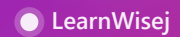


Arquitectura de localización en producción, pruebas y proyecto final

Guía de la lección · Guía del laboratorio / examen



Guía de la lección

Localización que sobrevive a producción: un `LocalizationService` ligero sobre `ResourceManager`, código de dominio que devuelve valores en lugar de frases, una política observable para las claves ausentes, una matriz de pruebas de localización, la seudolocalización que saca a la luz las cadenas escritas en el código, el despliegue de ensamblados satélite y el proyecto final de GlobalDesk. 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

Localización que sobrevive a producción: un `LocalizationService` ligero sobre `ResourceManager`, código de dominio que devuelve valores en lugar de frases, una política observable para las claves ausentes, una matriz de pruebas de localización, la seudolocalización que saca a la luz las cadenas escritas en el código, el despliegue de ensamblados satélite y el proyecto final de GlobalDesk.

1. El problema en producción

Después de seis módulos GlobalDesk habla dos idiomas, pero la maquinaria está dispersa: recursos de diseñador en el editor de clientes, un helper sobre los archivos `.resx` compartidos, un selector de idioma, un manejador de `CultureChanged` y una demostración de RTL. Todavía nada declara qué ocurre cuando una clave no tiene traducción, y nadie ha demostrado que la compilación en alemán siga cabiendo en una ventana estrecha. Un sistema de localización de producción es predecible: sabe de dónde viene cada cadena, qué cultura está activa en esta sesión, qué aspecto tiene una traducción ausente y qué le hace un cambio de idioma a las pantallas abiertas. Este módulo convierte esas cuatro respuestas en un pequeño servicio, una matriz de pruebas escrita y el proyecto final terminado.

2. Conceptos clave

Concepto	Por qué importa
<code>LocalizationService</code>	Una clase pequeña que centraliza las decisiones de tu aplicación alrededor de la localización de .NET en lugar de sustituirla: un nombre base de recursos, una política de claves ausentes, un conjunto de métodos de formato.
Política de claves ausentes	El comportamiento elegido y probado cuando <code>GetString</code> no devuelve nada: un marcador <code>[Key]</code> visible mientras desarrollas, y la clave registrada más el texto neutro de repliegue en producción.
Texto de producto, de sistema y de diagnóstico	Tres familias con tres propietarios: tus propias cadenas <code>.resx</code> , las etiquetas propias de Wisej.NET a través de <code>Resources-[LANG].resx</code> , y el texto de registros o excepciones que se queda en un único idioma para desarrolladores.
Dominio sin cultura	El modelo expone un valor como <code>TicketStatus.Closed</code> y la interfaz lo asigna a una etiqueta localizada, de modo que la lógica de negocio y sus pruebas nunca dependen de un idioma.
Matriz de pruebas de localización	Una lista escrita de ejecuciones (neutra, traducida, con repliegue de región, clave ausente, valores formateados, cambio en tiempo de ejecución, cadenas largas, RTL, etiquetas de sistema) para que la cobertura sea una lista de comprobación y no un recuerdo.
Seudolocalización	Un conjunto de recursos de prueba cuyos valores están visiblemente alterados y alargados, como <code>[!!! Save customer record !!!]</code> , creado para sacar a la luz los recortes y las cadenas escritas en el código, no para traducir nada.
Recursos satélite	.NET compila los recursos específicos de cultura en ensamblados satélite, así que las carpetas de cultura que produce la compilación tienen que viajar con la aplicación cuando la publicas o la empaquetas.

3. Desarrollo de la lección

Un servicio que se hace cargo de todas las búsquedas

Hasta ahora cada pantalla ha acudido directamente al helper de recursos. Funciona, pero dispersa las decisiones: qué nombre base de recursos, qué hace una clave ausente, qué cultura da formato a una moneda. Un `LocalizationService` ligero reúne esas decisiones en una clase y no cambia nada más: sigue leyendo a través de `Wisej.Resources.ResourceManager`, sigue apoyándose en el repliegue de cultura normal de .NET y sigue formateando con `Wisej.Web.Application.CurrentCulture`. Lo que te aporta es un único sitio donde cambiar de opinión y un único sitio que un revisor tiene que leer.

```
public sealed class LocalizationService
{
    private readonly Wisej.Resources.ResourceManager _resources =
        new("GlobalDesk.Resources.Strings", typeof(LocalizationService).Assembly);

    public string Text(string key)
        => _resources.GetString(key) ?? $"[{key}]";

    public string Currency(decimal value)
        => value.ToString("C", Wisej.Web.Application.CurrentCulture);

    public string Date(DateTime value)
        => value.ToString("d", Wisej.Web.Application.CurrentCulture);
}
```

