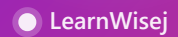


Fundamentos de localización: cultura, idioma y recursos

Guía de la lección · Guía del laboratorio / examen



Guía de la lección

Qué abarca realmente la localización más allá de la traducción: nombres de cultura como en, de-DE y fr-CA, archivos de recursos .resx leídos a través de ResourceManager, el repliegue de cultura de .NET hacia el recurso neutro, claves semánticas estables en lugar de frases en inglés dentro del código, y el formato sensible a la cultura de fechas, números y moneda. Qué significa esto para un desarrollador que conoce los fundamentos de C# y los controles de Wisej.NET y ahora tiene que publicar la aplicación en más de un idioma, y cómo leer este módulo.



Enfoque de producción

Qué abarca realmente la localización más allá de la traducción: nombres de cultura como en, de-DE y fr-CA, archivos de recursos .resx leídos a través de ResourceManager, el repliegue de cultura de .NET hacia el recurso neutro, claves semánticas estables en lugar de frases en inglés dentro del código, y el formato sensible a la cultura de fechas, números y moneda.

1. El problema en producción

GlobalDesk es un pequeño service desk escrito para una sola oficina. Cada encabezado, etiqueta de botón y mensaje de confirmación es un literal en inglés tecleado directamente en el C#, y el dashboard imprime importes y fechas tal y como los escribe una máquina estadounidense. Cuando abre una sucursal alemana, ninguno de esos textos tiene adónde ir, y el desarrollador que empieza editando etiquetas a mano se encuentra con la segunda mitad del problema: una etiqueta en alemán sobre un importe que sigue leyéndose **1,850.75** no es una pantalla alemana.

2. Conceptos clave

Concepto	Por qué importa
localización	Adaptar una aplicación al idioma y a la cultura del usuario: texto traducido, fechas, números e importes correctamente formateados, espacio para palabras más largas y un diseño que pueda reflejarse para los idiomas de derecha a izquierda.
nombre de cultura	Un identificador como <code>en</code> , <code>de-DE</code> o <code>fr-CA</code> que nombra un idioma y, opcionalmente, una región; la parte del idioma decide qué recursos se prefieren y el nombre completo gobierna las convenciones de formato.
recurso neutro	El conjunto completo de recursos de repliegue, guardado en un archivo como <code>Resources/Strings.resx</code> sin cultura en su nombre; toda clave que pida la aplicación debe existir aquí.
<code>Wisej.Resources.ResourceManager</code>	El objeto que toma un nombre base de recursos y un ensamblado y devuelve el valor de una clave en la cultura que mejor coincide, de modo que tu código nunca abre un archivo por su cuenta.
repliegue de cultura	La regla de .NET para encontrar un valor: probar la cultura completa, después la cultura solo de idioma, después el recurso neutro, y usar el primero que tenga la clave.
clave de recurso semántica	Un nombre estable que describe el significado y no la redacción, como <code>CustomerEditor.Save</code> o <code>Validation.Required</code> , para que el texto en inglés se pueda reescribir sin tocar ningún otro idioma.
<code>Application.CurrentCulture</code>	La cultura de la sesión actual de Wisej.NET; dos usuarios con la sesión iniciada a la vez pueden tener dos valores distintos, y por eso la cultura nunca debe vivir en un campo estático.

3. Desarrollo de la lección

La traducción es solo la mitad del trabajo

Dentro de la palabra "localización" se esconden dos trabajos distintos, y quien empieza suele ver solo el primero. El primero es la **localización del texto de la interfaz**: las palabras que lee el usuario, como el encabezado de bienvenida de GlobalDesk, la etiqueta de navegación Customers, el botón Save y el mensaje que se muestra tras guardar un cliente. Esas palabras las escribes tú, viven en archivos de recursos y el idioma de la sesión decide qué versión se devuelve.

