Culture

La clase CultureInfo

La clase **CultureInfo** contiene información específica de la cultura, como el idioma asociado, el idioma secundario, el país/región, el calendario y las convenciones culturales. Esta clase también proporciona acceso a instancias específicas de cultura de **DateTimeFormatInfo**, **NumberFormatInfo**, **CompareInfo** y **TextInfo**. Estos objetos contienen la información necesaria para las operaciones específicas de la cultura, como el formato de fechas y números y la comparación de cadenas.

La clase **String** usa indirectamente esta clase para obtener información sobre la cultura predeterminada.

Los nombres de cultura siguen el estándar RFC 1766 en el formato "<languagecode2>-<country/regioncode2>", donde <languagecode2> es un código de dos letras minúsculas derivado de la ISO 639-1 y <country/regioncode2> es unas mayúsculas dos Código derivado de la ISO 3166. Por ejemplo, el inglés de EE. UU. es "en-US". Algunos nombres de cultura tienen prefijos que especifican el script; por ejemplo, "Cy-" especifica el guion cirílico, "Lt-" especifica el guion latino.

En el último par de artículos, hemos hablado acerca de cuán útil es la clase **CultureInfo** cuando necesita un control total de cómo se muestran los números y las fechas en su aplicación. También hemos hablado acerca de cómo puede verificar y modificar qué cultura debe usar su aplicación como alternativa. Con todo eso en su lugar, es hora de profundizar en la clase **CultureInfo** real, para ver cómo podemos aprovecharla al máximo.

La clase **CultureInfo** es parte del espacio de nombres de **System.Globalization**, así que asegúrese de importarlo cada vez que pruebe los ejemplos:

using System.Globalization;

El identificador de cultura "**0x040A**" se puede usar para crear un **CultureInfo** para "**español - España**" que usa el orden tradicional, en lugar del orden de clasificación internacional predeterminado, identificador de cultura "**0x0c0a**".

Las culturas generalmente se agrupan en tres grupos: la cultura invariante, las culturas neutrales y las culturas específicas.

La cultura invariante es insensible a la cultura. Puede especificar la cultura invariante por nombre usando una cadena vacía (""). **CultureInfo.InvariantCulture** recupera una instancia de la cultura invariante. Se asocia con el idioma inglés, pero no con un país/región. Se puede usar en casi cualquier método en el espacio de nombres de Globalización que requiera de una cultura. La cultura invariable debe ser utilizada solo por procesos que requieren resultados independientes de la cultura, como los servicios

del sistema; de lo contrario, produce resultados que pueden ser lingüísticamente incorrectos o culturalmente inapropiados.

Una cultura neutral es una cultura que está asociada con un idioma, pero no con un país/región. Una cultura específica es una cultura que está asociada con un idioma y un país/región. Por ejemplo, "**fr**" es una cultura neutral y "**fr-FR**" es una cultura específica. Tenga en cuenta que "**zh-CHS**" (chino simplificado) y "**zh-CHT**" (chino tradicional) son culturas neutrales.

DateTimeFormatInfo o **NumberFormatInfo** se pueden crear solo para la cultura invariante o para culturas específicas, no para culturas neutrales.

Los usuarios están acostumbrados a modificar o anular algunos de los valores asociados con la cultura actual de Windows a través de las Opciones regionales y de idioma en el Panel de control. Por ejemplo, el usuario puede optar por mostrar la fecha en un formato diferente o utilizar una moneda diferente a la predeterminada para la cultura. Si la propiedad **CultureInfo.UseUserOverride** se establece en true, las propiedades de la instancia **CultureInfo.DateTimeFormat**, la instancia **CultureInfo.NumberFormat** y la instancia **CultureInfo.TextInfo** también se recuperan de la configuración del usuario. Si las configuraciones del usuario son incompatibles con la cultura asociada con **CultureInfo**, los resultados de los métodos y los valores de las propiedades no están definidos.

Para usar la configuración predeterminada de .NET Framework para la moneda, podemos usar una sobrecarga del constructor **CultureInfo** que acepte un parámetro **useUserOverride** y establecerlo en falso.

Esta clase implementa la interfaz **ICloneable** para habilitar la duplicación de objetos **CultureInfo**. También implementa **IFormatProvider** para suministrar información de formato a las aplicaciones.