Dos detalles importan más que el código. Primero, `Text` nunca devuelve una cadena vacía: una clave desconocida vuelve como `[CustomerEditor.Save]`, imposible de pasar por alto en pantalla y trivial de buscar en una captura. Segundo, todos los métodos de formato leen `Application.CurrentCulture` en el momento de la llamada. La cultura es estado por sesión en Wisej.NET, así que un `CultureInfo` guardado en un campo (peor aún, en un campo estático) le pasaría en silencio a un usuario el idioma de otro. Haz que el servicio sea el único sitio donde la interfaz toca `ResourceManager`, y una búsqueda de ese tipo en toda la solución debería devolver un único resultado.

Tres familias de texto, tres propietarios distintos

El texto de producto es tuyo: etiquetas, mensajes, redacción de las validaciones, todas las frases que un traductor llegará a ver, viviendo en tus archivos `.resx` bajo claves semánticas estables. El texto de sistema de Wisej.NET es del propio framework: los botones de un `MessageBox`, las etiquetas que aportan los diálogos integrados, sobrescritas por idioma con un archivo `Resources-[LANG].resx`. El texto de diagnóstico es para desarrolladores: lo que escribes en un registro o adjuntas a una excepción. Se queda en un único idioma a propósito, porque un stack trace traducido al alemán es un stack trace que nadie puede buscar.

La misma separación se aplica dentro de la aplicación. Un ticket no está "Closed"; tiene un estado, y `TicketStatus.Closed` es el valor que expone el modelo. La interfaz, a través del servicio de localización, asigna ese valor a una clave de recurso y pide la etiqueta. El código de dominio que devuelve frases arrastra un idioma hasta tus reglas de negocio y hasta todas las pruebas que las comprueban; el código de dominio que devuelve valores mantiene estables ambas cosas mientras la interfaz gana un cuarto idioma.

Una matriz de pruebas en lugar de una comprobación al azar

Pulsar una vez el selector de idioma y ver etiquetas en alemán demuestra muy poco. Una matriz de pruebas de localización útil está escrita y se repite: la cultura neutra; al menos una cultura traducida; una cultura de idioma y región que tiene que replegarse a su recurso de idioma; una traducción ausente a propósito, para que veas el repliegue y no un hueco en blanco; valores de fecha, número, porcentaje y moneda; un cambio de cultura en tiempo de ejecución con pantallas ya abiertas; una pasada con cadenas largas; una cultura RTL con un diseño reflejado; las etiquetas de sistema de Wisej.NET en un `MessageBox`; y los mensajes de validación y de error, que son las cadenas que los equipos olvidan más a menudo.

La seudolocalización no necesita herramientas especiales. Construye un archivo de recursos de prueba cuyos valores estén visiblemente alterados y alargados, como `[!!! Save customer record !!!]`, y ejecuta la aplicación contra él. La precisión da igual; el objetivo es que los botones recortados y las etiquetas apretadas se vuelvan evidentes, y que todo lo que siga en inglés a secas durante esa ejecución sea candidato a cadena escrita en el código que hay que corregir. Una advertencia: algunas extensiones premium de Wisej.NET envuelven bibliotecas JavaScript de terceros cuya localización tiene su propio ajuste de cultura del proveedor, y la configuración de las extensiones admite marcadores de cultura y de configuración regional para los recursos de los componentes. Trata eso como una capa de integración sobre tu cultura y prueba el componente envuelto después de un cambio de idioma en lugar de dar por hecho que sus etiquetas siguen a tus `.resx`.

Qué tiene que viajar con la compilación publicada

Normalmente .NET compila los recursos específicos de cultura en ensamblados satélite, que la compilación deja en carpetas con el nombre de la cultura junto a la aplicación. `ResourceManager` los encuentra mediante la búsqueda estándar de .NET; no hay nada que cargar a mano. La lección de despliegue es, por tanto, corta: no descartes las carpetas de salida específicas de cultura cuando publiques o empaquetes la aplicación.

