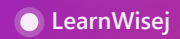


El modelo de imágenes de Wisej.NET: Image, ImageSource e ImageList

Guía de la lección · Guía del laboratorio / examen



Guía de la lección

Las dos canalizaciones de imagen: una propiedad Image que contiene un System.Drawing.Image que el servidor envía al navegador como PNG, frente a una cadena ImageSource o IconSource que el cliente resuelve como una URL, un nombre de tema, una ruta resource.wx o un elemento de un pack de iconos, más el contrato común IImage y la ImageList que permite que muchos controles compartan un mismo conjunto de imágenes direccionadas por clave. Qué significa esto para un desarrollador que conoce C# básico y ya ha trabajado algo con los controles y el diseñador de Wisej.NET — y cómo leer este módulo.



Enfoque de producción

Las dos canalizaciones de imagen: una propiedad Image que contiene un System.Drawing.Image que el servidor envía al navegador como PNG, frente a una cadena ImageSource o IconSource que el cliente resuelve como una URL, un nombre de tema, una ruta resource.wx o un elemento de un pack de iconos, más el contrato común IImage y la ImageList que permite que muchos controles compartan un mismo conjunto de imágenes direccionadas por clave.

1. El problema en producción

Wisej.NET ofrece a propósito más de una forma de poner un gráfico sobre un control, y todas funcionan. El problema empieza cuando un equipo las usa todas por accidente: una barra de herramientas en la que un botón lee un archivo del disco del servidor, el siguiente apunta a un nombre de tema y un tercero perdió su icono en silencio porque se asignaron dos propiedades al mismo control. Seis meses después, nadie sabe explicar de dónde sale realmente una imagen determinada ni por qué desapareció tras el despliegue.

2. Conceptos clave

Concepto	Por qué importa
Propiedad Image	Un objeto <code>System.Drawing.Image</code> que vive en la memoria del servidor; Wisej.NET lo envía al navegador como datos PNG, así que la imagen viaja como bytes que tu código ya decodificó.
Cadena ImageSource	Una cadena que identifica un recurso para que lo resuelva el cliente: una URL relativa, una URL absoluta, una imagen de tema con nombre, una ruta <code>resource.wx</code> o un elemento dentro del ensamblado de un pack de iconos.
IconSource	La misma idea de cadena en los controles que tienen un hueco de icono propio junto a su imagen principal, así que revisa la API del control que estés usando en lugar de suponer que todos los controles llaman igual a la propiedad.
Contrato IImage	La familia común de imagen que exponen los controles capaces de mostrar imágenes — <code>Image</code> , <code>ImageAlign</code> , <code>ImageIndex</code> , <code>ImageKey</code> , <code>ImageList</code> e <code>ImageSource</code> —, que permite que un único método auxiliar decore de forma uniforme un Button, una Label o cualquier otra cosa.
ImageList	Una colección reutilizable de objetos de imagen a la que pueden apuntar varios controles, direccionada por posición o por nombre, y capaz de fijar un tamaño de imagen común para los controles que la usan.
ImageKey frente a ImageIndex	Una clave como <code>"delete"</code> transmite significado y sobrevive a los reordenamientos; el índice <code>3</code> pasa a ser en silencio la imagen equivocada el día en que alguien inserta una imagen por encima.
Asignación mutuamente excluyente	Todas estas propiedades alimentan un único hueco visual, así que asignar una puede restablecer las demás, y por eso el icono equivocado suele ser una segunda asignación y no un archivo roto.

3. Desarrollo de la lección

Objeto o cadena: la primera pregunta que hay que responder

Antes de escribir nada, decide si el servidor es dueño de un *objeto* de imagen o si debe ser el navegador quien resuelva un *identificador* de imagen. Esas son las dos canalizaciones, y casi toda la confusión posterior de este curso viene de mezclarlas. Cuando asignas `Image`, le entregas a Wisej.NET un `System.Drawing.Image`: datos de píxeles reales ya decodificados que están en la memoria de tu sesión y que el framework envía después al navegador como PNG. Esa es la vía correcta cuando tu C# ha producido o recibido de verdad un objeto de imagen: un gráfico que renderizaste, una miniatura que redimensionaste, una foto que llegó desde un servicio como un stream.

Cuando asignas `ImageSource` o `IconSource`, le entregas a Wisej.NET una cadena. No se decodifica nada en el servidor. La cadena nombra algo que el cliente puede obtener y cachear por su cuenta:

`"Images/open.png"` relativo a la aplicación, una dirección absoluta, el nombre de una imagen de tema como `"icon-save"` o una ruta `resource.wx` dentro de un ensamblado. Como el recurso conserva su

identidad, un SVG sigue siendo un SVG hasta llegar al navegador, y por eso los iconos de interfaz se mantienen nítidos en cualquier tamaño y después se pueden recolorear. En IconDesk verás las dos cosas: `btnOpen` recibe un objeto de imagen que cargó el servidor, `btnSave` recibe un nombre de tema como cadena, y la etiqueta de diagnóstico tiene que distinguirlos.

