

Bitmaps, SVG y PictureBox: formatos, tamaño y carga

Guía de la lección · Guía del laboratorio / examen

● LearnWisej

Guía de la lección

Dos listas de formatos porque hay dos canalizaciones: bitmap, metafile, icon, JPEG, GIF y PNG a través de System.Drawing y PictureBox.Image, frente a los jpg, jpeg, png, gif, svg y bmp que resource.wx y las imágenes de tema aceptan como cadenas, junto con los modos de tamaño de PictureBox y los puntos de entrada síncrono Load y asíncrono LoadAsync. 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

Dos listas de formatos porque hay dos canalizaciones: bitmap, metafile, icon, JPEG, GIF y PNG a través de System.Drawing y PictureBox.Image, frente a los jpg, jpeg, png, gif, svg y bmp que resource.wx y las imágenes de tema aceptan como cadenas, junto con los modos de tamaño de PictureBox y los puntos de entrada síncrono Load y asíncrono LoadAsync.

1. El problema en producción

«¿Qué formatos de imagen admite Wisej.NET?» es la pregunta que hace todo equipo en la primera semana, y la respuesta honesta depende de qué vía de la API tome la imagen. Si la contestas con una lista vaga, acabas con un logotipo estirado en un portátil de alta densidad, con una página que se congela mientras se descarga la foto de un producto desde un servidor lento y con un informe de error porque un archivo `.ico` no se resuelve a través de una cadena de recurso. Este módulo sustituye la lista vaga por dos listas precisas, cada una asociada a la propiedad de control que la consume.

2. Conceptos clave

Concepto	Por qué importa
Canalización de System.Drawing	La propiedad <code>Image</code> contiene un <code>System.Drawing.Image</code> en la memoria del servidor; Wisej.NET lo envía al navegador como datos PNG. Sus orígenes documentados son archivos bitmap, metafile, icon, JPEG, GIF y PNG.
Canalización de orígenes de imagen	Una cadena que el cliente resuelve por su cuenta: una URL relativa o absoluta, una imagen de tema con nombre o una ruta <code>resource.wx</code> . Las extensiones de imagen documentadas para <code>resource.wx</code> son .jpg, .jpeg, .png, .gif, .svg y .bmp.
PictureBox.Image	El punto de entrada para los píxeles que tu código C# ya tiene o acaba de generar, como un gráfico dibujado con <code>Graphics</code> .
PictureBox.ImageSource	El punto de entrada para una imagen que el navegador puede obtener por sí mismo. Viaja una cadena corta en lugar de los bytes de la imagen, y por eso SVG pertenece aquí.
PictureBox.SizeMode	Los seis valores <code>Normal</code> , <code>AutoSize</code> , <code>CenterImage</code> , <code>StretchImage</code> , <code>Zoom</code> y <code>Cover</code> , que deciden si una imagen se recorta, se centra, se deforma, se ajusta o se recorta por los bordes.
Load y LoadAsync	<code>Load(url)</code> actualiza la imagen desde una URL de forma síncrona; <code>LoadAsync(url)</code> hace la misma petición de forma asíncrona, así que un servidor lento no bloquea la pantalla.
Vectorial frente a ráster	Un SVG es marcado que se dibuja al tamaño que pida el diseño, y una ilustración monocroma adecuada se puede recolorar; un archivo ráster es una rejilla fija de píxeles que solo puede remuestrearse.

3. Desarrollo de la lección

Dos listas de formatos, porque hay dos canalizaciones

El primer objetivo de este módulo es dejar de citar una sola lista. Para `PictureBox`, la documentación de controles de Wisej.NET describe gráficos cargados desde archivos bitmap, metafile, icon, JPEG, GIF y PNG. Ese es el mundo conocido de `System.Drawing.Image`: le entregas un objeto al control y el servidor lo convierte en datos PNG al enviarlo. Para los recursos en forma de cadena, las extensiones de imagen documentadas de `resource.wx` son .jpg, .jpeg, .png, .gif, .svg y .bmp, y las definiciones de imágenes de tema admiten png, gif, svg, jpg y bmp.

Las entradas interesantes son las que aparecen en una sola lista. Icon y metafile pertenecen al lado de System.Drawing y no forman parte de la lista de imágenes permitidas de `resource.wx`. Con SVG ocurre lo contrario: es un valor excelente de origen de imagen y una imagen de tema de primera clase, pero no es un formato de `System.Drawing.Image` en el mismo sentido que un Bitmap o un PNG. Decir «Wisej.NET admite SVG» sin nombrar la canalización es lo que produce una asignación fallida de `Image` más adelante en el sprint.

En el laboratorio lo anotas de forma explícita: la nota de formatos junto a tu página ImageLab enumera todos los recursos y nombra la canalización que los llevó.

