

Bitmap, SVG e PictureBox: formati, dimensionamento e caricamento

Guida della lezione · Guida al lab / esame

● LearnWisej

Guida della lezione

Due elenchi di formati perché le pipeline sono due: bitmap, metafile, icon, JPEG, GIF e PNG attraverso System.Drawing e PictureBox.Image, contro jpg, jpeg, png, gif, svg e bmp che resource.wx e le immagini del tema accettano come stringhe, insieme alle modalità di ridimensionamento della PictureBox e ai punti di ingresso Load sincrono e LoadAsync asincrono. Che cosa significa per uno sviluppatore che conosce le basi di C# e ha già lavorato un po' con i controlli e il designer di Wisej.NET — e come affrontare questo modulo.



Focus di produzione

Due elenchi di formati perché le pipeline sono due: bitmap, metafile, icon, JPEG, GIF e PNG attraverso System.Drawing e PictureBox.Image, contro jpg, jpeg, png, gif, svg e bmp che resource.wx e le immagini del tema accettano come stringhe, insieme alle modalità di ridimensionamento della PictureBox e ai punti di ingresso Load sincrono e LoadAsync asincrono.

1. Il problema in produzione

"Quali formati di immagine supporta Wisej.NET?" è la domanda che ogni team si pone nella prima settimana, e la risposta onesta dipende da quale percorso di API prende l'immagine. Rispondi con un unico elenco vago e ti ritrovi un logo stirato su un portatile ad alta densità, una pagina che si blocca mentre la foto di un prodotto scende da un host lento e una segnalazione di bug perché un file `.ico` non si risolve attraverso una stringa di risorsa. Questo modulo sostituisce l'elenco vago con due elenchi precisi, ciascuno legato alla proprietà del controllo che lo consuma.

2. Concetti chiave

Concetto	Perché è importante
pipeline di System.Drawing	La proprietà <code>Image</code> contiene un <code>System.Drawing.Image</code> nella memoria del server; Wisej.NET lo invia al browser come dati PNG. Le sue sorgenti documentate sono i file bitmap, metafile, icon, JPEG, GIF e PNG.
pipeline della sorgente immagine	Una stringa che il client risolve da sé: un URL relativo o assoluto, un'immagine del tema indicata per nome oppure un percorso <code>resource.wx</code> . Le estensioni di immagine documentate per <code>resource.wx</code> sono .jpg, .jpeg, .png, .gif, .svg e .bmp.
PictureBox.Image	Il punto di ingresso per i pixel che il tuo codice C# possiede già o ha appena generato, come un grafico disegnato con <code>Graphics</code> .
PictureBox.ImageSource	Il punto di ingresso per un'immagine che il browser può scaricare da sé. Al posto dei byte dell'immagine viaggia una breve stringa, ed è per questo che l'SVG appartiene a questa pipeline.
PictureBox.SizeMode	I sei valori <code>Normal</code> , <code>AutoSize</code> , <code>CenterImage</code> , <code>StretchImage</code> , <code>Zoom</code> e <code>Cover</code> , che decidono se un'immagine viene tagliata, centrata, distorta, adattata o ritagliata.
Load e LoadAsync	<code>Load(url)</code> aggiorna l'immagine da un URL in modo sincrono; <code>LoadAsync(url)</code> esegue lo stesso scaricamento in modo asincrono, così un host lento non tiene ferma la schermata.
vettoriale contro raster	Un SVG è markup disegnato alla dimensione che il layout richiede, e un disegno monocromatico adatto si può ricolorare; un file raster è una griglia fissa di pixel che può soltanto essere ricampionata.

3. Il filo della lezione

Due elenchi di formati, perché le pipeline sono due

Il primo obiettivo di questo modulo è smettere di citare un elenco solo. Per la `PictureBox`, la documentazione dei controlli di Wisej.NET descrive immagini caricate da file bitmap, metafile, icon, JPEG, GIF e PNG. È il familiare mondo di `System.Drawing.Image`: passi al controllo un oggetto, e il server lo trasforma in dati PNG sulla rete. Per le risorse a stringa, le estensioni di immagine documentate per `resource.wx` sono .jpg, .jpeg, .png, .gif, .svg e .bmp, e le definizioni delle immagini del tema supportano png, gif, svg, jpg e bmp.

