

# Localisation des vos applications

---

## Sommaire

Localisation des vos applications .....	1
1 Introduction.....	2
2 Gestion de la globalisation .....	3
2.1 Récupérer des informations.....	4
2.2 Outil de comparaison .....	13
3 Créer ses paramètres de zone.....	14
4 Conclusion .....	19

Dotnet-France Association

## 1 Introduction

Jusqu'ici, nos applications ne prenaient pas en compte les paramètres relatifs à la partie du monde dans laquelle elle s'exécutait. Si nous avons omis ce détail, nous aurions pu avoir des comportements indésirables en ce qui concerne les dates, le calcul de sommes ou encore l'affichage de caractères spéciaux ...



Dans ce chapitre, nous allons voir comment le .NET Framework permet de gérer ces paramètres intercontinentaux grâce à la globalisation.

Dans un premier temps, nous verrons comment récupérer ces informations puis nous verrons comment créer des informations personnalisées.

Tous les outils nécessaires à la globalisation se trouvent dans l'espace de nom `System.Globalization`.

## 2 Gestion de la globalisation

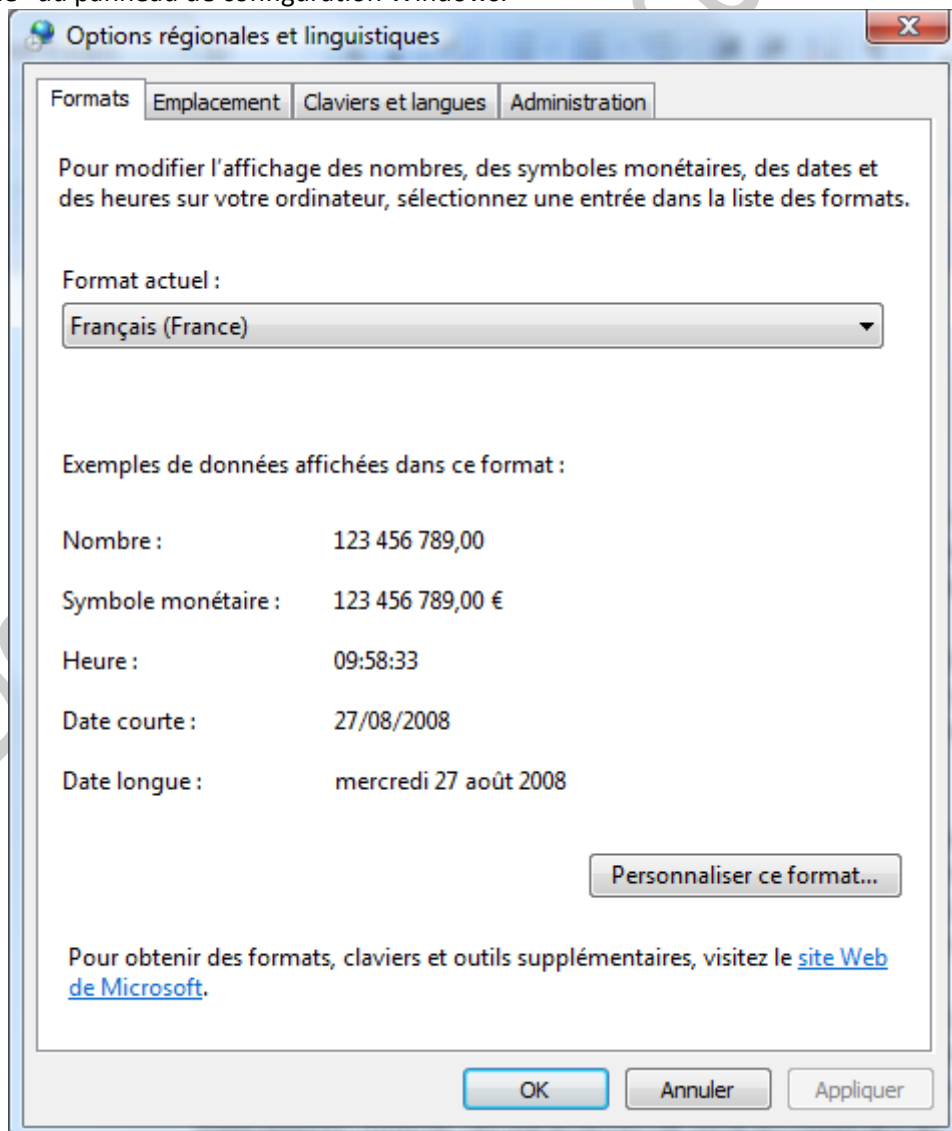
Dans le chapitre 3, nous avons vu que les tables de codage des caractères permettaient d'assurer une interopérabilité des textes dans les applications. Cela permettant entre autre à des personnes de différents pays de lire correctement les textes (le meilleur exemple étant surement celui du codage utilisé par les pages des sites web).

Dans ce chapitre nous allons pousser l'interopérabilité encore plus loin grâce à la Globalization. Ce procédé permet non seulement de conserver la lisibilité des textes mais également d'adapter l'application aux différents formats de calendriers, aux systèmes monétaires des différents pays ou à la notation des chiffres etc.