El segundo trabajo es el **formato sensible a la cultura**. Una fecha de vencimiento, un número de tickets y el total de un pedido no son palabras en absoluto; son valores. El mismo importe se escribe `$1,850.75` con `en-US` y `1.850,75 €` con `de-DE` , y ninguna de esas cadenas pertenece a un archivo de recursos ni a una columna de la base de datos. Tú guardas el `decimal` y el `DateTime` , y les das formato en el momento de ponerlos en pantalla, pasando la cultura de la sesión como proveedor de formato.

Esa separación es la razón de que el laboratorio termine con una nota escrita. En cuanto puedas señalar algo y decir "esto son datos" o "esto es presentación", el resto del curso tiene dónde sujetarse. Compón una cadena como `"€ " + amount` y habrás escrito sin darte cuenta un símbolo de moneda fijo para todos los países a los que vayas a llegar.

Leer un nombre de cultura: en, de-DE, fr-CA

Un nombre de cultura combina un idioma con convenciones regionales. Un nombre **solo de idioma** (neutro) es únicamente el idioma: `en`, `de`, `fr`. Un nombre de **idioma y región** añade un país: `en-US`, `de-DE`, `fr-CA`, `it-IT`. En .NET son valores `CultureInfo`, y Wisej.NET expone el que se aplica a la sesión actual como `Application.CurrentCulture`.

Dos nombres pueden coincidir en el idioma y discrepar en todo lo demás. Una sesión con `en-GB` y otra con `en-US` leen los mismos recursos en inglés, porque ambas seleccionan el idioma `en`; aun así escriben las fechas de forma distinta y usan símbolos de moneda diferentes. Lo contrario también ocurre: `fr-CA` y `fr-FR` comparten el texto en francés, pero no las convenciones de números ni de moneda. Así que la parte del idioma responde a "¿qué palabras?" y el nombre completo responde a "¿cómo se escriben los valores?".

Para GlobalDesk esto importa de inmediato. El panel de vista previa de la cultura que construyes en el laboratorio muestra la fecha de hoy, una cantidad y un importe monetario uno al lado del otro, para que veas cambiar tres valores mientras las etiquetas que hay encima se mantienen en un único idioma.

Archivos .resx y ResourceManager

Un archivo `.resx` es un archivo de recursos XML que Visual Studio edita como una cuadrícula de clave y valor. Wisej.NET no necesita un formato de traducción propio: usa el modelo estándar de .NET, así que los archivos `.resx`, `CultureInfo`, los ensamblados satélite y las herramientas RESX de terceros funcionan sin cambios. Añades `Resources/Strings.resx` como archivo neutro y, más adelante, `Strings.de.resx` a su lado con las mismas claves y los valores en alemán.

Tu código nunca abre esos archivos. Le pide una clave a un `ResourceManager` construido a partir del nombre base de recursos y su ensamblado. Envolver eso en un pequeño helper estático te da un único sitio donde decidir qué aspecto tiene una clave ausente:

```
using Wisej.Resources;

public static class Texts
{
    private static readonly ResourceManager RS =
        new ResourceManager("GlobalDesk.Resources.Strings", typeof(Texts).Assembly);

    public static string Get(string key)
        ⇒ RS.GetString(key) ?? $"[{{key}}]";
}
```

Las claves son la otra mitad de la disciplina. Usa `CustomerEditor.Save`, `Navigation.Customers` o `Validation.Required`: nombres que describen el significado, no la redacción. Usar la frase en inglés como clave hace que cada reescritura rompa todos los archivos de traducción. Y localiza solo el texto de

producto: etiquetas, menús, tooltips y mensajes de validación y confirmación. Los identificadores de los registros, los nombres de tipos de excepción y los nombres de columnas de la base de datos se quedan como están.

Repliegue: tres intentos y después el archivo neutro

Cuando una sesión que se ejecuta con `fr-CA` pide una clave, `.NET` no se rinde si `Strings.fr-CA.resx` no tiene nada para ella. Prueba primero el recurso más específico, después el menos específico `fr`, después el recurso neutro, y devuelve el primer valor que encuentra. Por eso el archivo neutro tiene que estar completo: es el último sitio donde mira la búsqueda, y una clave que falta ahí falta en todas partes.

