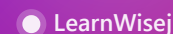


I pacchetti di icone di Wisej.NET e NuGet

Guida della lezione · Guida al lab / esame



Guida della lezione

Che cos'è davvero un pacchetto di icone: un assembly .NET che porta centinaia di file SVG come risorse incorporate, indirizzabili uno a uno tramite resource.wx e messi in cache dal browser un'icona alla volta, installabile dai package NuGet ufficiali Wisej-4, sfogliabile nell'Image Explorer di Visual Studio e richiamabile da C# attraverso il catalogo di icone del pacchetto. 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

Che cos'è davvero un pacchetto di icone: un assembly .NET che porta centinaia di file SVG come risorse incorporate, indirizzabili uno a uno tramite resource.wx e messi in cache dal browser un'icona alla volta, installabile dai package NuGet ufficiali Wisej-4, sfogliabile nell'Image Explorer di Visual Studio e richiamabile da C# attraverso il catalogo di icone del pacchetto.

1. Il problema in produzione

Dopo tre moduli IconDesk sa già mostrare un'immagine in cinque modi diversi, ma ogni icona che possiede è stata disegnata o scaricata a mano. Un'applicazione vera ha bisogno di un'icona Save, una Delete, una Search, una User e una Settings fin dal primo giorno, e di altre quaranta entro fine trimestre, tutte nello stesso linguaggio visivo. Copiare i file SVG uno alla volta produce una cartella di grafica sconsigliata, spessori di tratto incoerenti e nessuna traccia delle licenze.

2. Concetti chiave

Concetto	Perché è importante
pacchetto di icone	Un normale assembly .NET i cui file delle icone, per lo più SVG, sono compilati dentro come risorse incorporate; referenziarlo aggiunge un solo file alla tua distribuzione, per quante icone contenga.
risorsa incorporata	Un file compilato dentro l'assembly invece che copiato accanto a esso, così le icone non possono essere perse, rinominate o dimenticate quando l'applicazione viene distribuita su un altro server.
resource.wx	L'handler di Wisej.NET che serve una singola risorsa incorporata via HTTP; la sorgente immagine di un pacchetto di icone è un URL <code>resource.wx</code> che indica l'assembly del pacchetto e il file al suo interno.
catalogo delle icone	La classe statica che un pacchetto pubblica perché chi la usa abbia IntelliSense e un nome verificato dal compilatore invece di un percorso scritto a mano; il namespace e il nome esatto della classe variano da pacchetto a pacchetto.
richiesta per singola icona	Le icone dei pacchetti vengono richieste una a una, al primo utilizzo, e poi servite dalla cache del browser: è questo che rende sostenibile un catalogo da migliaia di icone.
precaricamento delle immagini del tema	Le immagini del tema viaggiano con il tema e sono già nella cache del tema sul client quando l'applicazione parte, quindi all'avvio si comportano in modo molto diverso dalle icone dei pacchetti.
famiglia di icone principale	L'unico pacchetto su cui un prodotto si impegna per i suoi comandi di ogni giorno, scelto di proposito per copertura, stile e licenza invece che accumulato pacchetto dopo pacchetto.

3. Il filo della lezione

Un pacchetto di icone è un assembly, non una cartella

La prima cosa da disimparare è l'idea che un pacchetto di icone sia uno zip di immagini. Un pacchetto di icone di Wisej.NET è un normale assembly .NET. Le sue icone, nei pacchetti ufficiali in stragrande maggioranza SVG, sono compilate dentro come risorse incorporate, quindi un pacchetto che contiene diverse migliaia di icone arriva comunque nella tua cartella `bin` come un'unica DLL. Nulla viene copiato accanto a `Default.html`, nulla deve essere pubblicato con il sito e nulla può essere cancellato per sbaglio durante una distribuzione.

Ogni icona resta comunque indirizzabile singolarmente. A runtime il browser chiede un'icona alla volta tramite l'handler `resource.wx`, che legge quella singola risorsa incorporata dall'assembly referenziato e la restituisce. La sorgente immagine che assegna è quindi solo una stringa: il nome dell'handler, l'assembly in cui vive la risorsa e il file al suo interno. Trattandosi di una normale risposta HTTP, il browser la mette in cache come qualsiasi altro file, così il secondo controllo che usa la stessa icona, e il caricamento di pagina successivo, non costano nulla.

È questa consegna icona per icona a rendere sostenibile un pacchetto da migliaia di icone: referenziare l'assembly non invia nulla al browser, usare un'icona invia un file. Un'immagine del tema indicata per nome funziona all'opposto: è precaricata con il tema prima che venga disegnata la prima schermata.

