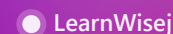


ClientProfile e proprietà responsive

Guida della lezione · Guida al lab / esame



Guida della lezione

Come ClientProfiles.json viene confrontato dall'alto verso il basso per determinare Application.ActiveProfile, i valori delle proprietà responsive assegnati nel designer per ogni profilo, ResponsiveProfileChanged per il comportamento che le proprietà responsive non possono esprimere, e le regole per progettare e testare i profili. Che cosa significa per uno sviluppatore Wisej.NET — e come affrontare questo modulo.



Focus di produzione

Come ClientProfiles.json viene confrontato dall'alto verso il basso per determinare Application.ActiveProfile, i valori delle proprietà responsive assegnati nel designer per ogni profilo, ResponsiveProfileChanged per il comportamento che le proprietà responsive non possono esprimere, e le regole per progettare e testare i profili.

1. Il problema in produzione

Adaptive Operations Console ha ora uno shell con tema costruito con contenitori Flow, Table e Flex che regge un ridimensionamento senza un gestore Resize, ma mostra a tutti la stessa schermata: su un telefono da 390 pixel la barra di navigazione, la griglia dei ticket e l'editor dei dettagli si contendono la stessa larghezza. Le media query CSS non possono risolverlo, perché i controlli vivono sul server: nascondere un pannello, passare un pulsante a solo icona o rimettere in dock una regione è una modifica di proprietà lato server, e un utente su telefono che perde il pannello dei dettagli ha bisogno di un altro modo per modificare un ticket, non di una copia rimpicciolita di quello desktop. Questo modulo aggiunge il meccanismo che Wisej.NET possiede per questo: profili client, proprietà responsive e l'evento `ResponsiveProfileChanged`.

2. Concetti chiave

Concetto	Perché è importante
ClientProfile	Un insieme con nome di regole di corrispondenza legate al dispositivo (larghezza del browser, larghezza dello schermo, tipo di dispositivo, user agent, orientamento orizzontale) che descrive come viene usata una classe di client, non il nome commerciale di un dispositivo.
ClientProfiles.json	Il file che Wisej.NET legge; i profili vengono valutati dall'alto verso il basso e vince il primo che corrisponde, quindi l'ordine delle voci fa parte della configurazione.
Application.ActiveProfile	Il profilo attualmente selezionato per questa sessione; il suo <code>Name</code> è ciò che mostra la label di stato e ciò su cui si ramifica il gestore del profilo.
proprietà responsive	Una normale proprietà .NET marcata con <code>ResponsivePropertyAttribute</code> che contiene un valore per ogni profilo client e applica il valore corrispondente ogni volta che cambia il profilo attivo.
adattamento guidato dal designer	I valori responsive si assegnano solo nel designer di Visual Studio, il che li mantiene serializzati, visibili e deterministici; il codice serve per gli adattamenti che dipendono dai dati o dallo stato a runtime.
ResponsiveProfileChanged	L'evento sollevato su <code>Application</code> , su contenitori come Form, Page, Desktop e UserControl e su DataGridView e ListView quando cambia il profilo attivo; <code>ResponsiveProfileChangedEventArgs.CurrentProfile</code> porta il nuovo profilo.
matrice di QA responsive	Il piano di test che elenca larghezza, altezza, orientamento, tipo di dispositivo, tocco, barre di scorrimento, scala dei font e il nome del profilo che ti aspetti a ogni dimensione.

3. Il filo della lezione

Come viene individuato un profilo

Ogni voce di `ClientProfiles.json` ha un `name` e qualsiasi combinazione di `device` (Mobile, Tablet o Desktop), `landscape`, `minWidth` e `maxWidth` per il viewport del browser, `minScreenWidth` e `maxScreenWidth` per lo schermo fisico, e una corrispondenza su `userAgent`. Wisej.NET legge il file, valuta i profili dall'alto verso il basso e seleziona la prima corrispondenza come `Application.ActiveProfile`. Nulla viene unito: un client corrisponde a un profilo o a nessuno, ed è per questo che lo schema qui sotto ripiega su "Default" quando `ActiveProfile` è null.