Una familia de propiedades, llamada IImage

Wisej.NET no le da a cada control su propio vocabulario para las imágenes. Los controles capaces de mostrar imágenes comparten un contrato llamado `IImage` que reúne `Image`, `ImageAlign`, `ImageIndex`, `ImageKey`, `ImageList` e `ImageSource` en una sola familia. Reconocer la familia le resulta más valioso a quien empieza que convertir a ella: cuando te encuentras con un control nuevo, ya sabes qué propiedades buscar y cuáles cuatro de ellas compiten por el mismo hueco.

La conversión sigue siendo útil para el código de infraestructura. Un método que acepta un `Control`, comprueba `is IImage` y después establece `ImageSource` puede aplicar el icono de guardar del proyecto a un Button, una Label o un elemento de menú sin una sentencia switch. La etiqueta de diagnóstico de IconDesk del laboratorio hace lo contrario: lee cada botón a través de `IImage`, comprueba cuál de las cuatro propiedades está asignada e imprime el mecanismo que hay detrás del icono en lugar de adivinarlo por la imagen.

Elegir el mecanismo para el recurso que tienes delante

Cuatro casos cubren casi todo. Un archivo desplegado junto a la aplicación es un origen de imagen con una URL relativa. Una imagen alojada en otro sitio es un origen de imagen con una URL absoluta. Un icono que el tema activo ya define es un origen de imagen con el nombre que le da el tema, lo que significa que el cliente lo toma de la caché del tema que ha cargado de todos modos. Una imagen que tu código creó, transformó o recibió como objeto .NET es un `Image`.

El quinto caso es el compartido, y es para lo que existe `ImageList`. Una `ImageList` contiene una colección de objetos de imagen; los controles referencian una entrada mediante el `ImageIndex` numérico o la `ImageKey` de texto. Dos propiedades de esa indirección importan en una aplicación real: muchos controles pueden compartir una misma colección, y la imagen almacenada bajo una clave se puede sustituir sin tocar ni un solo control que se refiera a esa clave. La lista también puede fijar un tamaño de imagen para los controles que la usan, algo práctico cuando, si no, sería el tema el que decidiría por ti el tamaño del icono.

```
var icons = new ImageList();
icons.Images.Add("print", Image.FromFile("Images/print.png"));
icons.Images.Add("delete", Image.FromFile("Images/delete.png"));

btnPrint.ImageList = icons;
btnPrint.ImageKey = "print";
btnDelete.ImageList = icons;
btnDelete.ImageKey = "delete";

// btnSave takes its icon from a string instead - no ImageList involved.
btnSave.ImageSource = "icon-save";
```

Fíjate en lo que las primeras líneas *no* están haciendo. `Image.FromFile` lee un archivo del *servidor*, nunca de la máquina del usuario, así que una ruta absoluta que funcionaba en tu portátil es un defecto de despliegue esperando a ocurrir. Una cadena de origen de imagen, en cambio, se direcciona igual que el navegador direcciona todo lo demás.

Por qué el icono que asignas no siempre es el icono que ves

Un control tiene un único hueco visual para su imagen, y `Image`, `ImageSource`, `ImageIndex` e `ImageKey` compiten todas por llenarlo. Wisej.NET lo resuelve haciéndolas mutuamente excluyentes: asignar una puede restablecer las demás, de modo que la configuración nunca es ambigua. Si `btnDelete` está mostrando tan tranquilo la entrada de la lista que hay bajo `ImageKey = "delete"` y una línea posterior establece `ImageSource`, la cadena pasa a ser la representación activa y la clave deja de importar.

Esa única regla explica la mayoría de los informes de error del tipo «se muestra el icono equivocado». El diseñador estableció una propiedad, el constructor del formulario estableció otra, y un tema o un método auxiliar estableció una tercera. Cuando la imagen es la incorrecta, no empieces abriendo el archivo: inspecciona todas las propiedades de imagen del control y localiza la última asignación que se ejecutó.

- Una sola propiedad de imagen por control, elegida de forma deliberada.
- Claves, no índices, siempre que la identidad tenga significado.
- Una alternativa textual para todo icono que transmita significado por sí solo.



Nombra primero el mecanismo

Antes de asignar cualquier imagen, di qué mecanismo estás usando: un objeto `System.Drawing.Image` que el servidor renderiza como PNG, una cadena de origen de imagen que resuelve el navegador, una imagen de tema con nombre, un recurso incrustado servido por `resource.wx` o una fuente de iconos dibujada como HTML. Asigna una sola propiedad de imagen por control para que las demás no puedan restablecerla en silencio, mantén la cadena de origen con significado en lugar de una ruta mágica pegada, prefiere SVG para los iconos de interfaz para que se mantengan nítidos en cualquier tamaño y escala, y dale una alternativa textual a todo icono que transmita significado.