Installare i package Wisej-4 ufficiali

Wisej.NET pubblica le sue famiglie di icone come package NuGet e per un'applicazione Wisej.NET 4 ti serve la famiglia `Wisej-4-*`: `Wisej-4-FontAwesome`, `Wisej-4-MaterialDesign`, `Wisej-4-ModernUI`, `Wisej-4-ElegantIcons`, `Wisej-4-VisualStudioIcons`, `Wisej-4-BootstrapIcons`, `Wisej-4-VaadinIcons`, `Wisej-4-FeatherIcons`, `Wisej-4-CSSGIcons`, `Wisej-4-TablerIcons` e `Wisej-4-IonicIcons`. Wisej.NET 4.1 porta in gioco anche Lucide; verifica come viene distribuito sul profilo NuGet ufficiale di Wisej e nelle note di rilascio della 4.1, invece di installare un package di terze parti non correlato che per caso porta lo stesso nome.

Aggiungili come aggiungi qualsiasi dipendenza, dalla UI del NuGet Package Manager di Visual Studio o con `dotnet add package Wisej-4-BootstrapIcons`. Tieni la linea major del pacchetto allineata a quella del framework: un'applicazione Wisej 4 usa pacchetti Wisej-4, e gli esempi copiati da un vecchio package Wisej 2 o Wisej 3 vanno riverificati contro quello attuale invece che incollati così come sono. Nel lab di IconDesk referenzierai due di questi package, il che significa esattamente due voci `PackageReference` nel file di progetto e due assembly in più nella storia delle dipendenze sotto controllo di versione.

C'è un passo che frega quasi tutti: compila dopo aver aggiunto il riferimento. L'Image explorer in fase di design scopre gli assembly dei pacchetti di icone che il progetto referencia davvero e contro cui ha compilato, quindi un selettore aperto prima della prima compilazione semplicemente non elencherà la nuova famiglia.

Dall'Image explorer a C#

Una volta installati i package, l'Image Selector di Visual Studio guadagna le nuove famiglie accanto alle immagini del progetto e a quelle del tema, e lì sfogli le icone e ne vedi l'anteprima invece di tirare a indovinare i nomi dei file. Quando ne scegli una, il designer scrive per te la sorgente immagine dentro il `Designer.cs` generato, e quella stringa generata vale la pena leggerla almeno una volta perché mostra l'intero meccanismo in una riga.

```
// generato dal designer: il selettore ha scritto per te l'URL resource.wx
buttonSave.ImageSource =
    "resource.wx/Wisej.Ext.FontAwesome/android.svg?color=Coral";

// da C#: il catalogo di icone del pacchetto, così non si scrive a mano alcun percorso
// (il namespace e la classe esatti del catalogo variano da pacchetto a pacchetto)
buttonSave.ImageSource = FontAwesome.Icons.Save;

// ricolora un SVG monocromatico del pacchetto con il nome di un colore del tema
buttonDelete.ImageSource =
    "resource.wx/Wisej.Ext.BootstrapIcons/trash.svg?color=error";
```

Entrambe le forme arrivano allo stesso punto. Nel codice applicativo la forma da preferire è il catalogo: dà IntelliSense, sopravvive a una rinomina dentro il pacchetto ed evita che i nomi di file letterali si spargano nei gestori di eventi, dove nessuno può controllarli. Il suffisso `?color=` funziona sugli SVG dei pacchetti

esattamente come funzionava sui tuoi SVG monocromatici nel modulo precedente, e funziona per lo stesso motivo, quindi una grafica piena o multicolore lo ignorerà. È un fatto da scoprire nel lab su due icone vere, non una supposizione da portarsi in produzione.

Pacchetto, immagine del tema o pacchetto tuo

Ora sono tre i meccanismi che si contendono lo stesso pulsante. Usa un'**immagine del tema indicata per nome** quando l'icona rappresenta una semantica standard del framework e deve cambiare quando cambia il tema. Usa un **pacchetto di icone pubblicato** quando ti serve un'ampia copertura dei concetti applicativi ordinari. Costruisci un **pacchetto di icone dell'applicazione**, che è il Modulo 6, quando i simboli sono proprietari, legati al marchio o un sottoinsieme curato che vuoi versionare tu. I meccanismi convivono senza problemi; ciò che rovina un prodotto è usarli tutti e tre a caso per lo stesso tipo di comando.