Ese es el tipo de error que un script de despliegue comete en silencio. Un paso de copia que se lleva solo el ensamblado principal, o una imagen construida a partir de una lista de archivos seleccionada a mano, produce una aplicación que se ve perfecta en tu máquina y se repliega al inglés en todas las demás. Añade una línea a la lista de comprobación del proyecto final: después de publicar, abre la carpeta de salida y confirma que las carpetas de cultura están ahí.

Terminar GlobalDesk

El proyecto final no es una aplicación grande; es la prueba de que sabes localizar una de principio a fin en lugar de traducir una etiqueta. GlobalDesk necesita un `.resx` neutro con un conjunto significativo de claves semánticas, al menos un archivo de recursos traducido, una pantalla localizada en el diseñador, el `LocalizationService`, un selector de idioma que establezca `Application.CurrentCulture`, la visualización de fechas, números y moneda según la cultura, una demostración de recursos de sistema de Wisej.NET, una demostración de RTL, las notas del flujo de trabajo de traducción y la lista de comprobación de QA.

Ensayo la demostración en un orden fijo, porque el orden es el argumento: arranca con la cultura automática del navegador, cambia al alemán, muestra cómo las etiquetas traducidas y los valores formateados cambian a la vez, provoca un `MessageBox` para que aparezca una etiqueta de sistema, enseña una traducción ausente a propósito replegándose, cambia a la cultura RTL y observa el diseño reflejado, y después enseña las notas del flujo de trabajo y la lista de comprobación. Si todos los pasos funcionan sin que edites nada, la localización está terminada.



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:

- ✓ Crear un pequeño servicio de localización alrededor de ResourceManager.
- ✓ Separar el texto de producto, el texto de sistema y el texto de diagnóstico.
- ✓ Probar el repliegue, el formato, RTL y el cambio de cultura en tiempo de ejecución.
- ✓ Entender a alto nivel el despliegue de recursos satélite.
- ✓ Completar el proyecto final de localización de GlobalDesk.

Guía del laboratorio / examen

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



Laboratorio práctico

Termina el proyecto final de GlobalDesk. Envuelve toda búsqueda de recursos en un `LocalizationService` con métodos `Text`, `Date`, `Number` y `Currency`, dale una política de claves ausentes que muestre la clave entre corchetes en desarrollo y la registre en producción, y haz que ese servicio sea el único sitio donde la interfaz toca `ResourceManager`. Mantén el dominio al margen: el modelo del ticket expone un valor `TicketStatus` y la interfaz lo asigna a una etiqueta localizada. Después ejecuta la matriz de pruebas de localización sobre la aplicación terminada, cubriendo la cultura neutra, el alemán, una cultura de idioma y región que se repliega, una clave ausente a propósito, fechas e importes formateados, un cambio en tiempo de ejecución, una pasada con cadenas largas y la cultura RTL, usando un conjunto de recursosseudolocalizado para sacar a la luz todo lo que siga escrito en el código. Confirma que las carpetas de cultura sobreviven a una publicación y entrega la lista de comprobación de QA como entregable del proyecto final.

Punto de partida

GlobalDesk tal y como lo dejaste tras el Módulo 6: un dashboard y un editor de clientes localizado en el diseñador, `Resources/Strings.resx` con sus archivos compañeros en alemán e italiano, el helper que los lee, un selector de idioma que establece `Application.CurrentCulture`, un manejador de `CultureChanged` que vuelve a aplicar el texto y recrea el editor, soporte RTL y tus notas del flujo de trabajo. Todas las capas funcionan, pero las búsquedas de recursos siguen ocurriendo en varios sitios, unos cuantos mensajes en tiempo de ejecución siguen siendo literales en inglés dentro del código, y nadie ha escrito cómo se prueba la aplicación.

