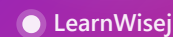


Il modello delle immagini di Wisej.NET: Image, ImageSource e ImageList

Guida della lezione · Guida al lab / esame



Guida della lezione

Le due pipeline delle immagini: una proprietà Image che contiene un System.Drawing.Image che il server invia al browser come PNG, contro una stringa ImageSource o IconSource che il client risolve come URL, nome di un'immagine del tema, percorso resource.wx o elemento di un pacchetto di icone, più il contratto condiviso Image e la ImageList che permette a molti controlli di condividere un unico insieme di immagini indicizzate per chiave. 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

Le due pipeline delle immagini: una proprietà Image che contiene un System.Drawing.Image che il server invia al browser come PNG, contro una stringa ImageSource o IconSource che il client risolve come URL, nome di un tema, percorso resource.wx o elemento di un pacchetto di icone, più il contratto condiviso Image e l'ImageList che permette a molti controlli di condividere un unico insieme di immagini con chiavi.

1. Il problema in produzione

Wisej.NET offre di proposito più di un modo per mettere una grafica su un controllo, e funzionano tutti. I guai iniziano quando un team li usa tutti per caso: una toolbar in cui un pulsante legge un file dal disco del server, quello accanto punta al nome di un tema e un terzo ha perso silenziosamente la sua icona perché sullo stesso controllo sono state impostate due proprietà. Sei mesi dopo nessuno sa più spiegare da dove arrivi davvero una certa immagine né perché sia sparita dopo il deploy.

2. Concetti chiave

Concetto	Perché è importante
proprietà Image	Un oggetto <code>System.Drawing.Image</code> che vive nella memoria del server; Wisej.NET lo invia al browser come dati PNG, quindi l'immagine viaggia come byte che il tuo codice ha già decodificato.
stringa ImageSource	Una stringa che identifica una risorsa che il client dovrà risolvere: un URL relativo, un URL assoluto, un'immagine del tema indicata per nome, un percorso <code>resource.wx</code> o un elemento dentro l'assembly di un pacchetto di icone.
IconSource	La stessa idea di stringa sui controlli che possiedono uno slot icona dedicato accanto all'immagine principale: controlla l'API del controllo che stai usando invece di dare per scontato che tutti chiamino la proprietà allo stesso modo.
contratto IImage	La famiglia comune di proprietà immagine che i controlli con supporto alle immagini espongono — <code>Image</code> , <code>ImageAlign</code> , <code>ImageIndex</code> , <code>ImageKey</code> , <code>ImageList</code> e <code>ImageSource</code> — che permette a un unico metodo di supporto di decorare in modo uniforme un Button, una Label o qualunque altro controllo.
ImageList	Una collezione riutilizzabile di oggetti immagine a cui più controlli possono puntare, indirizzata per posizione o per nome, e in grado di fissare una dimensione comune delle immagini per i controlli che la usano.
ImageKey contro ImageIndex	Una chiave come <code>"delete"</code> porta un significato e sopravvive ai riordini; l'indice <code>3</code> diventa silenziosamente l'immagine sbagliata il giorno in cui qualcuno ne inserisce una più in alto.
Assegnazione mutuamente esclusiva	Tutte queste proprietà alimentano un unico slot visivo, quindi assegnarne una può azzerare le altre — ed è per questo che l'icona sbagliata di solito è una seconda assegnazione, non un file rotto.

3. Il filo della lezione

Oggetto o stringa: la prima domanda a cui rispondere

Prima di scrivere qualsiasi cosa, decidi se è il server a possedere un *oggetto* immagine oppure se è il browser a dover risolvere un *identificatore* di immagine. Sono queste le due pipeline, e quasi tutta la confusione che arriva più avanti in questo corso nasce dal mescolarle. Quando assegni `Image` consegna a Wisej.NET un `System.Drawing.Image`: veri dati di pixel decodificati che stanno nella memoria della tua sessione e che il framework invia poi al browser come PNG. È la strada giusta quando il tuo C# ha davvero prodotto o ricevuto un oggetto immagine — un grafico che hai disegnato, una miniatura che hai ridimensionato, una foto arrivata come stream da un servizio.

