

Font di icone, contenuto HTML e strategia delle icone in produzione

Guida della lezione · Guida al lab / esame



Guida della lezione

L'ultimo meccanismo e la policy che li tiene insieme tutti: un foglio di stile di font di icone caricato da Default.html, AllowHtml sui controlli che devono interpretare il markup, glifi di icone dentro le celle di una DataGridView con click sugli elementi distinti per ruolo, e una checklist di produzione che copre accessibilità, licenze, caching, dipendenze esterne e coerenza visiva. 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

L'ultimo meccanismo e la policy che li tiene insieme tutti: un foglio di stile di font di icone caricato da Default.html, AllowHtml sui controlli che devono interpretare il markup, glifi di icone dentro le celle di una DataGridView con click sugli elementi distinti per ruolo, e una checklist di produzione che copre accessibilità, licenze, caching, dipendenze esterne e coerenza visiva.

1. Il problema in produzione

Dopo sei moduli IconDesk sa disegnare un'icona in cinque modi diversi, e funzionano tutti. Quello che al progetto manca ancora è una regola di decisione: quando una nuova schermata ha bisogno di un simbolo Save, nessuno dovrebbe stare a indovinare se arriva dal tema, da un pacchetto ufficiale Wisej-4 o da IconDesk.Icons. Questo modulo aggiunge l'ultima opzione, i font di icone disegnati come HTML, e poi scrive la policy che dice a quale ricorrere e che cosa controllare prima che una schermata vada in produzione.

2. Concetti chiave

Concetto	Perché è importante
font di icone	Un web font i cui glifi sono immagini invece che lettere, scelti con nomi di classe CSS e disegnati dal browser dentro il contenuto HTML invece che tramite una qualsiasi proprietà immagine.
AllowHtml	L'interruttore per singolo controllo che dice a Wisej.NET di interpretare il testo di un controllo come markup; senza di esso i tag compaiono come caratteri letterali, perché per impostazione predefinita il testo è testo semplice.
head di Default.html	La pagina host dell'applicazione, dove un unico link a un foglio di stile rende il CSS e i file del font di icone disponibili a ogni schermata che la sessione apre.
click sull'elemento distinto per ruolo	Un click dentro il contenuto HTML può riferire quale elemento è stato colpito, così che una sola cella della griglia offra un bersaglio Edit e un bersaglio Delete senza due controlli separati.
CDN contro hosting proprio	Puntare il link a una CDN pubblica è una decisione di produzione su disponibilità, privacy, Content Security Policy e licenze; le applicazioni interne ospitano spesso i file da sé.
policy delle icone a livelli	Una regola scritta che dà a ogni tipo di simbolo una casa sola: immagini del tema per i significati, un pacchetto principale per i comandi, il pacchetto personalizzato per la grafica di prodotto, file raster per le fotografie.
giro di QA delle icone	Un passaggio sulle schermate finite con un tema chiaro e uno scuro a più di una scala di visualizzazione, verificando che ogni icona sia ancora visibile e abbia ancora un'alternativa testuale.

3. Il filo della lezione

Caricare il foglio di stile e accendere il markup

Un font di icone non fa parte in alcun modo del sistema di immagini di Wisej.NET. Non si assegna nulla a `Image`, non si scrive nulla in `ImageSource` e il server non produce mai un PNG. È il browser a scaricare un foglio di stile e un file del font, e un nome di classe su un elemento inline dice al font quale glifo disegnare. Questo fa del caricamento una questione della pagina e non del controllo: il link al foglio di stile va nell'head del `Default.html` dell'applicazione, dove viene scaricato una volta sola per la sessione e vale per ogni finestra e ogni dialogo che l'utente apre.

Dove punta quel link è una decisione di produzione. Una CDN pubblica è comoda, ma l'interfaccia dipende allora dalla raggiungibilità di terzi, rivela a quell'host le richieste dei tuoi utenti, deve superare la tua Content Security Policy ed è regolata dalla licenza della libreria di icone. Per un'applicazione interna, servire da te il CSS e i file del font è spesso la scelta migliore: disponibilità e privacy dipendono allora soltanto dalla tua distribuzione.

Caricato il foglio di stile, non compare ancora nulla. I controlli Wisej.NET trattano `Text` come testo semplice, quindi il markup arriva con l'escape e l'utente vede le parentesi angolari. È impostare

`AllowHtml` a `true` a cambiare le cose, ed è deliberatamente una scelta per singolo controllo: Label, Button, celle della DataGridView, CheckBox, RadioButton e LinkLabel sono gli esempi capaci di contenuto HTML. Accendere l'interruttore sposta anche un confine di fiducia: da quel momento il testo del controllo è markup, quindi tienilo di proprietà dell'applicazione e metti le parole dell'utente in un controllo separato a testo semplice.