PictureBox tiene dos puntos de entrada, y eliges uno por control

El segundo objetivo trata de elegir la propiedad correcta. `PictureBox.Image` recibe un `System.Drawing.Image`. `PictureBox.ImageSource` recibe una cadena. La regla que mantiene legible un proyecto es sencilla: si tu código C# está creando o transformando los píxeles, usa `Image`; si la imagen es un archivo que el navegador podría obtener igual de bien por su cuenta, usa `ImageSource` y mantén la cadena con significado.

```
pictureBox1.ImageSource = "Images/product.png"; // a string the client resolves
pictureBox1.SizeMode = PictureBoxSizeMode.Zoom;

pictureBox2.ImageSource = "resource.wx/IconDesk/Logo.svg";
pictureBox2.SizeMode = PictureBoxSizeMode.Cover;

// a URL the user chose: fetch it without holding the page
pictureBox3.LoadAsync(remoteJpegUrl);
```

Los puntos de entrada por URL conviven con esas propiedades. `Load(url)` actualiza la imagen desde una URL de forma síncrona; `LoadAsync(url)` inicia en su lugar una carga asíncrona. Tu página ImageLab usa `LoadAsync` para el JPEG remoto precisamente para que un servidor externo lento se degrade a «la imagen llega un momento después» en lugar de a «la página está congelada».

Seis modos de tamaño, y qué le hace cada uno a la imagen

Wisej.NET documenta seis valores de `PictureBoxSizeMode`, y el tercer objetivo es elegir entre ellos de forma deliberada en lugar de aceptar lo que dejó el diseñador. `Normal` dibuja la imagen en su tamaño original desde la esquina superior izquierda y recorta lo que no cabe. `AutoSize` invierte la relación: el PictureBox crece o se encoge hasta la imagen. `CenterImage` conserva el tamaño original pero la centra, de modo que el recorte se reparte por igual en todos los lados. `StretchImage` llena el cuadro sobreviva o no la relación de aspecto. `Zoom` escala la imagen conservando la relación de aspecto hasta que cabe entera. `Cover` también conserva la relación de aspecto, pero llena el rectángulo y recorta los bordes que sobresalen.

En la práctica, `Zoom` es la opción segura para logotipos y fotos que deben verse completos, `Cover` es la adecuada para miniaturas y banners donde llenar el rectángulo importa más que mostrar todos los bordes, y `StretchImage` debería ser una decisión que alguien tomó a propósito, porque es el modo que deforma sin avisar una marca corporativa. El laboratorio muestra un mismo recurso ráster cuatro veces, en `Normal`, `Zoom`, `Cover` y `StretchImage`, para que el recorte y la deformación se vean uno al lado del otro en vez de describirse.

SVG para los iconos de interfaz, ráster para las fotografías

El cuarto objetivo es la regla de diseño que se deriva de las dos canalizaciones. SVG suele ser la primera opción para los iconos de interfaz: es marcado vectorial, así que se mantiene nítido a 16, 24, 32 y 48 píxeles y con cualquier factor de escala de pantalla, una ilustración monocroma sencilla ocupa poco y Wisej.NET puede recolorar un SVG adecuado mediante la cadena de origen de imagen. Las fotografías y

las capturas de pantalla van por el otro camino. PNG es la opción ráster de uso general cuando importan la transparencia o los píxeles exactos, JPEG suele ser mejor para fotografías en las que la compresión con pérdida es aceptable, GIF sigue siendo útil para gráficos sencillos o animados, y BMP se admite pero suele ser mucho más grande que un formato web comprimido.

La alta densidad de píxeles es donde la diferencia deja de ser teórica. Un icono vectorial no necesita una segunda copia para escalas del 150 % o del 200 %; un bitmap de 16 píxeles ampliado a 32 píxeles simplemente se ve ampliado. Si tienes que distribuir iconos ráster, parte de resolución suficiente para el mayor tamaño que vayas a dibujar y pruébalo en el navegador con una escala de pantalla elevada.

- Glifos de comando, navegación y estado: SVG mediante `ImageSource`.
- Fotos de producto, capturas de pantalla y escaneos: PNG o JPEG, normalmente con `Zoom` o `Cover`.
- Píxeles que dibuja tu propio código: un `System.Drawing.Image` en `Image`.

El tamaño es una decisión de diseño, no un accidente

Un recurso de imagen tiene dimensiones intrínsecas, pero tu interfaz también tiene un tamaño de icono objetivo, y no son el mismo número. Los temas de Wisej suelen controlar las métricas de los iconos, y una `ImageList` puede fijar un tamaño común explícito y redimensionar las imágenes que dibujan los controles. Para una `Label`, la documentación señala que se usa el tamaño original de la imagen salvo que participe una `ImageList`.

