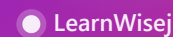


CSS, CssClass, CssStyle, stati e cambi di tema a runtime

Guida della lezione · Guida al lab / esame



Guida della lezione

Aggiungere CSS mirato in modo sicuro con `CssClass` e un foglio di stile applicativo circoscritto, riservare `CssStyle` a valori dinamici una tantum, usare `AppearanceKey` e stati personalizzati per le varianti semantiche, e cambiare tema all'avvio o per una singola sessione a runtime. Che cosa significa per uno sviluppatore Wisej.NET — e come affrontare questo modulo.



Focus di produzione

Aggiungere CSS mirato in modo sicuro con `CssClass` e un foglio di stile applicativo circoscritto, riservare `CssStyle` a valori dinamici una tantum, usare `AppearanceKey` e stati personalizzati per le varianti semantiche, e cambiare tema all'avvio o per una singola sessione a runtime.

1. Il problema in produzione

Il tema `AdaptiveOps` del Modulo 2 ora possiede l'identità di ogni controllo standard di `Adaptive Operations Console`, e le richieste che arrivano dopo sono tutte piccole: una card delle metriche ha bisogno di una superficie arrotondata e rialzata, un badge SLA deve essere compatto, una barra di avanzamento ha bisogno di una larghezza calcolata da dati in tempo reale, il comando `Assign` dovrebbe sembrare l'azione principale, e il turno di notte vuole una variante scura. Ogni richiesta ha una scorciatoia allettante — una stringa `CssStyle` qui, un `BackColor` là, il tema condiviso modificato sul posto — e ogni scorciatoia sposta in silenzio una decisione visiva a un livello che non la possiede. Sei mesi dopo nessuno sa più dire che cosa visualizza il tema, che cosa scavalca il CSS e che cosa rattoppa il codice. Questo modulo mette ciascuna di queste cinque richieste nel meccanismo che la possiede.

2. Concetti chiave

Concetto	Perché è importante
CssClass	Uno o più nomi di classe aggiunti all'elemento principale del widget nel browser: l'aggancio riutilizzabile a cui un foglio di stile applicativo circoscritto può puntare senza toccare il tema.
CssStyle	Una stringa CSS applicata direttamente come attributo style dell'elemento principale del widget; giusta per un singolo valore calcolato, sbagliata come strumento di stile generale perché gli stili inline sono difficili da ispezionare, da gestire con il tema o da passare a un designer.
foglio di stile applicativo circoscritto	Un file CSS caricato accanto al tema i cui selettori partono da nomi di classe che possiedi tu, come <code>.metric-card</code> , e non entrano mai nel DOM interno di un widget Wisej standard.
vecchio extender StyleSheet	Un componente più vecchio che aggiunge classi CSS ai controlli idonei e ospita le regole nella sua proprietà <code>StyleSheetSource</code> ; un aiuto per la migrazione, inutile nel nuovo codice perché ogni controllo ora ha <code>CssClass</code> e <code>CssStyle</code> .
AppearanceKey	Il nome di un'appearance del tema attivo con cui viene visualizzato un controllo, così il C# esprime l'intenzione (<code>action-button</code>) e il tema possiede colori, font e metriche di ogni stato.
stati personalizzati	Nomi aggiunti alla collezione <code>States</code> di un controllo a cui il tema può dare stile come a <code>hovered</code> o <code>focused</code> , per condizioni transitorie della UI come <code>stale</code> , <code>warning</code> o <code>dirty</code> che un designer dovrebbe poter vedere e ridisegnare.
tema di avvio e <code>Application.Theme</code>	Il tema a livello di applicazione si sceglie prima che l'app parta, in <code>Web.config</code> o <code>Default.json</code> ; <code>Application.Theme</code> permette a una sessione di lavorare con un oggetto tema a runtime senza cambiare ciò che vedono le altre sessioni.

3. Il filo della lezione

Quattro strumenti, una regola ciascuno

Ogni controllo Wisej.NET espone due agganci lato client. `CssClass` accetta uno o più nomi di classe che vengono aggiunti all'elemento principale del widget; `CssStyle` accetta una stringa CSS che viene applicata direttamente come attributo `style` di quell'elemento. Stanno uno accanto all'altro nella griglia delle proprietà e in una codebase si comportano in modo molto diverso. Una classe è un nome che puoi cercare, riutilizzare su venti controlli e ridisegnare in un solo file; uno stile inline è un override anonimo che vince sulla maggior parte di ciò che genera il tema e resta invisibile finché qualcuno non apre l'ispettore del browser.