Constructores públicos

ctor # 1	Sobrecarga: Inicializa una nueva instancia de la clase CultureInfo basada en la cultura especificada por el identificador de cultura. .ctor(int culture)
ctor # 2	Sobrecarga: Inicializa una nueva instancia de la clase CultureInfo basada en la cultura especificada por nombre. .ctor(string name)
ctor # 3	Sobrecarga: Inicializa una nueva instancia de la clase CultureInfo basada en la cultura especificada por el identificador de cultura y en el booleano que especifica si se debe usar la configuración de cultura seleccionada por el usuario del sistema. .ctor(int culture, bool useUserOverride)
ctor # 4	Sobrecarga: Inicializa una nueva instancia de la clase CultureInfo basada en la cultura especificada por nombre y en el Booleano que especifica si se debe usar

	la configuración de cultura seleccionada por el usuario del sistema. .ctor(string name, bool useUserOverride)

Propiedades públicas

Calendar	Solo lectura Obtiene el calendario predeterminado utilizado por la cultura.
CompareInfo	Solo lectura Obtiene el CompareInfo que define cómo comparar cadenas para la cultura.
CurrentCulture	Solo lectura Obtiene el CultureInfo que representa la cultura utilizada por el hilo actual.
CurrentUICulture	Solo lectura Obtiene el CultureInfo que representa la cultura actual utilizada por el Administrador de recursos para buscar recursos específicos de la cultura en tiempo de ejecución.
DateTimeFormat	Read-write Obtiene o establece un DateTimeFormatInfo que define el formato culturalmente apropiado para mostrar fechas y horas.
DisplayName	Solo lectura Obtiene el nombre de la cultura en el formato "<languagefull> (<country / regionfull>)" en el idioma de la versión localizada de .NET Framework.
EnglishName	Solo lectura Obtiene el nombre de la cultura en el formato "<languagefull> (<country / regionfull>)" en inglés.
InstalledUICulture	Solo lectura Obtiene la CultureInfo que representa la cultura instalada con el sistema operativo.
InvariantCulture	Solo lectura Obtiene el CultureInfo que es independiente

	de la cultura invariante.
IsNeutralCulture	Solo lectura Obtiene un valor que indica si el CultureInfo actual representa una cultura neutral.
IsReadOnly	Solo lectura Obtiene un valor que indica si el CultureInfo actual es de solo lectura.
LCID	Solo lectura Obtiene el identificador de cultura para el CultureInfo actual.
Name	Solo lectura Obtiene el nombre de la cultura en el formato "<languagecode2> - <country / regioncode2>".
NativeName	Solo lectura Obtiene el nombre de la cultura en el formato "<languagefull> (<country / regionfull>)" en el idioma que la cultura está configurada para mostrar.
NumberFormat	Lectura-escritura Obtiene o establece un NumberFormatInfo que define el formato culturalmente apropiado de visualización de números, moneda y porcentaje.
OptionalCalendars	Solo lectura Obtiene la lista de calendarios opcionales que puede usar la cultura.
Parent	Solo lectura Obtiene el CultureInfo que representa la cultura principal del CultureInfo actual.
TextInfo	Solo lectura Obtiene la información de texto que define el sistema de escritura asociado con la cultura.
ThreeLetterISOLanguageName	Solo lectura Obtiene el código de tres letras ISO 639-2 para el idioma de CultureInfo actual.
ThreeLetterWindowsLanguageName	Solo lectura Obtiene el código de tres letras para el idioma

	como se define en la API de Windows.
TwoLetterISOLanguageName	Solo lectura Obtiene el código de dos letras ISO 639-1 para el idioma de CultureInfo actual.
UseUserOverride	Solo lectura Obtiene un valor que indica si el CultureInfo actual usa la configuración de cultura seleccionada por el usuario.

Métodos Públicos

ClearCachedData	Actualiza la información relacionada con la cultura almacenada en caché.
Clon	Crea una copia de la actual CultureInfo.
CreateSpecificCulture	Crea una CultureInfo que representa la cultura específica que está asociada con el nombre especificado.
Equals	Anulado-Override: determina si el objeto especificado es de la misma cultura que el CultureInfo actual.
GetCultures	Obtiene la lista de culturas admitidas filtrada por los CultureTypes especificados.
GetFormat	Obtiene un objeto que define cómo formatear el tipo especificado.
GetHashCode	Anulado-Override: Sirve como una función de hash para el CultureInfo actual, adecuado para su uso en algoritmos de hash y estructuras de datos, como una tabla de hash.
GetType (heredado de System.Object)	Derivado de System.Object, la clase base primaria para todos los objetos.
ReadOnly	Devuelve un contenedor de solo lectura alrededor de CultureInfo especificado.
ToString	Anulado-Override: devuelve una cadena que contiene el nombre de CultureInfo actual en el formato "<languagecode2> - <country / regioncode2>".