Scegliere la famiglia principale è una revisione breve, a cui si può rispondere:

- Il pacchetto copre i concetti di dominio che il prodotto nomina davvero?
- I suoi stili a contorno o pieni stanno bene accanto al tema?
- Le icone sono SVG monocromatici che accettano la ricolorazione?
- La licenza a monte è compatibile con il modo in cui l'applicazione viene distribuita?
- Designer e sviluppatori riconosceranno la convenzione di denominazione?

Infine, tratta il pacchetto per la dipendenza che è. È un assembly che distribuisce, quindi registrane l'origine e la licenza, tieni il `PackageReference` sotto controllo di versione e testa un cambio di versione prima della produzione. Non installare ogni famiglia solo perché installare è facile. Il lab si chiude con una raccomandazione di un paragrafo proprio per questo motivo: le prove del pannello di rete, una richiesta per icona e poi una risposta dalla cache, sono ciò che la rende difendibile.



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 che cos'è un pacchetto di icone di Wisej.NET e come il browser ne ottiene le icone.
- ✓ Installare e usare i package NuGet ufficiali dei pacchetti di icone Wisej-4.
- ✓ Usare le risorse dei pacchetti di icone nel designer di Visual Studio e da C#.
- ✓ Confrontare le risorse dei pacchetti di icone con le immagini del tema e scegliere il meccanismo giusto per un'applicazione.

Guida al lab / esame

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



Lab pratico

Aggiungi a IconDesk due package NuGet ufficiali di pacchetti di icone Wisej-4 e costruisci una pagina IconCompare che mostra gli stessi cinque concetti — Save, Delete, Search, User e Settings — presi da entrambe le famiglie, così che la differenza visiva sia impossibile da ignorare. Scegli almeno cinque icone attraverso l'Image explorer di Visual Studio invece di scrivere il percorso, assegna le restanti dal catalogo del pacchetto in C#, ricolorane due con il suffisso `?color=` e osserva nel pannello di rete i file delle icone che arrivano una richiesta alla volta e poi arrivano dalla cache del browser. Chiudi con una breve raccomandazione che indica la famiglia di icone principale del progetto.

Punto di partenza

Apri la soluzione IconDesk così come l'hai lasciata dopo il Modulo 3: il pannello dei comandi, la pagina ImageLab e la pagina IconGallery funzionano tutti, e ogni immagine del progetto è un file locale, un'immagine del tema oppure un URL che hai scritto tu. Il progetto non ha ancora alcun riferimento a un pacchetto di icone, quindi l'Image Selector del designer al momento mostra solo le immagini del progetto e quelle del tema.

Approccio suggerito

- ✓ Scegli due famiglie ufficiali Wisej-4 dallo stile contrastante, aggiungile entrambe con la UI del NuGet Package Manager o con `dotnet add package` e compila il progetto prima di aprire qualsiasi selettore.
- ✓ Aggiungi una pagina `IconCompare` con due colonne etichettate e una riga per concetto: Save, Delete, Search, User e Settings.
- ✓ Riempi la prima colonna dall'Image explorer di Visual Studio, poi apri `IconCompare.Designer.cs` e leggi le stringhe `resource.wx` generate dal selettore.
- ✓ Riempi la seconda colonna in C# dal catalogo di icone di ciascun pacchetto e ricolora due icone aggiungendo il suffisso `?color=` con il nome di un colore del tema.
- ✓ Ricarica con il pannello di rete del browser aperto, registra le richieste per singola icona e le risposte servite dalla cache al secondo caricamento, poi scrivi la raccomandazione partendo da quelle prove.

Deliverable richiesti

- ✓ Due package NuGet ufficiali di pacchetti di icone Wisej-4 referenziati da IconDesk
- ✓ Pagina IconCompare che mostra Save, Delete, Search, User e Settings da entrambi i pacchetti
- ✓ Almeno cinque icone scelte tramite l'Image explorer di Visual Studio invece che scritte a mano
- ✓ Due icone ricolorate con il suffisso di colore e una nota su quale grafica ha accettato la ricolorazione

- ✓ Raccomandazione di un paragrafo che indica la famiglia di icone principale del progetto, con le prove raccolte dalla cache del browser

Domande di autovalutazione

- ✓ Referenziare un pacchetto da tremila icone aggiunge un solo assembly alla distribuzione. Che cosa costa al browser, e quando?
- ✓ Una delle tue due icone ricolorate ha ignorato il suffisso di colore. Che cosa c'è in quella grafica, e che cosa cambieresti per questo nei criteri di scelta dei pacchetti?
- ✓ Un collega vuole che ogni icona di comando segua il tema quando l'applicazione cambia tema. Quali delle cinque icone di IconCompare dovrebbero diventare immagini del tema, e quali no?



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.