Objetivos de aprendizaje

Al terminar este módulo serás capaz de:

- ✓ Explicar la diferencia entre un valor `System.Drawing.Image` y una cadena de imagen como `ImageSource` o `IconSource`.
- ✓ Usar el modelo común `Image` para reconocer las propiedades `Image`, `ImageSource`, `ImageList`, `ImageIndex` e `ImageKey`.
- ✓ Elegir el mecanismo de imagen correcto para un bitmap local, una URL, una imagen de tema o un icono reutilizable.
- ✓ Entender por qué asignar una de varias propiedades de imagen mutuamente excluyentes puede restablecer las demás.

Guía del laboratorio / examen

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



Laboratorio práctico

Crea el proyecto IconDesk y dale un panel de comandos con los botones Open, Save, Print y Delete, cada uno con su icono a través de un mecanismo distinto. Carga btnOpen desde un `System.Drawing.Image` del lado del servidor, asigna una imagen de tema con nombre a `btnSave.ImageSource` y apunta btnPrint y btnDelete a una misma `ImageList` compartida mediante `ImageKey`. Añade una etiqueta de diagnóstico que inspeccione cada botón a través de `IImage` e indique qué mecanismo está realmente en juego, después cambia la imagen almacenada bajo una clave de la `ImageList` y comprueba que todos los controles que usan esa clave siguen el cambio.

Punto de partida

Este es el primer módulo, así que partes de cero: crea una nueva aplicación Wisej.NET 4 llamada `IconDesk` y añade una página para el panel de comandos. Coloca junto a la aplicación una carpeta pequeña de imágenes —basta con un gráfico de abrir, guardar, imprimir y eliminar— y comprueba que el proyecto vacío compila y se abre en el navegador antes de asignar un solo icono.

Enfoque sugerido

- ✓ Coloca primero los cuatro botones solo con texto, para que el panel se lea antes de que exista ninguna imagen.
- ✓ Dale a `btnOpen` un `System.Drawing.Image` que cargue el servidor, y a `btnSave` el nombre de una imagen de tema en `ImageSource`; di en voz alta qué canalización usa cada uno.
- ✓ Construye una `ImageList`, añade las imágenes de imprimir y eliminar bajo claves con significado y apunta después ambos botones a la lista mediante `ImageKey` en lugar de `ImageIndex`.
- ✓ Escribe la etiqueta de diagnóstico: recorre los controles del panel, comprueba en cada uno `IImage` e indica cuál de `Image`, `ImageSource`, `ImageList` más `ImageKey` está realmente asignado.
- ✓ Sustituye la imagen almacenada bajo la clave delete, recarga la página y establece después a propósito una segunda propiedad de imagen en un botón para poder anotar qué hizo la primera asignación.

Entregables requeridos

- ✓ Panel de comandos de IconDesk con los botones Open, Save, Print y Delete
- ✓ btnOpen con un `System.Drawing.Image` cargado en el servidor y btnSave con una imagen de tema con nombre en `ImageSource`
- ✓ btnPrint y btnDelete compartiendo una misma `ImageList` mediante `ImageKey` en lugar de `ImageIndex`

- ✓ Etiqueta de diagnóstico que lee cada botón a través de `Image` y nombra el mecanismo que hay detrás de su icono
- ✓ Nota de laboratorio que recoja qué ocurrió al sustituir una clave de la `ImageList` y al asignar dos propiedades de imagen en el mismo botón

Preguntas de repaso

- ✓ Tu etiqueta de diagnóstico dice que `btnDelete` usa un origen de imagen, pero tú solo estableciste su `ImageKey`. ¿Qué se ha tenido que ejecutar y dónde lo buscarías?
- ✓ Tanto imprimir como eliminar cambiaron al sustituir una entrada de la `ImageList`. ¿Seguiría siendo así si los botones usaran `ImageIndex`, y qué pasaría si se insertara una quinta imagen al principio de la lista?
- ✓ El botón Open lee su archivo en el servidor. ¿Qué parte de ese montaje se rompe cuando la aplicación se despliega en otra máquina, y qué parte no se rompería si esa misma imagen fuera una cadena de origen de imagen?



Nombra primero el mecanismo

Antes de asignar cualquier imagen, di qué mecanismo estás usando: un objeto `System.Drawing.Image` que el servidor renderiza como PNG, una cadena de origen de imagen que resuelve el navegador, una imagen de tema con nombre, un recurso incrustado servido por `resource.wx` o una fuente de iconos dibujada como HTML. Asigna una sola propiedad de imagen por control para que las demás no puedan restablecerla en silencio, mantén la cadena de origen con significado en lugar de una ruta mágica pegada, prefiere SVG para los iconos de interfaz para que se mantengan nítidos en cualquier tamaño y escala, y dale una alternativa textual a todo icono que transmita significado.