Il faut savoir que les informations de zones peuvent être découpées en trois catégories :

- Les informations invariantes (Invariant Culture) qui permettent de ne pas prendre en compte les spécificités des langues dans les formatages.
- Les informations neutres (Neutral Culture) qui représentent les langues les plus utilisées dans le monde (Anglais, Espagnol, Français).
- Les informations spécifiques (Specific Culture) qui sont propre à chaque pays du monde (Par exemple, les cultures dont les noms affichés sont "Français (France)" et "Français (Luxembourg)" sont deux informations de zones différentes)

Vous pouvez modifier ces paramètres de zone en modifiant les "Options régionales et linguistique" du panneau de configuration Windows.



Toutes les classes nécessaires à la gestion de la Globalization se trouvent dans `System.Globalization`. Cependant, les paramètres de zones sont propres aux domaines d'application et/ou aux Threads qu'ils contiennent. Aussi, pour pouvoir modifier ou récupérer les paramètres de zone, vous devrez passer par la propriété `Thread.CurrentThread.CurrentCulture`.

## 2.1 Récupérer des informations

La classe `CultureInfo` joue un rôle clé dans la gestion de la globalization car c'est elle qui va contenir tous les objets permettant de modifier ou de récupérer les paramètres de zone. En voici les principaux membres :

### ➤ Membres statique

Membres	Description
<code>CreateSpecificCulture</code>	Retourne une instance de la classe <code>CultureInfo</code> dont le nom est passé en paramètre.
<code>GetCultureInfo</code>	Récupère un objet <code>CultureInfo</code> chargé en mémoire.
<code>GetCultures</code>	Retourne un tableau d'objet <code>CultureInfo</code> supportés par le système. Le paramètre passé est une valeur de l'énumération <code>CultureTypes</code> détaillée plus bas.
<code>CurrentCulture</code>	Retourne l'objet <code>CultureInfo</code> utilisé par le Thread courant. Elle peut être modifiée à tout moment de l'application.
<code>CurrentUICulture</code>	Retourne l'objet <code>CultureInfo</code> utilisé par le gestionnaire de ressources. Il est essentiellement utilisé pour l'affichage plus que pour que le calcul. Elle ne peut être modifiée correctement qu'au démarrage de l'application (avant l'affichage de fenêtre par exemple)
<code>InstalledUICulture</code>	Retourne un objet <code>CultureInfo</code> contenant les informations de zone sélectionnées lors de l'installation de l'OS.
<code>InvariantCulture</code>	Retourne l'objet <code>CultureInfo</code> invariant.

