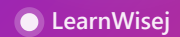


Idiomas de derecha a izquierda y diseños a prueba de localización

Guía de la lección · Guía del laboratorio / examen



Guía de la lección

El soporte de derecha a izquierda en Wisej.NET: `RightToLeft` con `Inherit`, `No` y `Yes`, `RightToLeftLayout` reflejando los hijos de un contenedor, el ajuste de aplicación `rightToLeft`, diseños que sobreviven a la expansión del texto, el estado `rightToLeft` del tema y los límites documentados en `DataGridView`, `ListView` y `PropertyGrid`. 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

El soporte de derecha a izquierda en Wisej.NET: `RightToLeft` con `Inherit`, `No` y `Yes`, `RightToLeftLayout` reflejando los hijos de un contenedor, el ajuste de aplicación `rightToLeft`, diseños que sobreviven a la expansión del texto, el estado `rightToLeft` del tema y los límites documentados en `DataGridView`, `ListView` y `PropertyGrid`.

1. El problema en producción

GlobalDesk ya puede cambiar de idioma mientras se ejecuta: el selector establece

`Application.CurrentCulture`, el manejador de `Application.CultureChanged` vuelve a aplicar todas las etiquetas respaldadas por recursos y el panel de vista previa reformatea sus ejemplos. Entonces un tester abre el dashboard en árabe. Todas las palabras están traducidas y todas están en el sitio equivocado: las etiquetas se pegan al borde izquierdo de sus campos, la columna de navegación queda donde la mirada de un lector árabe llega en último lugar, y las etiquetas alemanas largas del módulo anterior ya empujan un botón fuera de un panel de ancho fijo.

La traducción era la mitad fácil. Una pantalla localizada también tiene una dirección de lectura y un ancho que no controla, y ambas son decisiones de diseño que tomas mientras construyes el formulario, no reparaciones que haces después.

2. Conceptos clave

Concepto	Por qué importa
RightToLeft	Una propiedad de control con los valores <code>Inherit</code> , <code>No</code> y <code>Yes</code> que indica a un control que adapte su propio texto y la dirección de sus elementos internos.
RightToLeft.Inherit	La opción normal para un control hijo: sigue a su padre, así que toda la pantalla cambia de dirección a partir de una única decisión en lo alto.
RightToLeftLayout	Una propiedad distinta que refleja la posición horizontal de los hijos directos de un contenedor cuando RTL está activo, de modo que la disposición visual se invierte mientras tu diseño lógico se mantiene tal y como lo hiciste.
El ajuste de aplicación rightToLeft	Una entrada de <code>Default.json</code> ; con el valor <code>"auto"</code> hace que Wisej.NET derive el comportamiento RTL de la cultura de la sesión en lugar de comprobaciones de nombres de cultura repartidas por tus pantallas.
Application.RightToLeft	La misma decisión expresada desde el código, para las sesiones que hay que cambiar deliberadamente y no por detección de cultura.
Expansión del texto	Una etiqueta traducida rara vez mide lo mismo que la inglesa, y por eso las posiciones fijas en píxeles y los anchos de panel medidos a mano son lo primero que se rompe en una compilación localizada.
Límites RTL documentados	Wisej.NET documenta que <code>DataGridView</code> , <code>ListView</code> y <code>PropertyGrid</code> no admiten del todo RTL, así que un plan de pruebas tiene que nombrarlos en lugar de dar por hecho que se reflejan.

3. Desarrollo de la lección

Dos propiedades, dos trabajos

El primer error que comete casi todo el mundo es tratar la escritura de derecha a izquierda como un ajuste de alineación del texto. El árabe, el hebreo y los demás idiomas de derecha a izquierda necesitan que el texto corra en sentido contrario y, normalmente, que la disposición de la pantalla se refleje. Wisej.NET separa esos dos trabajos en dos propiedades, y saber cuál necesitas es casi todo este módulo.

`RightToLeft` pertenece a un solo control y cambia lo que ocurre dentro de él: la dirección de su texto y de los elementos que dibuja por su cuenta. Admite `Inherit` , `No` o `Yes` . Déjala en `Inherit` casi en todas partes, porque así un hijo sigue a su padre y todo el dashboard de GlobalDesk responde a una única decisión tomada en lo alto. Poner `Yes` en controles individuales te deja una pantalla donde la dirección se configura en cincuenta sitios y nadie puede decir cuál está mal.

`RightToLeftLayout` es el otro trabajo. Refleja la posición horizontal de los hijos directos de un contenedor cuando RTL está activo, así que sigues diseñando el editor de clientes en el orden que te resulta lógico mientras Wisej.NET invierte la disposición visual. Fíjate en las palabras "hijos directos": activarla en el dashboard refleja los paneles que contiene el dashboard, y cada panel decide por sí mismo

si sus propios controles se reflejan. Por eso el laboratorio la activa en el dashboard y en el editor de clientes.