Le voci interessanti sono quelle che compaiono in un elenco solo. Icon e metafile appartengono al lato System.Drawing e non fanno parte dell'elenco delle immagini ammesse da `resource.wx`. Con l'SVG succede il contrario: è un ottimo valore di sorgente immagine e un'immagine del tema di prima classe, ma non è un formato di `System.Drawing.Image` nello stesso senso in cui lo sono una Bitmap o un PNG. Dire "Wisej.NET supporta l'SVG" senza dare un nome alla pipeline è ciò che produce un'assegnazione a `Image` fallita più avanti nello sprint.

Nel lab lo metti nero su bianco: la nota sui formati accanto alla tua pagina ImageLab elenca ogni asset e nomina la pipeline che lo ha portato.

La PictureBox ha due punti di ingresso, e ne scegli uno per controllo

Il secondo obiettivo riguarda la scelta della proprietà giusta. `PictureBox.Image` riceve un `System.Drawing.Image`. `PictureBox.ImageSource` riceve una stringa. La regola che tiene leggibile un progetto è semplice: se è il tuo codice C# a creare o trasformare i pixel, usa `Image`; se l'immagine è un file che il browser potrebbe altrettanto bene scaricare da sé, usa `ImageSource` e tieni la stringa significativa.

```
pictureBox1.ImageSource = "Images/product.png"; // una stringa che il client risolve
pictureBox1.SizeMode = PictureBoxSizeMode.Zoom;

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

// un URL scelto dall'utente: scaricalo senza tenere ferma la pagina
pictureBox3.LoadAsync(remoteImageUrl);
```

I punti di ingresso per gli URL stanno accanto a quelle proprietà. `Load(url)` aggiorna l'immagine da un URL in modo sincrono; `LoadAsync(url)` avvia invece un caricamento asincrono. La tua pagina ImageLab usa `LoadAsync` per il JPEG remoto proprio perché un host di terze parti lento degradi in "l'immagine arriva un attimo dopo" invece che in "la pagina è bloccata".

Sei modalità di ridimensionamento, e che cosa fa ciascuna all'immagine

Wisej.NET documenta sei valori di `PictureBoxSizeMode`, e il terzo obiettivo è sceglierli deliberatamente invece di accettare quello che il designer ha lasciato lì. `Normal` disegna l'immagine alla sua dimensione originale a partire dall'angolo in alto a sinistra e taglia ciò che non ci sta. `AutoSize` ribalta la relazione: è la PictureBox a crescere o ridursi fino all'immagine. `CenterImage` mantiene la dimensione originale ma la centra, così il taglio avviene in modo uniforme su tutti i lati. `StretchImage` riempie il riquadro che le proporzioni sopravvivano o meno. `Zoom` scala l'immagine conservandone le proporzioni finché l'intera immagine non ci sta. `Cover` conserva anch'essa le proporzioni ma riempie il rettangolo, ritagliando i bordi che escono.

Nella pratica, `Zoom` è la scelta prudente per i loghi e per le foto che devono restare completamente visibili, `Cover` è quella giusta per miniature e banner, dove riempire il rettangolo conta più che mostrare ogni bordo, e `StretchImage` dovrebbe essere una decisione presa da qualcuno di proposito, perché è la modalità che distorce un marchio senza farsi notare. Il lab mostra uno stesso asset raster quattro volte, in `Normal`, `Zoom`, `Cover` e `StretchImage`, così che il taglio e la distorsione si vedano affiancati invece di essere soltanto descritti.