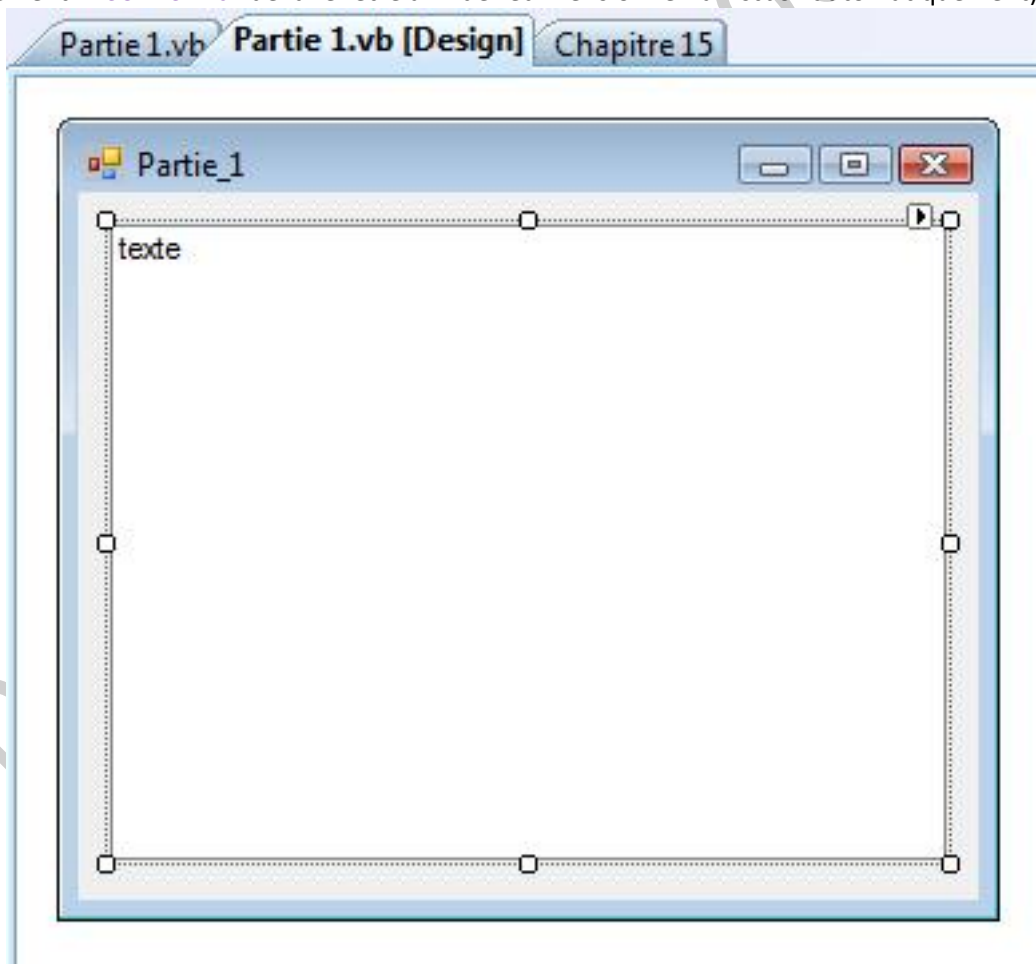
### ➤ Membres d'instance

Membres	Description
<code>GetFormat</code>	Retourne un objet définissant le format utilisé pour le type passé en paramètre.
<code>Calendar</code>	Contient l'objet <code>Calendar</code> utilisé pour la zone actuelle.
<code>CompareInfo</code>	Contient l'objet <code>CompareInfo</code> permettant d'obtenir des informations sur la procédure de comparaison de chaîne de caractères utilisée dans la zone actuelle.
<code>DateTimeFormat</code>	Contient l'objet <code>DateTimeFormatInfo</code> indiquant le formatage utilisé pour les dates.
<code>Name</code>	Obtient le nom de la culture sous forme abrégée.
<code>DisplayName</code>	Obtient le nom de la culture utilisée. Le nom sera retourné, formaté en utilisant les paramètres de formatage de texte de la zone.
<code>EnglishName</code>	Obtient le nom de la culture utilisée, écrit en anglais.
<code>LCID</code>	Obtient un identificateur de culture.
<code>NumberFormat</code>	Obtient ou définit un objet <code>NumberFormatInfo</code> qui indique comment sont formatés les nombres.
<code>OptionalCalendar</code>	Retourne une liste d'objets <code>Calendar</code> qui peuvent être utilisés dans la zone actuelle.
<code>IsNeutralCulture</code>	Indique si la culture en cours est de type neutre ou non.

## ➤ Eléments de CultureTypes

Valeur	Description
<code>NeutralCultures</code>	Retourne toutes les informations de zone de type neutre.
<code>SpecificCultures</code>	Retourne toutes les informations de zone de type spécifique.
<code>InstalledWin32Cultures</code>	Retourne toutes les informations de zone installées sur le système d'exploitation. Sachez que certaines informations prise en charge par le Framework ne sont pas installées sur l'OS.
<code>AllCultures</code>	Retourne toutes les informations de zone disponible.
<code>UserCustomCulture</code>	Retourne uniquement les informations personnalisées.
<code>ReplacementCultures</code>	Retourne les informations personnalisées qui remplacent celle du .NET Framework.
<code>WindowsOnlyCultures</code>	Retourne uniquement les informations installées sur l'OS qui ne sont pas contenue dans le Framework.
<code>FrameworkCultures</code>	Retourne les informations de zone fournies avec le .NET Framework.

Dans l'exemple suivant, nous avons créé un projet Winform dans lequel nous avons ajouté une Listbox sans configuration particulière (Optionnellement, vous pouvez ajouter un écouteur sur l'évènement "`ResizeEnd`" de la fenêtre afin de redimensionner la Listbox automatiquement) :



Côté code, nous effectuons un listing de toutes les cultures spécifiques disponibles. Nous en affichons leur nom ainsi qu'une valeur monétaire sur chaque ligne de la Listbox :

```
'VB
Imports System.Globalization
Imports System.Threading

Public Class partie1
    Public Sub New()
        InitializeComponent()

        For Each culture As CultureInfo In
CultureInfo.GetCultures(CultureTypes.SpecificCultures)
            Thread.CurrentThread.CurrentCulture = culture
            texte.Items.Add("Nom: " + culture.DisplayName + " : " +
Format("10000", "Currency"))
        Next
    End Sub
    'Optionnel. Uniquement pour l'auto-redimensionnement de la textbox
    Private Sub partie1_ResizeEnd(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.ResizeEnd
        texte.Size = New System.Drawing.Size(MyBase.Width - 50,
MyBase.Height - 50)
    End Sub
End Class
```

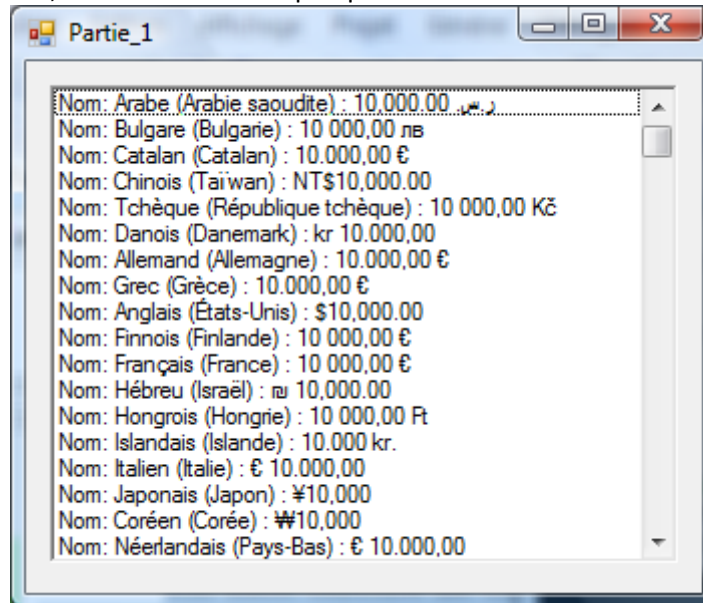
```
//C#
using System.Globalization;
using System.Threading;

public partial class Form1 : Form
{
    public Form1 ()
    {
        InitializeComponent();

        foreach(CultureInfo culture in
CultureInfo.GetCultures(CultureTypes.SpecificCultures))
        {
            Thread.CurrentThread.CurrentCulture = culture;
            texte.Items.Add("Nom: " + culture.DisplayName + " : " +
(10000).ToString("C"));
        }
    }

    private void Form1_ResizeEnd(object sender, EventArgs e)
    {
        texte.Size = new System.Drawing.Size(base.Width - 50, base.Height
- 50);
    }
}
```

Après compilation, vous devriez avoir quelque chose de similaire à ceci :



Chacune des cultures disponibles est affichée avec le format monétaire utilisé.