Glifi in una toolbar, in una card e in una cella di griglia

In IconDesk la prima superficie con font di icone è una piccola toolbar o una card di riepilogo. Ogni pulsante o etichetta riceve `AllowHtml = true` e un valore di testo che mette un elemento inline davanti a una parola: è qui che i font di icone risultano naturali, perché il glifo scorre insieme all'etichetta invece di stare in uno slot immagine separato, ed eredita dimensione e colore del font circostante. La stessa idea si estende a una `DataGridView` con `AllowHtml` attivo, dove una colonna di azioni compatta tiene due glifi in una sola cella invece di due controlli ospitati.

Una sola cella con due simboli funziona soltanto se l'applicazione sa dire quale dei due è stato cliccato. È a questo che serve un attributo `role` sull'elemento inline: il click riferisce l'elemento colpito e il gestore si dirama su quello. Ispeziona i dati dell'evento del mouse che la tua versione del framework espone davvero prima di farci affidamento, e non inventare un contratto che l'evento non offre.

```
labelActions.AllowHtml = true;
labelActions.Text =
    "<i class='fa-solid fa-pen' role='edit'></i> " +
    "<i class='fa-solid fa-trash' role='delete'></i>";

labelActions.MouseClick += labelActions_MouseClick;
```

La compattezza ha un costo. Una cella piena di markup è più difficile da raggiungere con la tastiera di quanto lo sia un pulsante vero, gli screen reader ricevono un glifo senza nome e un'icona che non si carica lascia una colonna vuota. Dai a ogni azione un tooltip o una parola visibile, e ricorri a una normale colonna immagine o a un controllo ospitato quando l'interazione è più di una scorciatoia a due icone.

I font di icone accanto a SVG e pacchetti di icone

Nessuno dei due meccanismi vince a mani basse. Il sistema proprio di Wisej.NET è basato su risorse e SVG, ed è lì che dà il meglio: una stringa di sorgente può indicare un'immagine del tema, un percorso `resource.wx` o un asset di un pacchetto; gli SVG monocromatici accettano il suffisso `?color=` con il nome di un colore del tema; l'Image Selector del designer sfoglia ogni pacchetto installato; il caching è costruito attorno a quelle risorse; e qualsiasi controllo con una proprietà immagine può usarle direttamente.

I font di icone si guadagnano il loro posto altrove. Sono familiari a chi sviluppa per il web, si inseriscono in un design system HTML e CSS già esistente, i loro cataloghi sono enormi e stanno inline con il testo in un modo che una proprietà immagine non può. Se il linguaggio visivo dell'azienda è già espresso come classi di font, allinearsi costa un link a un foglio di stile invece che ridisegnare ogni asset.

La risposta onesta, quindi, è che convivono, e la scelta appartiene alla superficie più che al progetto. I comandi su controlli con proprietà immagine appartengono alla pipeline delle risorse; i simboli inline

dentro il testo o in una cella di griglia densa sono un caso ragionevole per il font di icone. Ciò che non è accettabile è la deriva, in cui lo stesso concetto Delete è un SVG di un pacchetto su una schermata e un glifo del font su quella dopo.

La checklist di produzione e la policy di IconDesk

Il vero deliverable del capstone è una pagina di policy. Una regola a livelli funziona bene: immagini del tema per i significati comuni, un unico pacchetto ufficiale Wisej-4 principale per i comandi generali, il pacchetto personalizzato IconDesk.Icons per la grafica proprietaria, file raster compressi per le fotografie e i font di icone solo dove l'integrazione HTML ne vale la pena. Poiché la policy indica una prima scelta per ogni situazione, una nuova schermata è coerente per impostazione invece che per revisione.

Attorno a questo sta la checklist che il progetto esegue prima che una schermata vada in produzione:

- Ogni icona che porta un significato ha una parola visibile, un tooltip o un'altra alternativa testuale, e lo stato non viene mai segnalato con il solo colore.
- Ogni URL esterno o riferimento a una CDN è approvato in modo esplicito, e i termini di licenza della libreria sono registrati.
- Il markup del font di icone è di proprietà dell'applicazione oppure è codificato; i valori non attendibili non raggiungono mai un controllo con `AllowHtml`.
- Gli URL degli asset sono stabili così che la cache del browser possa lavorare, e nessuna query string casuale la vanifica.
- La schermata è stata aperta con un tema chiaro e uno scuro e a una scala di visualizzazione più alta, e i risultati sono annotati.

È in quel giro di QA che i cinque meccanismi si trovano finalmente uno accanto all'altro. Nella pagina di riepilogo di IconDesk compaiono tutte insieme un'immagine del tema, un'icona di un pacchetto ufficiale, un'icona del pacchetto personalizzato, un asset incorporato raggiunto tramite `resource.wx` e un SVG ricolorato, e cambiare tema mostra quali l'hanno seguito e quali sono rimaste esattamente come disegnate. Saper spiegare, per ogni immagine visibile, dove è conservata e quale API l'ha risolta è l'abilità verso cui questo corso ti ha portato.