SVG per le icone di interfaccia, raster per le fotografie

Il quarto obiettivo è la regola di progettazione che discende dalle due pipeline. L'SVG è di solito la prima scelta per le icone di interfaccia: è markup vettoriale, quindi resta nitido a 16, 24, 32 e 48 pixel e a ogni fattore di scala di visualizzazione, un semplice disegno monocromatico occupa poco, e Wisej.NET può ricolorare un SVG adatto attraverso la stringa di sorgente immagine. Fotografie e screenshot vanno nella direzione opposta. Il PNG è la scelta raster di uso generale quando contano la trasparenza o i pixel esatti, il

JPEG è normalmente migliore per le fotografie dove la compressione con perdita è accettabile, il GIF resta utile per immagini semplici o animate, e il BMP è supportato ma di solito è molto più pesante di un formato web compresso.

È con l'alta densità che la differenza smette di essere teorica. Un'icona vettoriale non ha bisogno di una seconda copia per la scala al 150% o al 200%; una bitmap da 16 pixel ingrandita a 32 pixel sembra semplicemente ingrandita. Se devi per forza distribuire icone raster, parti da una risoluzione sufficiente per la dimensione più grande che disegnerai, e prova nel browser con una scala di visualizzazione maggiorata.

- Glifi di comando, di navigazione e di stato: SVG tramite `ImageSource`.
- Foto di prodotto, screenshot e scansioni: PNG o JPEG, di solito con `Zoom` o `Cover`.
- Pixel disegnati dal tuo codice: un `System.Drawing.Image` su `Image`.

Il dimensionamento è una decisione di progetto, non un caso

Una risorsa immagine ha dimensioni intrinseche, ma anche la tua interfaccia ha una dimensione obiettivo per le icone, e i due numeri non coincidono. I temi di Wisej controllano tipicamente le metriche delle icone, e una `ImageList` può stabilire una dimensione comune esplicita e ridimensionare le immagini che i controlli disegnano. Per una `Label`, la documentazione segnala che viene usata la dimensione originale dell'immagine, a meno che non entri in gioco una `ImageList`.

L'ultimo deliverable chiude il cerchio con il Modulo 1: un gestore di report percorre ogni `PictureBox` della pagina `ImageLab` e indica se la sua immagine è arrivata tramite `Image` oppure tramite `ImageSource`. Quel gestore è il modo più rapido per trovare un controllo su cui qualcuno ha impostato entrambe le proprietà e solo una delle due sta vincendo.



Dai prima un nome al meccanismo

Prima di assegnare qualsiasi immagine, dichiara quale meccanismo stai usando: un oggetto `System.Drawing.Image` che il server disegna come PNG, una stringa di sorgente immagine che il browser risolve, un'immagine del tema indicata per nome, una risorsa incorporata `resource.wx` o un font di icone disegnato come HTML. Assegna una sola proprietà immagine per controllo, così che le altre non possano azzerarla di nascosto, mantieni la stringa della sorgente significativa invece di un percorso magico incollato, preferisci l'SVG per le icone di interfaccia così che restino nitide a ogni dimensione e scala, e dai a ogni icona che porta un significato un'alternativa testuale.



Obiettivi di apprendimento

Alla fine di questo modulo sarai in grado di:

- ✓ Individuare i formati di immagine che Wisej.NET accetta tramite `PictureBox`, temi e `resource.wx`.
- ✓ Usare correttamente `PictureBox.Image` e `PictureBox.ImageSource`.
- ✓ Scegliere le modalità di ridimensionamento della `PictureBox` tra `Normal`, `AutoSize`, `CenterImage`, `StretchImage`, `Zoom` e `Cover`.
- ✓ Spiegare perché l'SVG è di solito preferibile per le icone scalabili dell'interfaccia e perché le foto sono di solito immagini raster.

Guida al lab / esame

Che cosa costruirai nel lab pratico, l'approccio suggerito e i deliverable richiesti.



Lab pratico

Costruisci in IconDesk una pagina ImageLab che renda visibili le due pipeline una accanto all'altra. Mostra uno stesso asset raster quattro volte con SizeMode impostato su Normal, Zoom, Cover e StretchImage, così che le differenze di ritaglio e di distorsione siano evidenti, aggiungi una PictureBox che scarica un JPEG remoto con LoadAsync mantenendo la pagina reattiva mentre arriva, e inserisci un SVG assegnato tramite ImageSource a 16, 24, 32 e 48 pixel per dimostrare che resta nitido. Chiudi con un gestore che segnala, per ogni PictureBox della pagina, se l'immagine è arrivata da Image oppure da ImageSource.