**Note :** Les méthodes `Format` en VB.NET et `ToString(param)` en C# se basent sur les informations de zone pour formater des objets en respectant les contraintes imposées par la culture utilisée dans le Thread actuel. Vous pouvez avoir une liste des [formateurs disponible](#) ou plus d'informations sur le formatage sur [le site du MSDN](#).

En relation avec la classe `CultureInfo`, vous pouvez obtenir des informations plus précises (au niveau des pays et/ou des régions) en utilisant la classe `RegionInfo`. Cette classe possède, à peu de choses près, les mêmes membres que la classe `CultureInfo`.

Comme exemple, nous allons créer une nouvelle classe que nous appellerons `Data`. Elle contiendra un objet `CultureInfo` et un objet `RegionInfo` ainsi qu'une surcharge de la méthode `ToString` pour l'affichage dans la `ListBox` :

```
'VB
Imports System.Globalization
Imports System.Threading

Public Class Data
    Private _culture As CultureInfo
    Private _region As RegionInfo

    Public ReadOnly Property Region() As RegionInfo
        Get
            Return _region
        End Get
    End Property

    Public ReadOnly Property Culture() As CultureInfo
        Get
            Return _culture
        End Get
    End Property

    Public Sub New(ByVal c As CultureInfo, ByVal r As RegionInfo)
        _culture = c
        _region = r
    End Sub

    Public Overrides Function ToString() As String
        Thread.CurrentThread.CurrentCulture = _culture
        Return "Nom: " + _culture.DisplayName + " : " + Format("10000",
"Currency")
    End Function
End Class
```

```
//C#
using System.Globalization;
using System.Threading;

public class Data
{
    private CultureInfo _culture;
    private RegionInfo _region;

    public RegionInfo Region
    {
        get { return _region; }
    }

    public CultureInfo Culture
    {
        get { return _culture; }
    }

    public Data(CultureInfo c, RegionInfo r)
    {
        _culture = c;
        _region = r;
    }

    public override string ToString()
    {
        Thread.CurrentThread.CurrentCulture = _culture
        return "Nom: " + _culture.DisplayName + " : " +
(10000).ToString("C");
    }
}
```

Dans la classe principale, nous allons ajouter non plus les textes directement dans la Listbox mais des instances de notre classe `Data`. Sur la Listbox, nous ajouterons également un écouteur de l'évènement `"SelectedIndexChanged"` dans lequel nous afficherons un Message informant de quelques informations de la région en relation avec la culture sur laquelle on vient de cliquer. Dans ce message, nous afficherons également le format de date et de format de nombres grâce aux classes `DateTimeFormatInfo` et `NumberFormatInfo` :

```
'VB
Imports System.Globalization

Public Class partiel
    Public Sub New()
        InitializeComponent()
        For Each culture As CultureInfo In
CultureInfo.GetCultures(CultureTypes.SpecificCultures)
            texte.Items.Add(New Data(culture, New
RegionInfo(culture.LCID)))
        Next
    End Sub
    Private Sub texte_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles texte.SelectedIndexChanged
        Dim d As Data = CType(texte.SelectedItem, Data)
        System.Windows.Forms.MessageBox.Show("Symbole monétaire: " +
d.Region.CurrencySymbol + vbNewLine + _
"ID de Géolocalisation: " +
d.Region.GeoId.ToString() + vbNewLine + _
"Format de date: " +
d.Culture.DateTimeFormat.RFC1123Pattern + vbNewLine + _
"Premier jour: " +
d.Culture.DateTimeFormat.DayNames() (0) + vbNewLine + _
"Séparateur décimal: " +
d.Culture.NumberFormat.NumberDecimalSeparator + vbNewLine + _
"Signe de l'infini négatif:
" + d.Culture.NumberFormat.NegativeInfinitySymbol, "Nom : " +
d.Region.DisplayName, Windows.Forms.MessageBoxButtons.OK,
Windows.Forms.MessageBoxIcon.Information,
Windows.Forms.MessageBoxDefaultButton.Button1)
    End Sub
    'Optionnel. Uniquement pour l'auto-redimensionnement de la textbox
    Private Sub partiel_ResizeEnd(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.ResizeEnd
        texte.Size = New System.Drawing.Size(MyBase.Width - 50,
MyBase.Height - 50)
    End Sub
End Class
```