El repliegue es una red de seguridad, no una estrategia de traducción. Evita que un archivo alemán a medias deje una pantalla en blanco, pero también oculta los huecos: el usuario ve inglés donde esperaba alemán y quizá no lo informe nunca. Así que mantén el repliegue activo y busca los huecos a propósito, que es justo para lo que sirve el marcador entre corchetes del helper `Texts`. Una etiqueta que se muestra como `[Dashboard.Welcome]` es un error que puedes ver.

Preparar el dashboard de GlobalDesk

Preparar una página para la localización es mecánico una vez tomadas las decisiones. Construye el dashboard con normalidad, después recórrelo y pregunta de cada cadena visible: ¿quién lee esto? El encabezado de bienvenida, el botón Customers y todas las etiquetas de la vista previa los leen los usuarios, así que cada uno recibe una clave en `Resources/Strings.resx` y una llamada a `Texts.Get` en lugar de su literal. Doce claves son un comienzo realista para una pantalla.

Después haz la otra mitad. El panel de vista previa no debe imprimir texto que hayas compuesto tú; da formato a valores reales, pasando `Application.CurrentCulture` a `ToString` para que el patrón de fecha, los separadores y el símbolo de moneda se elijan para la sesión. Esa cultura es por sesión, así que dos usuarios pueden leer dos idiomas en el mismo proceso del servidor, y un campo de cultura estático dejaría que uno de ellos cambiara la pantalla del otro.



Regla de las capas de localización

Decide dónde vive cada cadena antes de teclearla. Los recursos del diseñador pertenecen a una pantalla diseñada y se aplican cuando se crean sus controles; los recursos de aplicación compartidos en `.resx` llevan el texto que genera el código o que reutilizan varias pantallas; `Resources-[LANG].resx` sobrescribe las etiquetas de sistema de Wisej.NET. Nunca guardes una fecha, un número o un importe ya formateados: conserva el valor y dale formato para la cultura activa en el momento de mostrarlo. Y recuerda que la cultura es por sesión, así que el idioma de un usuario nunca debe cambiar la pantalla de otro.



Objetivos de aprendizaje

Al terminar este módulo serás capaz de:

- ✓ Explicar la diferencia entre la localización del texto de la interfaz y el formato sensible a la cultura.

- ✓ Identificar nombres de cultura neutros, solo de idioma y de idioma y región, como en, de-DE y fr-CA.
- ✓ Explicar cómo participan los archivos .resx y ResourceManager en la localización de .NET.
- ✓ Describir el repliegue de cultura de .NET en términos para principiantes.
- ✓ Construir una página pequeña de Wisej.NET con todas las cadenas visibles preparadas para la localización.

Guía del laboratorio / examen

Lo que construirás en el laboratorio práctico, el enfoque sugerido y los entregables requeridos.



Laboratorio práctico

Crea la página del dashboard de GlobalDesk con un encabezado de bienvenida, un botón de navegación Customers y un pequeño panel de vista previa de la cultura. Añade Resources/Strings.resx como recurso neutro con al menos doce claves semánticas estables como Navigation.Customers y CustomerEditor.Save, escribe un helper estático Texts sobre Wisej.Resources.ResourceManager que devuelva un marcador de clave entre corchetes cuando falte una clave, y sustituye por una llamada a Texts.Get cada etiqueta escrita directamente en la página. Después rellena el panel de vista previa con valores reales: da formato a la fecha de hoy, a una cantidad y a un importe monetario con Application.CurrentCulture en lugar de componer las cadenas a mano, y termina con una nota breve de laboratorio que separe el valor almacenado de su presentación sensible a la cultura.

