg )  [static]
```

Converts a FormattableString value to a PlainString.

This implicit operator is needed to avoid FormattableString values to be automatically converted to string and then to PlainString when resolving parameter overloads.

Value

Exceptions

InvalidOperationException	Always thrown

6.8.3.2 operator PlainString() [2/2]

```
static implicit operator PlainString (
            string value )  [static]
```

Converts a string value to a PlainString.

Parameters

value	Value

6.8.4 Property Documentation

6.8.4.1 Value

```
string Value  [get]
```

Value of the string.

Index