Quando assegni `ImageSource` o `IconSource`, invece, consegna a Wisej.NET una stringa. Sul server non viene decodificato nulla. La stringa indica qualcosa che il client può scaricare e mettere in cache da solo: `"Images/open.png"` relativo all'applicazione, un indirizzo assoluto, il nome di un'immagine del tema come `"icon-save"` oppure un percorso `resource.wx` dentro un assembly. Poiché la risorsa conserva la propria identità, un SVG resta un SVG fino al browser, ed è per questo che le icone di interfaccia restano

nitide a qualsiasi dimensione e possono essere ricolorate in seguito. In IconDesk vedrai entrambe le cose: `btnOpen` riceve un oggetto immagine caricato dal server, `btnSave` riceve il nome di un tema come stringa, e la label diagnostica deve saperli distinguere.

Un'unica famiglia di proprietà, chiamata `IImage`

Wisej.NET non dà a ogni controllo un vocabolario tutto suo per le immagini. I controlli con supporto alle immagini condividono un contratto chiamato `IImage` che raccoglie `Image`, `ImageAlign`, `ImageIndex`, `ImageKey`, `ImageList` e `ImageSource` in un'unica famiglia. Per chi inizia, riconoscere la famiglia vale più che farne il cast: quando incontri un controllo nuovo sai già quali proprietà cercare e quali quattro di esse si contendono lo stesso slot.

Il cast resta comunque utile nel codice infrastrutturale. Un metodo che accetta un `Control`, verifica `is IImage` e poi imposta `ImageSource` può applicare l'icona di salvataggio del progetto a un Button, a una Label o a una voce di menu senza uno switch. La label diagnostica di IconDesk nel lab fa il contrario: legge ogni pulsante attraverso `IImage`, controlla quale delle quattro proprietà è valorizzata e stampa il meccanismo dietro l'icona invece di indovinarlo dall'immagine.

Scegliere il meccanismo adatto all'asset che hai davanti

Quattro casi coprono quasi tutto. Un file distribuito accanto all'applicazione è una sorgente immagine con un URL relativo. Un'immagine ospitata altrove è una sorgente immagine con un URL assoluto. Un'icona che il tema attivo definisce già è una sorgente immagine che contiene il nome che il tema le dà, il che significa che il client la prende dalla cache del tema che ha comunque caricato. Un'immagine che il tuo codice ha creato, trasformato o ricevuto come oggetto .NET è un `Image`.

Il quinto caso è quello condiviso, ed è la ragione per cui esiste `ImageList`. Un `ImageList` contiene una collezione di oggetti immagine; i controlli fanno riferimento a una voce tramite l' `ImageIndex` numerico o la `ImageKey` testuale. In un'applicazione reale contano due proprietà di questa indirazione: molti controlli possono condividere un'unica collezione, e l'immagine memorizzata sotto una chiave può essere sostituita senza toccare un solo controllo che fa riferimento a quella chiave. La lista può anche fissare una dimensione delle immagini per i controlli che la usano, comodo quando altrimenti sarebbe il tema a decidere per te la dimensione delle icone.

```
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 prende invece la sua icona da una stringa – nessun ImageList coinvolto.
btnSave.ImageSource = "icon-save";
```

Nota che cosa le prime righe *non* stanno facendo. `Image.FromFile` legge un file sul server, mai sulla macchina dell'utente, quindi un percorso assoluto che funzionava sul tuo portatile è un difetto di deploy

pronto a esplodere. Una stringa di sorgente immagine, al contrario, viene indirizzata come il browser indirizza tutto il resto.

Perché l'icona che imposti non è sempre l'icona che vedi

Un controllo ha un solo slot visivo per la propria immagine, e `Image`, `ImageSource`, `ImageIndex` e `ImageKey` se lo contendono tutte. Wisej.NET risolve la cosa rendendole mutuamente esclusive: impostarne una può azzerare le altre, così la configurazione non è mai ambigua. Se `btnDelete` sta tranquillamente mostrando la voce della lista sotto `ImageKey = "delete"` e una riga successiva imposta `ImageSource`, la stringa diventa la rappresentazione attiva e la chiave smette di contare.

Questa sola regola spiega gran parte delle segnalazioni del tipo "si vede l'icona sbagliata". Il designer ha impostato una proprietà, il costruttore della form un'altra e un tema o un metodo di supporto una terza. Quando l'immagine è sbagliata non iniziare aprendo il file: ispeziona tutte le proprietà del controllo legate alle immagini e trova l'ultima assegnazione che è stata eseguita.