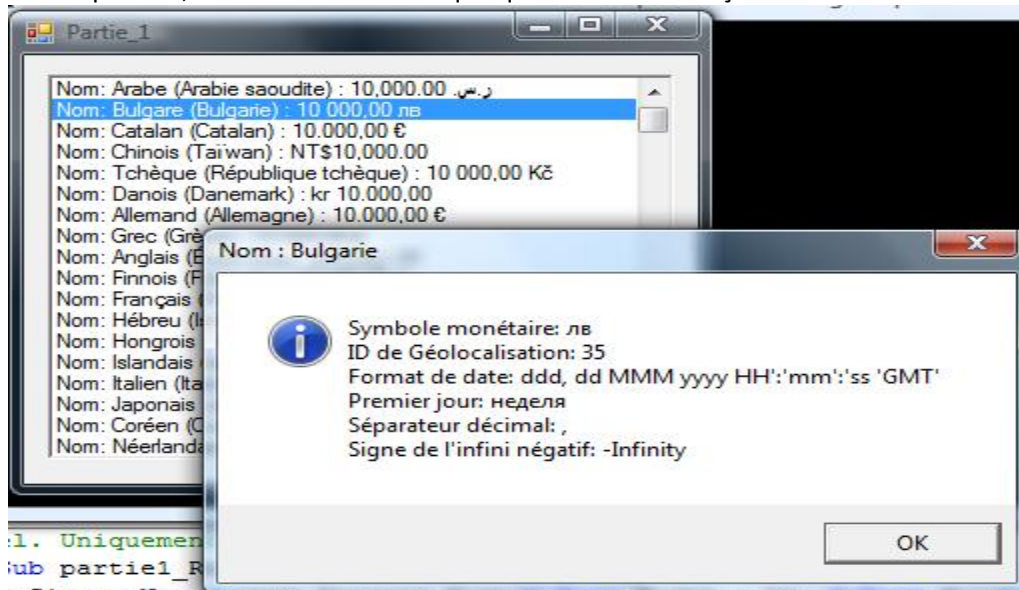
```
//C#
using System.Globalization;

public partial class Form1 : Form
{
    public Form1 ()
    {
        InitializeComponent ();

        foreach (CultureInfo culture in
CultureInfo.GetCultures (CultureTypes.SpecificCultures))
        {
            texte.Items.Add (new Data (culture, new
RegionInfo (culture.Name));
        }
        private void texte_SelectedIndexChanged (object sender, EventArgs e)
        {
            Data d = (Data)texte.SelectedItem;
            System.Windows.Forms.MessageBox.Show ("Symbole monétaire: " +
d.Region.CurrencySymbol + "\n" +
                    "ID de Géolocalisation: " +
d.Region.GeoId.ToString () + "\n" +
                    "Format de date: " +
d.Culture.DateTimeFormat.RFC1123Pattern + "\n" +
                    "Premier jour: " +
d.Culture.DateTimeFormat.DayNames [0] + "\n" +
                    "Séparateur décimal: " +
d.Culture.NumberFormat.NumberDecimalSeparator + "\n" +
                    "Signe de l'infini négatif: " +
d.Culture.NumberFormat.NegativeInfinitySymbol, "Nom : " +
d.Region.DisplayName, MessageBoxButtons.OK, MessageBoxIcon.Information,
MessageBoxDefaultButton.Button1);
        }
        private void Form1_ResizeEnd (object sender, EventArgs e)
        {
            texte.Size = new System.Drawing.Size (base.Width - 50, base.Height
- 50);
        }
    }
}
```

Dotnet

A la compilation, vous devriez obtenir quelque chose comme ça :



La fenêtre affichant les informations apparaîtra dès que l'on sélectionnera une autre culture affichée dans la Listbox.

## 2.2 Outil de comparaison

La classe `CultureInfo` contient également une propriété retournant un objet `CompareInfo`. Cet objet va vous permettre de comparer deux chaînes de caractères en prenant en compte les restrictions liées à la langue utilisée par la culture désignée. Dans cette méthode de comparaison, vous pourrez spécifier l'une des valeurs de l'énumération `CompareOptions` afin d'en modifier le comportement :

Valeur	Description
<code>IgnoreCase</code>	Rend la comparaison insensible à la casse.
<code>IgnoreKanaType</code>	Indique que le type d'écriture Kana doit être ignoré.
<code>IgnoreNonSpace</code>	Indique que la comparaison ignore les caractères sans espaces (comme les accents sur le "e" en français).
<code>IgnoreSymbols</code>	Indique que la comparaison ignore les caractères non alphanumériques.
<code>IgnoreWidth</code>	Indique que la comparaison ignore la taille des caractères (utile dans le cas des langues comme le japonais qui peuvent représenter leurs symboles en entier ou à moitié)
<code>None</code>	Aucune option de filtrage (par défaut).
<code>Ordinal</code>	Force le comparateur à utiliser les valeurs Unicode pour effectuer la comparaison.
<code>OrdinalIgnoreCase</code>	Combinaison de <code>Ordinal</code> et <code>IgnoreCase</code> .
<code>StringSort</code>	Indique que les caractères seront triés avant d'effectuer la comparaison.

Par exemple, nous pourrions comparer les chaînes "Hotels" et "Hôtels" pour savoir si elles sont identiques sans tenir compte des accents :

```
'VB
Imports System.Globalization

Public Class partiel
    Public Sub New()
        Dim c As CultureInfo = CultureInfo.GetCultureInfo("fr-FR")
        Console.WriteLine(c.CompareInfo.Compare("Hotels", "Hôtels",
        CompareOptions.IgnoreCase Or CompareOptions.IgnoreNonSpace))
    End Sub
End Class
```

```
//C#
using System.Globalization;

static void Main(string[] args)
{
    CultureInfo c = CultureInfo.GetCultureInfo("fr-FR");
    Console.WriteLine(c.CompareInfo.Compare("Hotels", "Hôtels",
    CompareOptions.IgnoreCase | CompareOptions.IgnoreNonSpace));
}
```

Ce qui donnera :

0

Le zéro correspondant à l'égalité des chaînes.