La corrispondenza può avvenire al caricamento della pagina o al ridimensionamento del browser, a seconda della proprietà usata da una regola: una regola sulla larghezza del viewport viene rivalutata mentre si trascina il browser, mentre dimensione dello schermo, tipo di dispositivo e user agent descrivono il client stesso. Ogni volta che il profilo selezionato cambia, Wisej.NET solleva `ResponsiveProfileChanged`. Per il lab scrivi sei profili in ordine di prima corrispondenza: Phone, Phone Landscape, Tablet, Tablet Landscape, Small Desktop e Desktop. Le due voci del telefono condividono `device` Mobile e differiscono

solo per `Landscape` ; Small Desktop è un dispositivo Desktop con `maxWidth` 1024; Desktop è la regola generale in fondo. Mettila in cima e nasconde ogni profilo sotto di sé, e nulla ti avvisa.

Le proprietà responsive appartengono al designer

Le proprietà responsive sono proprietà ordinarie marcate con `ResponsivePropertyAttribute` : `Visible` , `Display` , `Size` , `Location` , `Dock` , `Anchor` , `Font` e i valori legati al layout dove il controllo li supporta. Nel designer una proprietà di questo tipo contiene un valore diverso per ogni profilo di `ClientProfiles.json` , e quando il profilo attivo cambia il controllo applica il valore che gli appartiene. Il designer è l'unico modo supportato per assegnare questi valori, ed è una regola, non una preferenza: i valori assegnati lì vengono serializzati con il form, sono visibili al prossimo sviluppatore e deterministici a runtime, mentre i valori impostati nel codice sono invisibili a design time e si allontanano dall'immagine che il designer dà della schermata.

Nel lab ogni differenza deterministica della console è lavoro del designer. Su entrambi i profili Phone lo UserControl della barra di navigazione e il pannello dei dettagli ricevono `Visible` false. I pulsanti della toolbar ricevono un `Display` solo icona su Phone e Tablet così che la barra dei comandi stia su una riga. Sui profili Tablet il `Dock` della regione dei dettagli passa da Right a Bottom così che stia sotto la griglia dei ticket invece che accanto. Nessuna di queste modifiche dipende dai dati, quindi nessuna appartiene al codice; il codice serve per gli adattamenti che dipendono da dati, permessi, un calcolo a runtime o l'evento del profilo stesso.

ResponsiveProfileChanged per il comportamento

Alcuni adattamenti non sono valori. Su un telefono l'editor dei dettagli non può proprio essere un pannello laterale; deve aprirsi come `Form` modale tramite `ShowDialog` , e la label di stato dovrebbe mostrare e registrare nel log il nome del profilo così che il supporto sappia quale layout vede un utente. È a questo che serve `ResponsiveProfileChanged` . Scatta su `Application` , sui tipi ContainerControl come Form, Page, Desktop e UserControl, e su alcuni controlli complessi come DataGridView e ListView. Gestiscilo su `Application` per il comportamento a livello di applicazione e su un contenitore chiave quando la logica è locale a quella regione; gli argomenti dell'evento espongono `CurrentProfile` .

Lo schema qui sotto è la forma che il lab si aspetta: sottoscriviti una volta, applica il profilo una volta all'avvio partendo da `Application.ActiveProfile` così che il layout iniziale sia corretto, fai passare l'evento dallo stesso metodo `ApplyProfile` , e annulla la sottoscrizione in `Dispose` perché `Application` sopravvive alla pagina. Tieni `ApplyProfile` leggero e idempotente: un telefono che ruota tra Phone e Phone Landscape lo attiva ripetutamente, e un editor già aperto deve essere riutilizzato, mai aperto di nuovo. Il lavoro tipico nell'evento: caricare meno colonne della griglia, cambiare l'intervallo di debounce della ricerca, trasformare un pannello dei dettagli in un form modale, applicare un modello di dati diverso.

```
// MainPage.cs – un unico metodo applica il profilo all'avvio e a ogni cambio
Application.ResponsiveProfileChanged += Application_ResponsiveProfileChanged;
ApplyProfile(Application.ActiveProfile?.Name);

private void Application_ResponsiveProfileChanged(object sender,
ResponsiveProfileChangedEventArgs e)
{
    ApplyProfile(e.CurrentProfile?.Name);
}

private void ApplyProfile(string profileName)
{
    var phone = profileName != null && profileName.StartsWith("Phone");
    this.lblProfile.Text = profileName ?? "Default"; // mostrato e registrato nel
log
    if (phone && this.ticketEditor == null) OpenDetailsAsModal(); // pannello laterale →
modale, una volta sola
}
```