Métodos protegidos

Finalize	Derivado de System.Object, la clase base primaria para todos los objetos.
MemberwiseClone	Derivado de System.Object, la clase base principal para todos los objetos.

Culturas neutras y específicas

Hasta ahora solo hemos utilizado culturas específicas, que es una cultura que especifica un idioma y un país/región. Un ejemplo de esto es la cultura en-EE. UU. que establece claramente que el idioma deseado debe ser el inglés y que la región es los EE. UU. Una alternativa a eso es la cultura en-GB, que es el mismo idioma, pero con Gran Bretaña como la región en lugar de los Estados Unidos.

Habrá momentos en que estas diferencias sean importantes, en cuyo caso debe usar estas versiones específicas de la región de la clase CultureInfo. Por otro lado, también habrá situaciones en las que el inglés es solo un idioma y no desea vincular este idioma a un país o región específica. Para esto, el.NET Framwork define las llamadas culturas neutrales, que solo especifican un idioma. De hecho, tanto en-US como en-GB se heredan de una cultura tan neutral y se puede acceder a él desde la propiedad Parent.

Por ejemplo:

```
using System;
using System.Globalization;

public class Ejemplo_Culture
{

    public static void Main(String[] args)
    {
        CultureInfo enGb = new CultureInfo("en-GB");
        CultureInfo enUs = new CultureInfo("en-US");
        Console.WriteLine(enGb.DisplayName);
        Console.WriteLine(enUs.DisplayName);
        Console.WriteLine(enGb.Parent.DisplayName);
        Console.WriteLine(enUs.Parent.DisplayName);
        Console.ReadKey();
    }

}
```

Resultado

```
Inglés (Reino Unido)

Inglés (Estados Unidos)

Inglés

Inglés
```

Conseguir la cultura adecuada

Como hemos visto podemos obtener la clase de CultureInfo deseada pasando el identificador de país/región de idioma al constructor de la clase. Pero como podría estar buscando una cultura neutral, como se describió anteriormente, también podría pasar un identificador de idioma:

```
CultureInfo en = new CultureInfo("en");
```

.NET Framework nos devolverá una instancia de CultureInfo independiente de la región en inglés.

Otra forma de identificar una cultura específica es con el llamado LCID (LoCale ID) que lo podemos encontrar como una propiedad en instancias de CultureInfo existentes, pero si conoce el ID, también puede usarlo para crear una instancia de un objeto CultureInfo.

Por ejemplo:

```
CultureInfo enUs = new CultureInfo(1033);
```

En la mayoría de las situaciones, es mucho más fácil usar el especificador de idioma/país/región.

Obteniendo una lista de Culturas disponibles

Podemos obtener una cultura específica y usarla para varios propósitos, pero necesitaremos una lista de las culturas disponibles, por ejemplo, para permitirle al usuario seleccionar un idioma y/o país/región. .NET Framework también nos lo pone fácil, por ejemplo:

```
using System;
using System.Globalization;

public class Ejemplo_Culture
{

    public static void Main(String[] args)
    {
        CultureInfo[] CulturasEspecificas =
CultureInfo.GetCultures(CultureTypes.SpecificCultures);
        foreach (CultureInfo cultura in CulturasEspecificas)
            Console.WriteLine(cultura.DisplayName);
        Console.WriteLine("Total: " + CulturasEspecificas.Length);
        Console.ReadKey();

    }

}
```

Resultado

```
Afar (Yibuti)

Afar (Eritrea)

Afar (Etiopía)

Afrikáans (Namibia)

…

isiZulu (Sudáfrica)

Total: 563
```

En la primera línea de código, usamos el método estático GetCultures en la clase CultureInfo para obtener una lista de culturas. Requiere el parámetro CultureTypes, que especifica qué tipo de culturas está buscando. En este ejemplo hemos pedido las culturas específicas, que son las culturas que están vinculadas tanto a un idioma como a un país/región.