### 3 Créer ses paramètres de zone

Nous avons vu dans la partie précédente comment utiliser les formats de culture dans vos applications. Nous allons maintenant voir brièvement comment créer vos propres formats personnalisés.

Le but premier de la création de formats personnalisés va être de pouvoir les réutiliser dans chacune de vos applications et de les installer sur les machines clientes plutôt que de devoir reconfigurer à chaque fois les formats culturels.

Pour créer vos formats de culture, vous aurez besoin d'ajouter une référence à l'assembly `sysglobl.dll` et d'utiliser l'espace de nom `System.Globalization`.

Pour créer vos formats de culture, vous aurez besoin de la classe `CultureAndRegionInfoBuilder`. Le constructeur de cette classe prend en premier paramètre une chaîne de caractère correspondant à un paramètre de culture existant ou non, et en second paramètre elle prend une valeur de l'énumération `CultureAndRegionModifiers`.

Vous pouvez créer des formats de culture de trois sortes. Soit vous partez de rien et remplacez complètement une culture déjà existante en utilisant la valeur d'énumération `Replacement`, soit vous créez un nouveau paramètre spécifique grâce à `None` ou bien vous pouvez créer un paramètre de culture neutre avec `Neutral`.

Nous allons étudier deux cas de figure : tout d'abord nous allons utiliser la valeur d'énumération `Neutral` créer une nouvelle culture neutre que nous enregistrerons sur le disque de la machine et, ensuite, nous utiliserons la valeur `None` pour créer un nouveau paramètre de culture spécifique.

**Note** : Pour enregistrer un paramètre de culture sur le disque d'une machine il faut impérativement les droits d'administrateurs. Aussi, il peut être judicieux de créer une petite application accompagnant la principale qui demandera les droits d'administrateurs pour pouvoir installer la culture personnalisée.

Avant de détailler les exemples, je vous invite à vous renseigner sur les membres de la classe `CultureAndRegionInfoBuilder` sur [MSDN](#).

Nous allons donc commencer par créer un nouveau format de culture neutre représentant la très exotique culture Pokemon. Celle-ci va servir de base à une culture spécifique que nous créerons plus loin. Voilà le code permettant de créer notre culture Pokemon :

```
'VB
Sub Main()
    Dim pokemon As CultureAndRegionInfoBuilder = New
CultureAndRegionInfoBuilder("Pokemon-FR",
CultureAndRegionModifiers.Neutral)

    pokemon.Parent = CultureInfo.InvariantCulture
    pokemon.CultureEnglishName = "Pokemon"
    pokemon.CultureNativeName = "PokeMon TM"
    pokemon.ThreeLetterISOLanguageName = "pkm"
    pokemon.ThreeLetterWindowsLanguageName = "pkm"
    pokemon.TwoLetterISOLanguageName = "pk"
    pokemon.IetfLanguageTag = "pk-FR"
    pokemon.TextInfo = CultureInfo.InvariantCulture.TextInfo
    pokemon.CompareInfo = CultureInfo.InvariantCulture.CompareInfo
    pokemon.KeyboardLayoutId = 1081

    pokemon.Register()

    Console.WriteLine(CultureInfo.GetCultureInfo("Pokemon-FR"))
    Console.Read()
End Sub
```

```
//C#
static void Main(string[] args)
{
    CultureAndRegionInfoBuilder pokemon = new
CultureAndRegionInfoBuilder("Pokemon-FR",
CultureAndRegionModifiers.Neutral);

    pokemon.Parent = CultureInfo.InvariantCulture;
    pokemon.CultureEnglishName = "Pokemon";
    pokemon.CultureNativeName = "PokeMon TM";
    pokemon.ThreeLetterISOLanguageName = "pkm";
    pokemon.ThreeLetterWindowsLanguageName = "pkm";
    pokemon.TwoLetterISOLanguageName = "pk";
    pokemon.IetfLanguageTag = "pk-FR";
    pokemon.TextInfo = CultureInfo.InvariantCulture.TextInfo;
    pokemon.CompareInfo = CultureInfo.InvariantCulture.CompareInfo;
    pokemon.KeyboardLayoutId = 1081;

    pokemon.Register();

    Console.WriteLine(CultureInfo.GetCultureInfo("Pokemon-FR"));
    Console.Read();
}
```

Nous commençons par instancier un objet `CultureAndRegionInfoBuilder` qui fera référence au format de culture neutre Pokemon-FR.

Ensuite nous assignons plusieurs valeurs à notre format de culture, sans trop détailler (reportez vous à la documentation sur [MSDN](#)), sachez que toutes les propriétés que j'ai utilisées sont obligatoires pour créer un format de culture neutre.

Nous assignons à `Parent` `CultureInfo.InvariantCulture` qui indique que le format de culture parent est Invariant.

De même pour `TextInfo` et `CompareInfo`, nous prenons les propriétés de `InvariantCulture`, les comparaisons et le formatage de texte seront donc sans aucune spécificité.

Les propriétés possédant le mot Name sont celles qui permettent de définir les divers noms du format sous plusieurs formes, en langue anglaise, dans la langue du pays, et sous divers normes, notamment ISO. Il en est de même pour letFlanguageTag qui permet d'indiquer le nom du format respectant la norme RFC 4646.