La regola è breve. Usa `CssClass` ogni volta che vuoi un aggancio di stile riutilizzabile supportato da un file CSS dell'applicazione. Usa `CssStyle` solo per stili una tantum molto piccoli o per valori calcolati dinamicamente: una larghezza che segue una percentuale, un colore che arriva con i dati. L'abuso di `CssStyle` rende la console difficile da ispezionare, da gestire con il tema e da mantenere per un designer,

ed è per questo che il lab consente esattamente un valore `CssStyle` calcolato e ti chiede di annotare perché non poteva essere una classe.

Due meccanismi più vecchi compaiono ancora nei progetti reali. L'extender `StyleSheet` aggiunge classi CSS ai controlli idonei e ospita le regole nella sua proprietà `StyleSheetSource`; dato che ogni controllo moderno ha già `CssClass` e `CssStyle`, è un aiuto per la migrazione, non qualcosa da introdurre nel nuovo codice. La sezione `Stylesheet` del Theme Builder porta regole CSS grezze dentro il tema; esiste per i widget di terze parti, l'HTML personalizzato e le superfici di integrazione senza una normale appearance Wisej, e sui widget standard le regole generate da appearance e stati possono scavalcarla.

Classi circoscritte per la UI propria della console

Il lab aggiunge ad Adaptive Operations Console `AdaptiveOps.css`, un foglio di stile che l'applicazione carica accanto al tema. Il suo compito è limitato: dare stile alle parti della console che sono tue, non di Wisej. Una classe `.metric-card` dà ai pannelli KPI raggio, padding ed elevazione, `.metric-card .metric-title` imposta il piccolo titolo in maiuscolo, e `.compact-badge` trasforma una label in una pillola per gli indicatori di SLA e priorità. Ogni selettore parte da una classe che possiedi tu, quindi il foglio di stile non può ridisegnare per sbaglio un pulsante o l'intestazione di una griglia altrove.

Ciò che resta fuori da quel file conta quanto ciò che ci entra. Lo stile dei controlli standard — i colori di hover e pressed dei pulsanti, i font delle intestazioni delle griglie, il bordo dell'editor non valido — appartiene alle appearance del tema del Modulo 2. Rattopparli dal CSS significa selettori fragili che dipendono dal DOM interno di un widget e si rompono in silenzio al prossimo aggiornamento del framework o del tema. Il CSS dei widget di terze parti segue la stessa disciplina: circoscritto al contenitore del widget e documentato.

Prima di rilasciare una classe, verifica che cosa può vanificarla. Un controllo con `BackColor`, `ForeColor` o `Font` impostati direttamente nel designer scavalca sia il tema sia la tua classe; un `CssStyle` dimenticato scavalca tutto. Quando una classe "non funziona", la checklist è: la classe è sull'elemento, c'è una proprietà impostata sul controllo, c'è uno stile inline impostato, e la regola è circoscritta al contenitore giusto.

Varianti semantiche: AppearanceKey e stati personalizzati

Impostare `BackColor`, `ForeColor` e `Font` sul pulsante Save lo fa sembrare principale oggi e intoccabile domani. `AppearanceKey` è l'alternativa: il controllo viene visualizzato con un'apparence con nome del tema attivo, e il tema possiede i dettagli visivi di default, hovered, pressed, disabled, focused e invalid. L'apparence `action-button` del Modulo 2 eredita da `button` e ridefinisce solo tre stati; assegnarla al comando principale esprime l'intenzione di business in C# e lascia i pixel al tema. Lo stesso schema si estende a `destructive-button`, `quiet-panel`, `kpi-card` o `sidebar-button` ogni volta che il concetto di business è riutilizzabile.