Si estamos elaborando una lista de los idiomas disponibles, sin importar en qué país o región están relacionados podemos usar la cultura neutral, por ejemplo:

```
using System;
using System.Globalization;

public class Ejemplo_Culture
{

    public static void Main(String[] args)
    {
        CultureInfo[] CulturasNeutrales =
CultureInfo.GetCultures(CultureTypes.NeutralCultures);
        foreach (CultureInfo cultura in CulturasNeutrales)
            Console.WriteLine(cultura.DisplayName);
        Console.WriteLine("Total: " + CulturasNeutrales.Length);
        Console.ReadKey();

    }

}
```

Resultado

```
Todos los idiomas (todos los países)

Afar
```

```
Afrikaans

…

Chino (tradicional) heredado

Total: 280
```

Propiedades y métodos importantes de CultureInfo

Estos miembros pueden ayudarnos a lograr muchas cosas útiles con respecto a la cultura.

DateTimeFormat

Con la propiedad DateTimeFormat, se obtiene acceso a la información sobre cómo se debe formatear la fecha y la hora, así como a una gran cantidad de información útil sobre el calendario de la cultura determinada. Un ejemplo de esto son las propiedades FirstDayOfWeek y CalendarWeekRule: pueden informarnos en qué día comienza la semana y cómo se decide la primera semana del calendario del año, por ejemplo, solo el primer día o la primera semana completa:

```
using System;
using System.Globalization;

public class Ejemplo_Culture
{

    public static void Main(String[] args)
    {
        CultureInfo enUs = new CultureInfo("en-US");
        CultureInfo esES = new CultureInfo("es-ES");
        Console.WriteLine("El Primer Día de la Semana Anglosajón es: " +
enUs.DateTimeFormat.FirstDayOfWeek.ToString());
        Console.WriteLine("El Primer Día de la Semana Hispanoamericano es: " +
esES.DateTimeFormat.FirstDayOfWeek.ToString());
        Console.WriteLine("La Primera semana del calendario del año Anglosajón
es: " + enUs.DateTimeFormat.CalendarWeekRule.ToString());
        Console.WriteLine("La Primera semana del calendario del año
Hispanoamericano: " + esES.DateTimeFormat.CalendarWeekRule.ToString());
        Console.ReadKey();
    }

}
```

Resultado

```
El Primer Día de la Semana Anglosajón es: Sunday
```

```
El Primer Día de la Semana Hispanoamericano es: Monday

La Primera semana del calendario del año Anglosajón es: FirstDay

La Primera semana del calendario del año Hispanoamericano: FirstFourDayWeek
```

También podemos obtener información sobre los nombres de mes y día para la cultura específica utilizando propiedades como MonthNames y métodos como GetMonthName (). Por ejemplo:

```
using System;
using System.Globalization;

public class Ejemplo_Culture
{

    public static void Main(String[] args)
    {
        CultureInfo esES = new CultureInfo("es-ES");

        foreach (string nombreMes in esES.DateTimeFormat.MonthNames)
            Console.WriteLine(nombreMes);
        Console.WriteLine("Mes Actual: " +
esES.DateTimeFormat.GetMonthName(DateTime.Now.Month));
        Console.ReadKey();
    }

}
```

Resultado

```
enero

febrero

marzo

abril

mayo

junio

julio

agosto

septiembre
```

```
octubre

noviembre

diciembre



Mes Actual: febrero
```

Lo mismo se puede lograr por días, usando la propiedad DayNames y el método GetDayName ():

```
using System;
using System.Globalization;

public class Ejemplo_Culture
{

    public static void Main(String[] args)
    {
        CultureInfo esES = new CultureInfo("es-ES");

        foreach (string nombreDia in esES.DateTimeFormat.DayNames)
            Console.WriteLine(nombreDia);
        Console.WriteLine("Hoy es: " +
esES.DateTimeFormat.GetDayName(DateTime.Now.DayOfWeek));
        Console.ReadKey();
    }

}
```

Resultado

```
domingo

lunes

martes

miércoles

jueves

viernes

sábado

Hoy es: martes
```

Hay muchas propiedades y métodos en la propiedad DateTimeFormat, por ejemplo: DateSeparator, YearMonthPattern y así sucesivamente.