Deja que decida el ajuste de la aplicación

`Default.json` lleva un ajuste `rightToLeft` junto al ajuste `culture` que configuraste en el módulo anterior. Con el valor `"auto"`, Wisej.NET deriva el comportamiento RTL de la cultura con la que se ejecuta la sesión, así que basta con añadir el árabe al selector para que la dirección siga al idioma.

`Application.RightToLeft` expone la misma decisión al código cuando hay que gobernar una sesión deliberadamente.

La alternativa es el patrón que hay que evitar: una comprobación del nombre de cultura dentro de cada pantalla, después una segunda lista de prefijos cuando llega el hebreo, y después una pantalla que alguien olvidó. Un único ajuste de la aplicación sustituye a todo eso, y es lo que permite que tus controles se queden en `Inherit`. La dirección se convierte en estado de sesión, exactamente igual que

`Application.CurrentCulture`: que un tester cambie al árabe nunca debe reflejar la pantalla de un compañero.

Diseñar un diseño que pueda reflejarse

El reflejo solo queda bien si el diseño era flexible desde el principio. Un control en una `Location` medida a mano dentro de un panel de tamaño fijo se refleja a una posición que nunca se diseñó, y una etiqueta un 40 % más larga en alemán simplemente se sale por su borde. Las herramientas son las de diseño habituales de Wisej.NET: docking para que una región se quede con un borde, anchoring para que un control crezca con su contenedor, `AutoSize` donde una etiqueta deba poder ser tan larga como su traducción, un contenedor de diseño donde una fila de botones deba recolocarse, y tamaños mínimos razonables para que nada se venga abajo.

Hay una excepción deliberada, y el laboratorio te pide que escribas el comentario que la explica. Algunas cadenas no son prosa: un código de cliente, un número de serie, el contenido de un editor de código. Esas se leen de izquierda a derecha en todas las culturas, e invertirlas convierte un identificador correcto en uno confuso. Para eso está `RightToLeft.No`: usado con moderación y siempre con un motivo al lado.

```
private void ConfigureDirection()
{
    // Default.json carries rightToLeft "auto", so Application.RightToLeft
    // already follows the session culture. Children stay on Inherit.
    this.RightToLeft = RightToLeft.Inherit;
    this.RightToLeftLayout = true;

    // The customer code is a technical identifier, not prose:
    // it must keep reading left to right in every culture.
    txtCustomerCode.RightToLeft = RightToLeft.No;
}
```

Probar la expansión y los controles reflejados

Ninguno de los dos problemas se ve en el código, así que el módulo termina en una lista de pruebas y no en una funcionalidad. Pasa cada pantalla localizada por al menos cinco casos:

- Texto corto en el idioma neutro, que es el estado en el que se diseñó todo.
- Un idioma con traducciones notablemente más largas, como el alemán.
- Una ventana estrecha del navegador, que es donde el anchoring y los tamaños mínimos demuestran su valor.
- Una cultura RTL como el árabe, comprobando la disposición y no solo la alineación.
- Contenido mixto, como una etiqueta en árabe junto a un código de producto latino.

Un archivo de recursos seudolocalizado hace honestos los casos de expansión. Copia las claves neutras y después alarga y pon entre corchetes cada valor, de modo que una etiqueta pase a ser algo como `[[Save changes ~~~~]]`. Todo lo que siga leyéndose como inglés a secas nunca fue una clave de recurso, y todo lo que se recorte te dice qué control necesita anclaje. Los temas de Wisej.NET también entienden un estado `rightToLeft`, así que los iconos y los estilos de un solo lado pueden adaptarse sin un segundo tema; las reglas de tema personalizadas y los widgets personalizados son las partes que todavía pueden necesitar un reflejo escrito a mano.

Los controles con límites documentados

Por último, la parte honesta. Wisej.NET documenta que `DataGridView`, `ListView` y `PropertyGrid` no admiten del todo la escritura de derecha a izquierda. GlobalDesk muestra sus contactos en un `DataGridView`, así que el laboratorio te pide que anotes lo que hace realmente esa cuadrícula en árabe y no lo que esperabas. El orden de las columnas, la alineación de las cabeceras y el lado de la barra de desplazamiento son las cosas que hay que mirar.

Esto no es un motivo para evitar esos controles; es un motivo para nombrarlos en el plan de pruebas y verificarlos en la versión y la configuración exactas que publicas. Un diseño localizado nunca debería recortar una etiqueta traducida, empujar un control fuera del área visible, depender de la longitud de las palabras inglesas, incrustar una flecha direccional en una imagen sin variante RTL ni forzar un identificador a la dirección de lectura equivocada: cada una de esas cosas sale más barata de arreglar en la primera pantalla que en la centésima.