El último entregable cierra el círculo con el Módulo 1: un manejador de informe recorre todos los `PictureBox` de la página `ImageLab` e indica si su imagen llegó a través de `Image` o de `ImageSource`. Ese manejador es la forma más rápida de encontrar un control en el que alguien asignó ambas propiedades y solo una está ganando.



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:

- ✓ Identificar los formatos de imagen que Wisej.NET acepta mediante `PictureBox`, los temas y `resource.wx`.
- ✓ Usar correctamente `PictureBox.Image` y `PictureBox.ImageSource`.
- ✓ Elegir los modos de tamaño de `PictureBox`, incluidos `Normal`, `AutoSize`, `CenterImage`, `StretchImage`, `Zoom` y `Cover`.

- ✓ Explicar por qué SVG suele preferirse para los iconos escalables de interfaz y por qué las fotos suelen ser imágenes ráster.

Guía del laboratorio / examen

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



Laboratorio práctico

Construye una página ImageLab en IconDesk que haga visibles las dos canalizaciones una al lado de la otra. Muestra un mismo recurso ráster cuatro veces con SizeMode en Normal, Zoom, Cover y StretchImage para que las diferencias de recorte y distorsión resulten evidentes, añade un PictureBox que traiga un JPEG remoto con LoadAsync y mantenga la página receptiva mientras llega, y coloca un SVG asignado mediante ImageSource a 16, 24, 32 y 48 píxeles para demostrar que se mantiene nítido. Termina con un manejador que informe, para cada PictureBox de la página, si su imagen vino de Image o de ImageSource.

Punto de partida

Abre la solución IconDesk que creaste en el Módulo 1, donde el panel de comandos ya lleva cuatro iconos de botón a través de cuatro mecanismos distintos. Añade al proyecto una nueva página llamada **ImageLab** y ten a mano un recurso ráster, un JPEG de tipo fotografía alojado en un servidor remoto y un icono SVG monocromo antes de empezar a colocar controles.

Enfoque sugerido

- ✓ Coloca en ImageLab cuatro controles PictureBox de tamaño idéntico y dales el mismo recurso ráster, después establece **SizeMode** en Normal, Zoom, Cover y StretchImage y etiqueta cada uno con el modo que demuestra.
- ✓ Añade un quinto PictureBox y llama a **LoadAsync** con la URL del JPEG remoto desde la carga de la página, dejando un marcador o un pie de texto visible para que el usuario vea que la página siguió siendo utilizable mientras la imagen estaba en camino.
- ✓ Coloca otros cuatro controles PictureBox de 16, 24, 32 y 48 píxeles, asigna a cada uno el mismo SVG mediante **ImageSource** y compáralos con el recurso ráster escalado a esos mismos cuatro tamaños.
- ✓ Escribe un manejador **btnReport_Click** que recorra los controles de la página y, por cada PictureBox, escriba una línea que nombre el control e indique si su imagen vino de **Image** o de **ImageSource**.
- ✓ Escribe la nota de formatos: enumera todos los recursos que usa la página, indica cuáles viajan por System.Drawing y cuáles siguen siendo una cadena de recurso, y anota qué modo de tamaño defenderías en una revisión para un logotipo, una miniatura y una fotografía.

Entregables requeridos

- ✓ Página ImageLab que muestra un mismo recurso ráster en los modos de tamaño Normal, Zoom, Cover y StretchImage
- ✓ PictureBox que obtiene un JPEG remoto con LoadAsync sin bloquear la página
- ✓ Un SVG asignado mediante ImageSource y renderizado a 16, 24, 32 y 48 píxeles

- ✓ Manejador de informe que indica, para cada PictureBox, si la imagen la suministró Image o ImageSource
- ✓ Nota de formatos que enumere qué recursos del laboratorio viajan por System.Drawing y cuáles siguen siendo cadenas de recurso

Preguntas de repaso

- ✓ Tu recurso ráster es más ancho que alto y el cuadro es cuadrado. ¿Cuál de Zoom y Cover pierde parte de la imagen, cuál deja espacio vacío y cuál defenderías para el logotipo de un cliente?
- ✓ El servidor del JPEG remoto está inaccesible durante diez segundos. ¿Qué experimenta el usuario de tu página con `LoadAsync` y qué habría experimentado con el `Load` síncrono?
- ✓ Tu manejador de informe dice que la imagen de un control vino de `ImageSource`, pero el archivo del diseñador también asigna `Image` en ese mismo control. ¿Cuál se dibuja realmente y cómo lo demostrarías en lugar de adivinarlo?



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.