Formato numérico

Al igual que DateTimeFormat tiene información sobre fechas, podemos acceder a información sobre cómo una cultura específica trata los números de la propiedad NumberFormat. Esta información se usa cada vez que solicitamos una representación visual de un número, por ejemplo, al convertirlo en una cadena y escribirlo en la consola, pero también podemos acceder a la información, utilizando las propiedades y los métodos en la propiedad NumberFormat. Por ejemplo:

```
using System;
using System.Globalization;

public class Ejemplo_Culture
{

    public static void Main(String[] args)
    {
        CultureInfo enUs = new CultureInfo("en-US");
        Console.WriteLine(enUs.DisplayName + ":");
        Console.WriteLine("Separador de Grupos de Números: " +
enUs.NumberFormat.NumberGroupSeparator);
        Console.WriteLine("Separador de Números Decimales: " +
enUs.NumberFormat.NumberDecimalSeparator);

        CultureInfo esES = new CultureInfo("es-ES");
        Console.WriteLine(esES.DisplayName + ":");
        Console.WriteLine("Separador de Grupos de Números: " +
esES.NumberFormat.NumberGroupSeparator);
        Console.WriteLine("Separador de Números Decimales: " +
esES.NumberFormat.NumberDecimalSeparator);
        Console.ReadKey();
    }

}
```

Resultado

```
Inglés (Estados Unidos):

Separador de Grupos de Números: ,

Separador de Números Decimales: .

Español (España):

Separador de Grupos de Números: .
```

```
Separador de Números Decimales: ,
```

Utilizamos las propiedades **NumberGroupSeparator** y **NumberDecimalSeparator** para obtener información sobre cómo se muestra un número, por ejemplo, 1,000.00 o 1,000,00 para las culturas inglesa y alemana. También hay propiedades coincidentes para las monedas como **CurrencyGroupSeparator** y **CurrencyDecimalSeparator** y también para porcentajes **PercentGroupSeparator** y **PercentDecimalSeparator**.

Para el formato moneda tenemos la propiedad **NumberFormat** al que también podemos decirle qué símbolo usa una cultura determinada para mostrar una cantidad monetaria, para ello usamos la propiedad **CurrencySymbol**:

```
using System;
using System.Globalization;

public class Ejemplo_Culture
{

    public static void Main(String[] args)
    {
        CultureInfo enUs = new CultureInfo("en-US");
        Console.WriteLine(enUs.DisplayName + " - Símbolo de la Moneda: " +
enUs.NumberFormat.CurrencySymbol);
        CultureInfo esES = new CultureInfo("es-ES");
        Console.WriteLine(esES.DisplayName + " - Símbolo de la Moneda: " +
esES.NumberFormat.CurrencySymbol);
        CultureInfo ruRu = new CultureInfo("ru-RU");
        Console.WriteLine(ruRu.DisplayName + " - Símbolo de la Moneda: " +
ruRu.NumberFormat.CurrencySymbol);
        Console.ReadKey();
    }

}
```

Resultado

```
Inglés (Estados Unidos) - Símbolo de la Moneda: $

Español (España) - Símbolo de la Moneda: ?

Ruso (Rusia) - Símbolo de la Moneda: ?
```

Nombres e identificadores

Tenemos propiedades que representan la instancia de **CultureInfo**. Ya hemos usado algunos de ellos, por ejemplo, Name. Estas funcionan de la siguiente manera, primero, aquí hay una lista de las propiedades disponibles que se usan para identificar un **CultureInfo**:

- **El Name identificará un CultureInfo** en el formato de código de idioma/ código de país, por ejemplo, "en-US" para inglés en los EE. UU., En-GB para inglés en Gran Bretaña, etc. Si no se especifica ningún país/región, solo se devolverá la primera parte, por ejemplo, "en" para inglés.
- **TwoLetterISOLanguageName** hará prácticamente lo mismo que Name, pero solo devolverá el código de idioma, sin importar si un país/región ha sido especificado o no. Por ejemplo, "en" se devolverá para "en-US" y "en-GB".
- **ThreeLetterISOLanguageName** funciona de forma muy similar a TwoLetterISOLanguageName, pero devuelve tres letras en lugar de dos, según lo especificado por el estándar ISO 639-2.
- **EnglishName** devolverá el nombre del idioma en inglés. Si se ha especificado un país/región, esto se agregará al resultado, dentro de un conjunto de paréntesis.
- **NativeName** devolverá el nombre del idioma en el idioma especificado por la instancia de CultureInfo. Si se ha especificado un país/región, esto se agregará al resultado, dentro de un conjunto de paréntesis.

FIN