Gli stati personalizzati coprono le condizioni transitorie che una chiave di appearance non può esprimere. Ogni controllo ha una collezione `States`; aggiungere un nome come `stale`, `warning`, `dirty` o `compact` permette al tema di dare stile a quello stato esattamente come fa con hovered o focused. Il lab aggiunge uno stato `stale` alle righe della griglia dei ticket o al pannello di stato quando un ticket non è stato aggiornato in tempo, e il tema decide che aspetto ha `stale`. Uno stato è una condizione della UI che il tema può rappresentare visivamente, non un posto in cui tenere i dati applicativi, e il codice non dovrebbe mai rileggere uno stato per prendere una decisione.

```
// MainPage.cs – ogni livello possiede la propria decisione
this.cardOpen.CssClass = "metric-card"; // CSS applicativo circoscritto
this.lblPriorityBadge.CssClass = "compact-badge";
this.btnSave.AppearanceKey = "action-button"; // gli stati li possiede il tema

// l'unico valore calcolato: clip-path, non width – il motore di layout riscrive
// una width inline a ogni passata di layout e la cancellerebbe al ridimensionamento
// successivo
this.barFill.CssStyle = $"clip-path:inset(0 {100 - slaPercent}% 0 0)";

if (ticket.IsStale)
    row.States.Add("stale"); // stato personalizzato gestito dal
// tema
else
    row.States.Remove("stale");
```

Temi di avvio, temi di sessione e il tema condiviso

Un tema viene applicato prima che l'applicazione parta indicandolo in Web.config o in Default.json. Cambiare il tema mostrato nel designer di Visual Studio non cambia il tema a runtime; la console gira con AdaptiveOps solo perché lo dice la configurazione. I file di tema con nome sono la casa giusta per il branding stabile: versionati, revisionati e identici per ogni utente.

Wisej.NET espone anche `Application.Theme`, tramite cui una sessione può lavorare con un oggetto tema a runtime. È lì che vive il cambio chiaro/scuro del lab: l'interruttore nella toolbar carica la variante scura, la assegna alla sessione corrente e ricorda la scelta in `Application.Session`. Il pericolo sottile è che un oggetto tema può essere modificato sul posto. Modificare il tema condiviso da tutte le sessioni perché un utente ha chiesto la modalità scura cambia la console per l'intero turno di notte; lo schema sicuro è un file di tema separato con un nome, o una copia per sessione, mai una modifica dell'oggetto condiviso.

La regola di decisione: file di tema con nome per il branding stabile, cambi di tema a runtime per personalizzazione, accenti per tenant, demo e anteprime controllate. Dimostra il confine con due sessioni del browser affiancate, una scura e una chiara, prima di considerare concluso il deliverable.



Errori tipici

Stringhe `CssStyle` copiate su ogni pulsante finché nessuno sa più che cosa visualizza il tema. Selettori che entrano nel DOM interno di un widget standard. `BackColor`, `ForeColor` o `Font` impostati nel designer che scavalcano in silenzio il tema e la classe. Usare `States` come archivio di dati. Un interruttore per la modalità scura che modifica il tema condiviso per ogni sessione.



La regola del sistema visivo

Usa il meccanismo Wisej.NET che possiede il problema: il tema per l'identità dei controlli standard, `AppearanceKey` per le varianti semantiche, `CssClass` per il CSS circoscritto e specifico dell'applicazione, `CssStyle` solo per valori dinamici una tantum, i contenitori di layout invece del codice di ridimensionamento, le proprietà responsive per i valori di profilo impostati nel designer, e `ResponsiveProfileChanged` per il comportamento che non è semplicemente visivo.



Obiettivi di apprendimento

Alla fine di questo modulo sarai in grado di:

- ✓ Usare correttamente `CssClass`, `CssStyle`, il CSS personalizzato e gli schemi del vecchio extender `StyleSheet`.
- ✓ Definire classi circoscritte per la UI specifica dell'applicazione mantenendo nei temi lo stile dei controlli standard.
- ✓ Usare `AppearanceKey` e gli stati personalizzati per creare varianti semantiche dei controlli.
- ✓ Applicare i temi di avvio e capire la differenza tra i cambi di tema specifici di una sessione e i cambi di tema globali.