Punto de partida

Abre el proyecto Wisej.NET GlobalDesk en Visual Studio. Es un pequeño service desk con una página principal, un esbozo de editor de clientes y una cuadrícula de contactos, y ahora mismo todas las palabras visibles para el usuario son literales en inglés tecleados en el C#, con importes y fechas impresos como el servidor los formatee. En este laboratorio añades la página del dashboard y le das a la aplicación su primera capa de localización: claves de recursos para el texto y formato según la cultura para los valores.

Enfoque sugerido

- ✓ Añade una página `Dashboard` y coloca un encabezado de bienvenida, un botón de navegación `btnCustomers` y un pequeño panel de vista previa de la cultura con tres etiquetas de solo lectura para una fecha, una cantidad y un importe. Constrúyelo primero en inglés, exactamente como lo harías siempre.
- ✓ Añade `Resources/Strings.resx` como recurso neutro y rellénalo con al menos doce claves semánticas como `Dashboard.Welcome`, `Navigation.Customers`, `CustomerEditor.Save` y `Preview.Amount`. Escribe aquí los valores en inglés, y en ningún otro sitio.
- ✓ Añade una clase estática `Texts` con un único `Wisej.Resources.ResourceManager` construido a partir del nombre base de recursos y del propio ensamblado del helper, y un método `Get(string key)` que devuelva el valor o un marcador `[key]` entre corchetes cuando la búsqueda no encuentre nada.
- ✓ Recorre la página y sustituye por una llamada a `Texts.Get` cada etiqueta escrita en el código. Después pide a propósito una clave que nunca añadiste y confirma que el marcador entre corchetes aparece en pantalla en lugar de una etiqueta vacía.
- ✓ Rellena el panel de vista previa con valores reales: llama a `ToString` sobre `DateTime.Today`, sobre una cantidad y sobre un importe `decimal`, pasando `Application.CurrentCulture` cada vez. No concatenes nunca tú mismo un símbolo ni un separador, y después escribe la nota de laboratorio.

Entregables requeridos

- ✓ Página del dashboard de GlobalDesk cuyas etiquetas proceden todas de claves de recursos, sin texto visible para el usuario escrito en el código
- ✓ Archivo neutro Resources/Strings.resx con al menos doce claves semánticas estables
- ✓ Helper Texts sobre Wisej.Resources.ResourceManager que marca una clave ausente en lugar de devolver una cadena vacía
- ✓ Panel de vista previa de la cultura que da formato a una fecha, un número y una moneda con Application.CurrentCulture
- ✓ Nota de laboratorio que explica qué valores son datos y cuáles son presentación sensible a la cultura

Preguntas de repaso

- ✓ Cambias la redacción en inglés del botón Save de "Save" a "Save customer". ¿Qué archivos tienen que cambiar si tu clave es `CustomerEditor.Save`, y qué archivos habrían tenido que cambiar si la clave hubiera sido la propia frase en inglés?
- ✓ Tu helper `Texts` devuelve `[Preview.Amount]` en una etiqueta. Nombra dos errores distintos que produzcan exactamente ese marcador y explica cómo los distinguirías sin un depurador.
- ✓ Un compañero propone guardar el total del pedido en la base de datos como la cadena "1.850,75 €" para que los usuarios alemanes siempre lo vean bien. Explica qué se rompe y qué parte de esa cadena son datos y qué parte es presentación.



Regla de las capas de localización

Decide dónde vive cada cadena antes de teclearla. Los recursos del diseñador pertenecen a una pantalla diseñada y se aplican cuando se crean sus controles; los recursos de aplicación compartidos en .resx llevan el texto que genera el código o que reutilizan varias pantallas; Resources-[LANG].resx sobrescribe las etiquetas de sistema de Wisej.NET. Nunca guardes una fecha, un número o un importe ya formateados: conserva el valor y dale formato para la cultura activa en el momento de mostrarlo. Y recuerda que la cultura es por sesión, así que el idioma de un usuario nunca debe cambiar la pantalla de otro.