- Una sola proprietà immagine per controllo, scelta in modo deliberato.
- Chiavi, non indici, ovunque l'identità abbia un significato.
- Un'alternativa testuale per ogni icona che porta da sola un significato.



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:

- ✓ Spiegare la differenza tra un valore `System.Drawing.Image` e una stringa immagine come `ImageSource` o `IconSource`.
- ✓ Usare il modello comune `Image` per riconoscere le proprietà `Image`, `ImageSource`, `ImageList`, `ImageIndex` e `ImageKey`.
- ✓ Scegliere il meccanismo immagine corretto per una bitmap locale, un URL, un'immagine del tema o un'icona riutilizzabile.
- ✓ Capire perché assegnare una di più proprietà immagine mutuamente esclusive può azzerare le altre.

Guida al lab / esame

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



Lab pratico

Crea il progetto IconDesk e dotalo di un pannello comandi con i pulsanti Open, Save, Print e Delete, ciascuno con la propria icona portata da un meccanismo diverso. Carica btnOpen da un `System.Drawing.Image` lato server, assegna un'immagine del tema indicata per nome a `btnSave.ImageSource` e fai puntare btnPrint e btnDelete a un unico `ImageList` condiviso tramite `ImageKey`. Aggiungi una label diagnostica che ispeziona ogni pulsante attraverso `Image` e riporta quale meccanismo è davvero in gioco, poi cambia l'immagine memorizzata sotto una chiave dell'`ImageList` e verifica che ogni controllo che usa quella chiave segua il cambiamento.

Punto di partenza

Questo è il primo modulo, quindi parti da zero: crea una nuova applicazione Wisej.NET 4 chiamata **IconDesk** e aggiungi una pagina per il pannello comandi. Metti accanto all'applicazione una piccola cartella di immagini — basta una grafica per open, save, print e delete — e verifica che il progetto vuoto compili e si apra nel browser prima di assegnare una sola icona.

Approccio suggerito

- ✓ Disponi prima i quattro pulsanti con il solo testo, così che il pannello sia leggibile prima che esista un'immagine.
- ✓ Dai a `btnOpen` un `System.Drawing.Image` caricato dal server e a `btnSave` il nome di un'immagine del tema su `ImageSource`; dichiara a voce alta quale pipeline usa ciascuno dei due.
- ✓ Costruisci un unico `ImageList`, aggiungi le immagini di stampa ed eliminazione sotto chiavi significative, poi fai puntare entrambi i pulsanti alla lista tramite `ImageKey` invece che `ImageIndex`.
- ✓ Scrivi la label diagnostica: percorri i controlli del pannello, verifica su ciascuno `IImage` e riporta quale tra `Image`, `ImageSource`, `ImageList` più `ImageKey` è davvero valorizzata.
- ✓ Sostituisci l'immagine memorizzata sotto la chiave delete, ricarica la pagina, poi imposta deliberatamente una seconda proprietà immagine su un pulsante per poter annotare che fine ha fatto la prima assegnazione.

Deliverable richiesti

- ✓ Pannello comandi di IconDesk con i pulsanti Open, Save, Print e Delete
- ✓ btnOpen che porta un `System.Drawing.Image` caricato sul server e btnSave un'immagine del tema indicata per nome su `ImageSource`
- ✓ btnPrint e btnDelete che condividono un unico `ImageList` tramite `ImageKey` invece che `ImageIndex`

- ✓ Label diagnostica che legge ogni pulsante attraverso l'Image e dà un nome al meccanismo dietro la sua icona
- ✓ Nota di lab che registra che cosa è successo quando è stata sostituita una chiave dell'ImageList e quando sullo stesso pulsante sono state impostate due proprietà immagine

Domande di autovalutazione

- ✓ La tua label diagnostica dice che `btnDelete` usa una sorgente immagine, ma tu hai impostato solo la sua `ImageKey`. Che cosa deve essere stato eseguito, e dove lo cercheresti?
- ✓ Sia print sia delete sono cambiate quando hai sostituito una voce dell' `ImageList`. Sarebbe ancora vero se i pulsanti usassero `ImageIndex`, e che cosa succederebbe se una quinta immagine venisse inserita in cima alla lista?
- ✓ Il pulsante Open legge il proprio file sul server. Quale parte di questa soluzione si rompe quando l'applicazione viene distribuita su un'altra macchina, e quale non si romperebbe se la stessa immagine fosse una stringa di sorgente immagine?



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.