Punto di partenza

Apri la soluzione IconDesk che hai creato nel Modulo 1, dove il pannello dei comandi porta già quattro icone sui pulsanti attraverso quattro meccanismi diversi. Aggiungi al progetto una nuova pagina chiamata **ImageLab** e tieni a portata di mano un asset raster, un JPEG in stile fotografico ospitato su un host remoto e un'icona SVG monocromatica prima di iniziare a disporre i controlli.

Approccio suggerito

- ✓ Disponi su ImageLab quattro controlli PictureBox di dimensioni identiche e assegna loro lo stesso asset raster, poi imposta **SizeMode** su Normal, Zoom, Cover e StretchImage ed etichetta ciascuno con la modalità che dimostra.
- ✓ Aggiungi una quinta PictureBox e chiama **LoadAsync** con l'URL del JPEG remoto dal percorso di caricamento della pagina, lasciando un segnaposto o una didascalia visibile perché l'utente veda che la pagina è rimasta utilizzabile mentre l'immagine era in arrivo.
- ✓ Colloca altri quattro controlli PictureBox di 16, 24, 32 e 48 pixel, assegna a ciascuno lo stesso SVG tramite **ImageSource** e confrontali con l'asset raster scalato alle stesse quattro dimensioni.
- ✓ Scrivi un gestore **btnReport_Click** che percorre i controlli della pagina e, per ogni PictureBox, scrive una riga con il nome del controllo e se la sua immagine è arrivata da **Image** oppure da **ImageSource**.
- ✓ Scrivi la nota sui formati: elenca ogni asset usato dalla pagina, indica quali passano da System.Drawing e quali restano una stringa di risorsa, e annota quale modalità di ridimensionamento difenderesti in una revisione per un logo, per una miniatura e per una fotografia.

Deliverable richiesti

- ✓ Pagina ImageLab che mostra uno stesso asset raster nelle modalità di ridimensionamento Normal, Zoom, Cover e StretchImage
- ✓ PictureBox che scarica un JPEG remoto con LoadAsync senza bloccare la pagina
- ✓ Un SVG assegnato tramite ImageSource e mostrato a 16, 24, 32 e 48 pixel

- ✓ Gestore di report che indica, per ciascuna PictureBox, se l'immagine è stata fornita da Image o da ImageSource
- ✓ Nota sui formati che elenca quali asset del lab passano da System.Drawing e quali restano stringhe di risorsa

Domande di autovalutazione

- ✓ Il tuo asset raster è più largo che alto e il riquadro è quadrato. Tra Zoom e Cover, quale perde una parte dell'immagine, quale lascia spazio vuoto e quale difenderesti per il logo di un cliente?
- ✓ L'host del JPEG remoto è irraggiungibile per dieci secondi. Che cosa vive l'utente della tua pagina con `LoadAsync`, e che cosa avrebbe vissuto con il `Load` sincrono?
- ✓ Il tuo gestore di report dice che l'immagine di un controllo è arrivata da `ImageSource`, ma il file del designer assegna anche `Image` sullo stesso controllo. Quale delle due viene davvero disegnata, e come lo dimostreresti invece di tirare a indovinare?



Dai prima un nome al meccanismo

Prima di assegnare qualsiasi immagine, dichiara quale meccanismo stai usando: un oggetto `System.Drawing.Image` che il server disegna come PNG, una stringa di sorgente immagine che il browser risolve, un'immagine del tema indicata per nome, una risorsa incorporata `resource.wx` o un font di icone disegnato come HTML. Assegna una sola proprietà immagine per controllo, così che le altre non possano azzerarla di nascosto, mantieni la stringa della sorgente significativa invece di un percorso magico incollato, preferisci l'SVG per le icone di interfaccia così che restino nitide a ogni dimensione e scala, e dai a ogni icona che porta un significato un'alternativa testuale.