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:

- ✓ Usar correctamente `RightToLeft` y `RightToLeftLayout` de Wisej.NET.
- ✓ Explicar el ajuste de aplicación `rightToLeft="auto"`.

- ✓ Construir un diseño que se refleje adecuadamente para las culturas RTL.
- ✓ Probar la expansión del texto y los controles reflejados.
- ✓ Identificar los controles de Wisej cuyo soporte RTL tiene limitaciones documentadas.

Guía del laboratorio / examen

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



Laboratorio práctico

Haz que GlobalDesk sobreviva a una cultura de derecha a izquierda. Añade el árabe al selector de idioma y pon `rightToLeft` en auto en `Default.json`, deja los controles hijos en `RightToLeft.Inherit` y activa `RightToLeftLayout` en el dashboard y en el editor de clientes para que sus hijos se reflejen. Mantén exactamente un control de izquierda a derecha, el campo del código de cliente, y explica en un comentario por qué ese identificador no debe invertirse. Reconstruye con docking, anchoring y un contenedor de diseño la parte del diseño con posiciones fijas para que una traducción larga no pueda recortarse, añade un archivo de recursos seudolocalizado cuyos valores estén alargados y entre corchetes, y anota lo que hace realmente la cuadrícula `DataGridView` de contactos bajo RTL en lugar de dar por supuesto que se refleja. Termina con una lista de cinco casos de prueba de diseño que puedas repetir en cualquier pantalla.

Punto de partida

GlobalDesk después del Módulo 4: el dashboard, el editor de clientes y la cuadrícula de contactos leen todas sus etiquetas de los recursos, el selector de idioma ofrece inglés, alemán y una cultura de idioma y región, y un manejador de `Application.CultureChanged` ya llama a `ApplyTextResources` y a `UpdateCulturePreview`. Todas las pantallas siguen construidas de izquierda a derecha, y el panel de navegación del dashboard sigue usando los valores fijos de `Location` y `Size` del Módulo 1.

Enfoque sugerido

- ✓ Añade una entrada de árabe al selector de idioma, pon `rightToLeft` en `"auto"` junto al ajuste `culture` que ya existe en `Default.json`, y después ejecuta la aplicación y cambia al árabe antes de tocar nada más, para ver exactamente qué hace la detección automática por sí sola.
- ✓ Pon `RightToLeftLayout` en `true` en el dashboard y en el editor de clientes, y comprueba que todos los controles hijos siguen en `RightToLeft.Inherit`: resiste la tentación de poner `Yes` en controles individuales para "arreglar" lo que debería decidir el contenedor.
- ✓ Pon `RightToLeft.No` solo en el campo del código de cliente y escribe el comentario que explique por qué un identificador técnico debe mantener su dirección de lectura mientras el texto que lo rodea se invierte.
- ✓ Sustituye las posiciones absolutas del panel de navegación por docking y anchoring, coloca la fila de botones del editor en un contenedor de diseño para que pueda recolocarse, y da `AutoSize` a las etiquetas y tamaños mínimos razonables a los paneles.
- ✓ Añade un archivo de recursos seudolocalizado cuyos valores estén alargados y entre corchetes, pasa por él los cinco casos de prueba, anota lo que hace realmente el `DataGridView` de contactos en árabe y convierte toda la pasada en una lista de pruebas reutilizable.

Entregables requeridos

- ✓ Entrada de árabe en el selector de idioma con `rightToLeft` en auto en `Default.json`

- ✓ Dashboard y editor de clientes usando `RightToLeftLayout` con sus hijos en `Inherit`
- ✓ Un control deliberadamente de izquierda a derecha con un comentario que explique la excepción
- ✓ Diseño reconstruido con docking, anchoring y un contenedor de diseño para que las traducciones largas no puedan recortarse
- ✓ Lista de pruebas RTL que cubra texto corto, texto largo, una ventana estrecha, una cultura RTL y la limitación del `DataGridView`

Preguntas de repaso

- ✓ Un compañero pone `RightToLeft.Yes` en veinte controles y la pantalla en árabe se ve correcta, pero la pantalla en alemán no ha cambiado. ¿Qué ha configurado en realidad y qué única propiedad habría hecho el trabajo para todo el contenedor?
- ✓ Tu pasada seudolocalizada muestra tres etiquetas todavía en inglés a secas. Nada se recortó y nada lanzó una excepción. ¿Qué te dice eso sobre esas tres etiquetas y en qué módulo anterior quedó un hueco de trabajo?
- ✓ La cuadrícula de contactos no se refleja igual que el resto del dashboard. De tus opciones (aceptarlo y documentarlo, sustituir el control o escribir el reflejo a mano), ¿cuál puedes defender ante un revisor y qué tendrías que verificar antes de prometer cualquiera de ellas?



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.