Varianti di design che cambiano il comportamento

I profili dovrebbero descrivere il comportamento, non i dispositivi. Phone, Tablet Landscape, Small Desktop, Kiosk e Touch Workstation indicano ciascuno un modo di usare la console, e ogni variante dovrebbe cambiare ciò che la console fa: il telefono ottiene un editor modale e un pulsante menu, il tablet impila la regione dei dettagli sotto la griglia, il small desktop mantiene lo shell con una toolbar solo icona, il desktop mostra tutto. Usa regole di larghezza per il layout, regole sul dispositivo per le ipotesi sull'input come le aree di tocco e le barre di scorrimento nascoste, `userAgent` solo quando è necessario, e regole landscape solo quando l'orientamento cambia davvero la UI. Un profilo che non cambia nient'altro che una dimensione CSS non aveva bisogno di esistere.

Testa in entrambi i modi. Il ridimensionamento reale del browser mette alla prova le regole di larghezza; l'emulazione del dispositivo mette alla prova le regole su dispositivo, larghezza dello schermo e user agent, e dopo aver cambiato dispositivo nell'emulatore mobile di Chrome devi ricaricare la pagina così che quei profili vengano rivalutati. La matrice di QA responsive ha una riga per scenario, da Desktop a 1440 per 900 fino a Phone a 390 per 844 e Phone Landscape a 844 per 390, ciascuna con il nome del profilo atteso e i controlli che dimostrano il comportamento: navigazione visibile, dettagli impilati, editor modale raggiungibile, nessun overflow orizzontale.

- Profili stretti o specifici prima di quelli generici; vince la prima corrispondenza e nulla viene unito.
- Valori responsive nel designer; `ResponsiveProfileChanged` solo quando un valore non basta, e allora leggero e idempotente.
- Mostra il nome del profilo nelle build di test e documenta ogni controllo nascosto con la sua alternativa mobile.



Errori tipici

Elencare per primo il profilo generico Desktop così che i profili telefono e tablet non vengano mai individuati. Assegnare i valori responsive nel codice invece che nel designer. Mettere ogni adattamento nel gestore dell'evento, compresi quelli che un valore di `Visible` o `Dock` avrebbe espresso. Riaprire l'editor modale a ogni rotazione perché il gestore non è idempotente. Nascondere un pannello sul telefono senza un'alternativa mobile per ciò che conteneva.



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:

- ✓ Spiegare come ClientProfiles.json viene letto e confrontato e come viene selezionato Application.ActiveProfile.
- ✓ Usare le proprietà responsive nel designer per proprietà come Visible, Display, Size, Location, Dock, Anchor, Font e i valori legati al layout quando disponibili.
- ✓ Gestire ResponsiveProfileChanged su Application e sui contenitori chiave quando serve una logica specifica per profilo.
- ✓ Progettare varianti per telefono, tablet, small desktop e desktop che cambiano il comportamento della UI, non solo le dimensioni CSS.

Guida al lab / esame

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



Lab pratico

Aggiungi ad Adaptive Operations Console un `ClientProfiles.json` con i profili Phone, Phone Landscape, Tablet, Tablet Landscape, Small Desktop e Desktop ordinati dal più stretto al più largo. Nel designer, usa le proprietà responsive per nascondere la barra di navigazione e il pannello dei dettagli sul telefono, passare i pulsanti della toolbar a un Display solo icona e rimettere in dock la regione dei dettagli sotto la griglia sul tablet. Gestisci `ResponsiveProfileChanged` per mostrare il nome del profilo attivo nella label di stato, registrare ogni cambio nel log e aprire l'editor dei dettagli come form modale sul telefono invece che come pannello laterale.