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:

- ✓ Caricare un foglio di stile di font di icone e disegnare icone nei controlli Wisej.NET con AllowHtml.
- ✓ Usare i font di icone nel contenuto di Label, Button e DataGridView e capire i click sugli elementi distinti per ruolo.
- ✓ Confrontare i font di icone con le risorse SVG e dei pacchetti di icone di Wisej.NET.
- ✓ Applicare una checklist di produzione che copre accessibilità, sicurezza, caching, dipendenze esterne e coerenza.

Guida al lab / esame

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



Lab pratico

Completa il capstone di IconDesk. Referenzia un foglio di stile di font di icone da `Default.html`, imposta `AllowHtml` su una piccola toolbar o su una card e disegna le sue icone come glifi del font, poi costruisci una colonna di `DataGridView` le cui celle contengono azioni realizzate con il font di icone e leggi quale elemento ha cliccato l'utente, così che `Edit` e `Delete` facciano cose diverse. Costruisci una pagina di riepilogo che dimostra tutti insieme i meccanismi visti nel corso: un'immagine del tema, un'icona di un pacchetto ufficiale, un'icona del tuo pacchetto, un logo incorporato e un SVG ricolorato. Poi scrivi la policy delle icone in produzione del progetto ed esegui un giro di QA con un tema chiaro e uno scuro.

Punto di partenza

Apri la soluzione IconDesk come l'ha lasciata il Modulo 6: il pannello dei comandi, le pagine ImageLab e IconGallery, due pacchetti di icone ufficiali Wisej-4, la cartella Resources incorporata servita tramite `resource.wx` e la class library IconDesk.Icons con il suo catalogo `AppIcons`. Nel progetto non c'è ancora nulla che usi un font di icone, e il `Default.html` del progetto ha ancora un head vuoto a parte il titolo.

Approccio suggerito

- ✓ Decidi per prima cosa se il font di icone viene referenziato da una CDN o ospitato accanto all'applicazione, e registra le ragioni di disponibilità, privacy, Content Security Policy e licenza di quella scelta prima di aggiungere il link a `Default.html`.
- ✓ Aggiungi una piccola toolbar o card, imposta `AllowHtml` su ogni Label o Button, e conferma che un controllo senza l'interruttore mostra il markup come testo letterale, così da aver visto entrambi i comportamenti.
- ✓ Attiva `AllowHtml` sulla griglia, costruisci la colonna di azioni con due elementi inline che portano ruoli distinti, poi ispeziona i dati dell'evento del mouse che la tua versione di Wisej.NET espone e dirama `Edit` e `Delete` sull'elemento che è stato colpito.
- ✓ Costruisci la pagina di riepilogo così che ciascuno dei cinque meccanismi compaia una volta, etichettato con il posto in cui l'immagine è conservata e l'API che la risolve, e assegna una sola proprietà immagine per controllo.
- ✓ Esegui il giro di QA: passa da un tema chiaro a uno scuro, prova una scala di visualizzazione più alta, controlla i tooltip e la raggiungibilità da tastiera sulle azioni con la sola icona, e annota tutto ciò che è cambiato o è sparito.

Deliverable richiesti

- ✓ Foglio di stile di font di icone referenziato da `Default.html` e una toolbar o card con `AllowHtml` che disegna glifi del font
- ✓ Colonna di `DataGridView` le cui celle mostrano azioni con font di icone e rispondono all'elemento cliccato

- ✓ Pagina di riepilogo di IconDesk che dimostra un'immagine del tema, un'icona di un pacchetto ufficiale, un'icona del pacchetto personalizzato, un asset incorporato e un SVG ricolorato
- ✓ Policy scritta delle icone in produzione che copre accessibilità, licenze, caching e dipendenze esterne
- ✓ Giro di QA eseguito con un tema chiaro e uno scuro, con i risultati annotati

Domande di autovalutazione

- ✓ Il link al foglio di stile è in `Default.html` e i nomi di classe sono giusti, ma una Label mostra i tag grezzi mentre il Button accanto mostra il glifo. Che cosa differisce tra i due controlli, e perché l'impostazione è per singolo controllo invece che per applicazione?
- ✓ La tua colonna di azioni funziona, e un collega chiede di mettere nella stessa cella il testo della nota scritta dal cliente, così da poterlo formattare. Che cosa gli rispondi, e che cosa costruiresti al suo posto?
- ✓ Nella pagina di riepilogo, quali delle cinque immagini cambiano quando l'applicazione cambia tema e quali restano esattamente come disegnate? Che cosa ti dice questo su dove ciascuna è conservata?



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.