Enfoque sugerido

- ✓ Escribe primero `LocalizationService`: un único `Wisej.Resources.ResourceManager` sobre tu nombre base `Strings` y el ensamblado, un método `Text(key)` que devuelva un marcador `[Key]` entre corchetes cuando la búsqueda no encuentre nada, y métodos `Date`, `Number` y `Currency` que den formato con `Application.CurrentCulture` leída en el momento de la llamada. Dale al camino de la clave ausente un único interruptor: marcador durante el desarrollo, clave registrada más texto neutro en producción.
- ✓ Haz que sea la única puerta. Sustituye por una llamada al servicio todas las llamadas directas a recursos que queden y todos los literales en inglés que pueda ver el usuario; después busca `ResourceManager` en toda la solución y confirma que el servicio es el único resultado. Deja como está el texto de tus registros y excepciones y añade un comentario que diga que es diagnóstico a propósito.
- ✓ Saca la cultura del dominio: haz que el modelo del ticket devuelva un valor `TicketStatus`, añade en la interfaz o en el servicio una pequeña correspondencia entre ese valor y una clave de recurso como `TicketStatus.Closed`, y comprueba que ya no queda ninguna etiqueta en inglés en el modelo ni en sus pruebas.

- ✓ Construye el conjunto de recursos seudolocalizado (las mismas claves con los valores alargados y entre corchetes, por ejemplo `[!!! Save customer record !!!]`), ejecuta la aplicación contra él y lista todas las etiquetas que se quedaron en inglés y todos los controles cuyo texto se recortó. Corrige las dos listas.
- ✓ Escribe la lista de comprobación de QA como una tabla y ejecútala: cultura neutra, alemán, una cultura de idioma y región que se repliega a su recurso de idioma, una clave ausente a propósito, una fecha, un número y un importe monetario, un cambio en tiempo de ejecución con el editor abierto, la pasada con cadenas largas, la cultura RTL y una etiqueta de sistema en un `MessageBox` . Después publica la aplicación y confirma que las carpetas de cultura y sus ensamblados de recursos están en la salida.

El proyecto final se revisa como una aplicación única en funcionamiento, no como diez ejercicios sueltos: un archivo de recursos neutro con un conjunto significativo de claves semánticas, al menos un archivo de recursos traducido, una pantalla localizada en el diseñador, el `LocalizationService` , un selector de idioma que cambia `Application.CurrentCulture` , la visualización de fechas, números y moneda según la cultura, una demostración de recursos de sistema de Wisej.NET, una demostración de RTL, las notas del flujo de trabajo de traducción y la lista de comprobación de QA. Ensayo la demostración en un orden fijo: arranca con la cultura automática del navegador, cambia al alemán, muestra cómo las etiquetas traducidas y los valores formateados cambian a la vez, provoca un `MessageBox` para que aparezca una etiqueta de sistema, enseña una traducción ausente a propósito replegándose, cambia a la cultura RTL y después muestra las notas del flujo de trabajo. Recuerda que la cultura es estado de sesión, así que si algo de lo que has escrito guarda en caché un `CultureInfo` fuera de la sesión, pasará tu propia demostración y fallará en cuanto un compañero abra la aplicación a tu lado.

Entregables requeridos

- ✓ `LocalizationService` con métodos `Text`, `Date`, `Number` y `Currency` y una política documentada para las claves ausentes
- ✓ Código de dominio que devuelve un valor `TicketStatus` que la interfaz asigna a una etiqueta localizada
- ✓ Conjunto de recursos seudolocalizado usado para encontrar todas las cadenas que sigan escritas en el código
- ✓ Matriz de pruebas de localización completada, que cubre el repliegue, el formato, el cambio en tiempo de ejecución y RTL
- ✓ Salida publicada comprobada en busca de sus carpetas de cultura, más la lista de comprobación de QA de localización de GlobalDesk

Preguntas de repaso

- ✓ Un revisor abre GlobalDesk en francés, una cultura que nunca tradujiste, y ve etiquetas en inglés con fechas francesas correctamente formateadas. Explica qué mecanismo produjo cada mitad de esa pantalla y di si hay algo roto.
- ✓ Tu ejecución seudolocalizada muestra una etiqueta que sigue poniendo "Save" en inglés a secas. Enumera los tres sitios de los que podría venir ese texto y describe cómo los distinguirías sin

adivinar.

- ✓ Un compañero propone guardar en caché el `CultureInfo` resuelto en un campo estático de `LocalizationService` "porque buscarlo en cada llamada es un derroche". Describe qué experimentaría un segundo usuario con la sesión iniciada y qué propondrías tú en su lugar.



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.