Enfin [KeyboardLayoutId](#) est une propriété un peu particulière permettant de définir ou obtenir un identifiant représentant les paramètres régionaux d'un périphérique d'entrée (notamment le clavier, mais aussi un dispositif de reconnaissance vocal par exemple).

Enfin nous vérifions que notre format de culture a bien été enregistré en affichant le retour de la méthode statique [GetCultureInfo](#) :

```
Pokemon-FR
```

Maintenant que nous avons notre format de culture neutre, nous allons pouvoir créer notre format de culture spécifique :

```
//C#
Sub Main()
    Dim pikatchu As CultureAndRegionInfoBuilder = New
CultureAndRegionInfoBuilder("Pikatchu-Pokemon",
CultureAndRegionModifiers.None)
    pikatchu.Parent = New CultureInfo("Pokemon-FR")
    pikatchu.LoadDataFromCultureInfo(pikatchu.Parent)
    pikatchu.CultureEnglishName = "Pikatchu"
    pikatchu.CultureNativeName = "Pika Pika"
    pikatchu.RegionEnglishName = "Pikatchu Land"
    pikatchu.RegionNativeName = "Pika pika Pikaaaaa"
    pikatchu.ThreeLetterWindowsRegionName = "pkt"
    pikatchu.ThreeLetterISORegionName = "pkt"
    pikatchu.TwoLetterISORegionName = "pt"
    pikatchu.CurrencyNativeName = "Pika$"
    pikatchu.CurrencyEnglishName = "PK$"
    pikatchu.ISOCurrencySymbol = "FR"

    Dim dates As DateTimeFormatInfo = New DateTimeFormatInfo()
    dates.DateSeparator = ":"
    dates.TimeSeparator = ":"
    dates.DayNames = New String() {"Pikatchutchu", "Pikadi",
"Marchu", "Pikatchudi", "Pikapikadi", "Vendrechu", "Pikasang"}
    pikatchu.GregorianCalendarFormat = dates

    Dim nombres As NumberFormatInfo = New NumberFormatInfo()
    nombres.CurrencySymbol = "#"
    nombres.CurrencyDecimalDigits = 2
    pikatchu.NumberFormat = nombres

    pikatchu.Register()
End Sub
```

```
//C#
static void Main(string[] args)
{
    CultureAndRegionInfoBuilder pikatchu = new
CultureAndRegionInfoBuilder("Pikatchu-Pokemon",
CultureAndRegionModifiers.None);
    pikatchu.Parent = new CultureInfo("Pokemon-FR");
    pikatchu.LoadDataFromCultureInfo(pikatchu.Parent);
    pikatchu.CultureEnglishName = "Pikatchu";
    pikatchu.CultureNativeName = "Pika Pika";
    pikatchu.RegionEnglishName = "Pikatchu Land";
    pikatchu.RegionNativeName = "Pika pika Pikaaaaa";
    pikatchu.ThreeLetterWindowsRegionName = "pkt";
    pikatchu.ThreeLetterISORegionName = "pkt";
    pikatchu.TwoLetterISORegionName = "pt";
    pikatchu.CurrencyNativeName = "Pika$";
    pikatchu.CurrencyEnglishName = "PK$";
    pikatchu.ISOCurrencySymbol = "FR";

    DateTimeFormatInfo date = new DateTimeFormatInfo();
    date.DateSeparator = ":";
    date.TimeSeparator = ":";
    date.DayNames = new String[] { "Pikatchutchu", "Pikadi", "Marchu",
"Pikatchudi", "Pikapikadi", "Vendrechu", "Pikasam" };
    pikatchu.GregorianCalendarFormat = date;

    NumberFormatInfo nombres = new NumberFormatInfo();
    nombres.CurrencySymbol = "#";
    nombres.CurrencyDecimalDigits = 2;
    pikatchu.NumberFormat = nombres;

    pikatchu.Register();
}
```

Nous commençons par instancier un objet `CultureAndRegionInfoBuilder` avec en paramètres `Pikatchu-Pokemon` et de type `None`.

Nous assignons à la propriété `Parent` une référence vers un objet `CultureInfo` représentant le format de culture `Pokemon-FR`. Puis nous chargeons les informations de culture du format parent grâce à la méthode `LoadDataFromCultureInfo`.

Ensuite nous indiquons successivement toutes les informations nécessaires à la création du format de culture, nous retrouvons des propriétés aux noms quasi identiques à ceux vu précédemment.

Enfin nous associons divers informations spécifique à notre format, comme par exemple le format de date où nous indiquons les séparateurs de date et d'heure, ainsi que le nom des mois, ou bien le format des nombres en indiquant un symbole pour la monnaie et le nombre de chiffre après la virgule.

Une fois enregistré, notre format de culture `Pikatchu-Pokemon` est maintenant disponible dans les options régionales et linguistiques comme le prouve cette image :

Format actuel :

Pika Pika\*

\* Paramètres régionaux personnalisés

Exemples de données affichées dans ce format :

Nombre :	123,456,789.00
Symbole monétaire :	#123,456,789.00
Heure :	12:09:20
Date courte :	08:27:2008
Date longue :	Pikatchudi, 27 August 2008

Nous pouvons constater que le symbole monétaire est bien le #, que le séparateur de date et d'heure est bien le : et que nous sommes le Pikatchudi 27 août 2008.