Guida al lab / esame

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



Lab pratico

Aggiungi alla Adaptive Operations Console un foglio di stile circoscritto `AdaptiveOps.css` con classi per le card delle metriche e i badge compatti, applicale tramite `CssClass` e usa `CssStyle` solo per un valore calcolato, come la larghezza di una barra di avanzamento. Assegna l'`AppearanceKey` `action-button` al comando principale, aggiungi uno stato personalizzato stale alle righe della griglia dei ticket o a un pannello di stato, e implementa nella toolbar un cambio di tema chiaro/scuro che cambia il tema solo per la sessione corrente, senza modificare il tema condiviso.

Punto di partenza

Apri la Adaptive Operations Console del Modulo 2: lo shell in dock con i pannelli toolbar, navigazione, area di lavoro, dettagli e stato, il tema `AdaptiveOps` selezionato in `Web.config` o `Default.json`, i suoi token con nome e l'appearance `action-button`. L'area di lavoro contiene già quattro pannelli delle metriche, una barra di avanzamento SLA e la griglia dei ticket, ma hanno ancora le appearance predefinite o valori improvvisati di `BackColor`, `ForeColor` e `CssStyle`, e la toolbar non ha un cambio di tema.

Approccio suggerito

- ✓ Crea `AdaptiveOps.css`, caricalo con l'applicazione accanto al tema e scrivi solo selettori ancorati a classi che possiedi tu: `.metric-card`, `.metric-card .metric-title` e `.compact-badge`. Applicale tramite `CssClass` ai quattro pannelli delle metriche e alle label di SLA e priorità.
- ✓ Trova l'unico valore che deve essere calcolato a runtime, la larghezza di avanzamento SLA, impostalo tramite `CssStyle` in un solo metodo e scrivi un commento che spiega perché una classe non poteva esprimerlo. Rimuovi ogni altro `CssStyle` e ogni override di `BackColor`, `ForeColor` o `Font` rimasto su quei controlli.
- ✓ Assegna `AppearanceKey = "action-button"` al comando principale (`btnSave`) e verifica che l'aspetto hovered, pressed e disabled arrivi dal tema senza alcun codice.
- ✓ Nel Theme Builder aggiungi uno stato `stale` all'appearance della riga del ticket o del pannello di stato, con lo stile basato sul token warning, poi aggiungi e rimuovi `"stale"` tramite `States` quando l'ultimo aggiornamento di un ticket supera la soglia. Tieni la soglia nei dati e l'aspetto nel tema.
- ✓ Aggiungi alla toolbar un interruttore del tema che carica un secondo file di tema con nome, `AdaptiveOps-Dark`, lo assegna ad `Application.Theme` per la sessione corrente e ricorda la scelta in `Application.Session`. Verifica con due sessioni del browser che il tema condiviso resti intatto.

Deliverable richiesti

- ✓ `AdaptiveOps.css` circoscritto con le classi `metric-card` e `compact-badge` applicate tramite `CssClass`

- ✓ `CssStyle` usato per esattamente un valore calcolato una tantum, con il motivo documentato
- ✓ Comando principale che usa l'`AppearanceKey` action-button
- ✓ Stato personalizzato stale aggiunto tramite `States` e con lo stile definito nel tema
- ✓ Cambio di tema chiaro/scuro valido solo per la sessione che lascia intatto il tema condiviso

Domande di revisione per lo studente

- ✓ Un designer chiede un raggio degli angoli più piccolo su ogni card delle metriche. Quale file cambia, e in quanti punti avresti dovuto intervenire se il raggio fosse stato impostato con `CssStyle` ?
- ✓ Perché il pulsante Assign non ha bisogno di codice per hover o pressed, e che cosa perderesti esattamente sostituendo il suo `AppearanceKey` con valori di `BackColor` e `ForeColor` che corrispondono alla palette?
- ✓ Due persone hanno effettuato l'accesso e una delle due passa alla modalità scura. Descrivi che cosa vede ciascuna, e indica la riga del tuo codice che lo garantisce.



La regola del sistema visivo

Usa il meccanismo Wisej.NET che possiede il problema: il tema per l'identità dei controlli standard, `AppearanceKey` per le varianti semantiche, `CssClass` per il CSS circoscritto e specifico dell'applicazione, `CssStyle` solo per valori dinamici una tantum, i contenitori di layout invece del codice di ridimensionamento, le proprietà responsive per i valori di profilo impostati nel designer, e `ResponsiveProfileChanged` per il comportamento che non è semplicemente visivo.