Punto di partenza

Apri Adaptive Operations Console come l'ha lasciata il Modulo 5: lo shell è fatto di contenitori annidati con un'area Flow per le card delle metriche, un editor dei ticket con `TableLayoutPanel` dentro lo `UserControl` dei dettagli e un'area di lavoro Flex, il tema `AdaptiveOps` e il foglio di stile circoscritto sono caricati, e la label di stato del Modulo 1 riporta ancora la larghezza del browser. Non esiste ancora un

`ClientProfiles.json`, quindi `Application.ActiveProfile` è null per ogni client e ogni larghezza mostra lo shell desktop.

Approccio suggerito

- ✓ Scrivi `ClientProfiles.json` nella radice dell'applicazione con sei voci in ordine di prima corrispondenza (Phone, Phone Landscape, Tablet, Tablet Landscape, Small Desktop, Desktop): le due voci del telefono usano `device` Mobile e differiscono per `landscape`, la coppia del tablet fa lo stesso con Tablet, Small Desktop è un dispositivo Desktop con `maxWidth` 1024 e Desktop è la regola generale per ultima. Esegui una volta e verifica che i nomi dei profili compaiano nel designer.
- ✓ Nel designer, dai allo `UserControl` della barra di navigazione e al pannello dei dettagli un valore di `Visible` false per entrambi i profili Phone, e controlla che la griglia dell'area di lavoro abbia ancora il suo `MinimumSize` così che il layout del telefono non collassi a zero.
- ✓ Sempre nel designer, imposta il `Display` dei pulsanti della toolbar su solo icona per i profili Phone e Tablet, e cambia il `Dock` della regione dei dettagli da Right a Bottom per entrambi i profili Tablet così che stia sotto la griglia dei ticket; lascia intatti i valori desktop.
- ✓ In `MainPage`, sottoscriviti una sola volta ad `Application.ResponsiveProfileChanged`, scrivi un metodo `ApplyProfile(string profileName)` che imposta `lblProfile.Text` e scrive una riga di log per ogni cambio, chiamalo nel costruttore con `Application.ActiveProfile?.Name` e dal gestore con `e.CurrentProfile?.Name`, e annulla la sottoscrizione in `Dispose`.
- ✓ Per i profili Phone, fai in modo che `ApplyProfile` apra l'editor dei ticket in un `Form` modale tramite `ShowDialog`, riutilizzando il form se è già aperto; poi prova trascinando il browser e con l'emulazione del dispositivo, ricaricando dopo ogni cambio di dispositivo, e cattura la label del profilo a ogni dimensione.

Deliverable richiesti

- ✓ ClientProfiles.json con i profili Phone, Phone Landscape, Tablet, Tablet Landscape, Small Desktop e Desktop in ordine di prima corrispondenza
- ✓ Valori delle proprietà responsive assegnati nel designer per nascondere la barra di navigazione e il pannello dei dettagli sul telefono
- ✓ Display solo icona della toolbar e regione dei dettagli rimessa in dock per il tablet, impostati tramite proprietà responsive
- ✓ Gestore ResponsiveProfileChanged che mostra e registra nel log il nome del profilo attivo
- ✓ Editor dei dettagli aperto come form modale sul telefono tramite il gestore del profilo

Domande di revisione per lo studente

- ✓ Se Small Desktop venisse spostato in cima a ClientProfiles.json, quali degli altri cinque profili potrebbero ancora essere individuati, e che cosa vedrebbe un tablet in orizzontale?
- ✓ Il nuovo docking del tablet vive nel designer e l'editor modale del telefono vive nel codice. Che cosa decide quale meccanismo possiede ciascuna modifica, e che cosa andrebbe storto se li scambiassi?
- ✓ Un tester ha ruotato un telefono quattro volte e il tuo gestore è stato eseguito quattro volte. Quale stato deve essere identico dopo la quarta esecuzione e dopo la prima, e come lo garantisce il tuo codice?



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.