Enfin, sachez que vous pouvez supprimer les formats de culture de votre machine très simplement en utilisant la méthode statique `CultureAndRegionInfoBuilder.Unregister` (string)

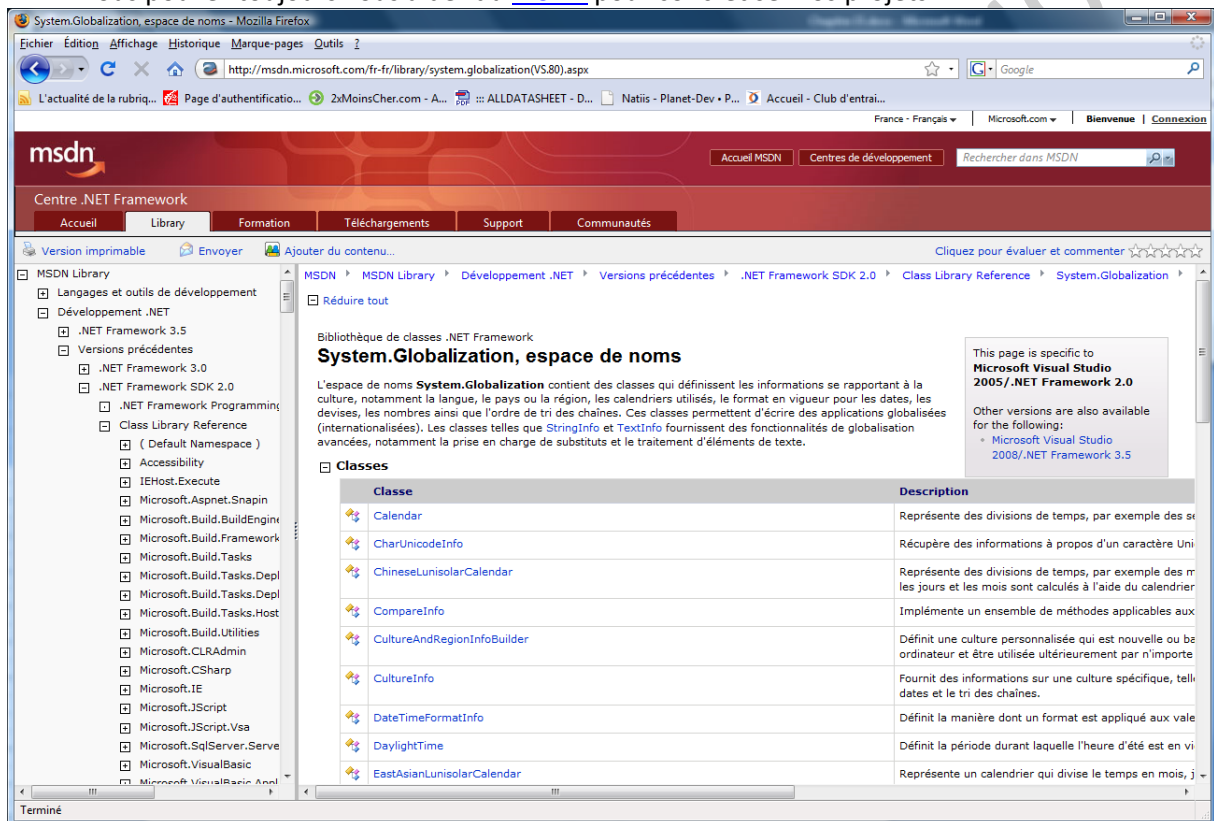
## 4 Conclusion

Vous voici arrivé à la fin des chapitres concernant l'utilisant du Framework .NET. Vous savez désormais comment gérer les paramètres par régions/pays pour adapter vos applications en fonction de la zone du globe dans laquelle elle est exécutée.

Après ce chapitre vous devriez savoir :

- Récupérer des informations aux sujets des formatages de données (DateTimeFormatInfo, NumberFormatInfo).
- Formater des données en utilisant les paramètres de zone (avec la méthode Format ou la méthode ToString)
- Créer ses propres formats avec [CultureAndInfoRegionBuilder](#).

Vous pouvez toujours vous aider du [MSDN](#) pour concrétiser vos projets :



The screenshot shows the MSDN website for the **System.Globalization** namespace. The page title is "Bibliothèque de classes .NET Framework System.Globalization, espace de noms". The main content area contains a table of classes with the following entries:

Classe	Description
<a href="#">Calendar</a>	Représente des divisions de temps, par exemple des se...
<a href="#">CharUnicodeInfo</a>	Récupère des informations à propos d'un caractère Uni...
<a href="#">ChineseLunisolarCalendar</a>	Représente des divisions de temps, par exemple des m...
<a href="#">CompareInfo</a>	Implémente un ensemble de méthodes applicables aux...
<a href="#">CultureAndRegionInfoBuilder</a>	Définit une culture personnalisée qui est nouvelle ou be...
<a href="#">CultureInfo</a>	Fournit des informations sur une culture spécifique, telle...
<a href="#">DateTimeFormatInfo</a>	Définit la manière dont un format est appliqué aux vale...
<a href="#">DaylightTime</a>	Définit la période durant laquelle l'heure d'été est en vi...
<a href="#">EastAsianLunisolarCalendar</a>	Représente un calendrier qui divise le temps en mois, j...

On the right side of the page, there is a note: "This page is specific to Microsoft Visual Studio 2005/.NET Framework 2.0. Other versions are also available for the following: Microsoft Visual Studio 2008/.NET Framework 3.5".