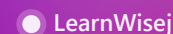


Grafici, dashboard, contenuti, media e documenti

Guida della lezione · Guida al lab / esame



Guida della lezione

DataSet, etichette e opzioni di ChartJS; gauge e ProgressBar; PdfViewer, HtmlPanel, Upload e GoogleMaps tenendo presente il confine browser/server; dashboard aggiornate come un unico modello. Che cosa significa per uno sviluppatore Wisej.NET — e come affrontare questo modulo.



Focus di produzione

DataSet, etichette e opzioni di ChartJS; gauge e ProgressBar; PdfViewer, HtmlPanel, Upload e GoogleMaps tenendo presente il confine browser/server; dashboard aggiornate come un unico modello.

1. Il problema in produzione

Le dashboard non sono solo grafici. Una dashboard è una composizione di controlli che trasforma i dati in decisioni, e il tab dashboard della Operations Console sarà la schermata che un responsabile del servizio apre per prima ogni mattina. Wisej.NET ti dà controlli nativi per contenuti e media più estensioni per grafici, mappe, visualizzazione di documenti e gauge; l'abilità intermedia sta nel capire che cosa appartiene al modello a controlli di Wisej.NET, che cosa conviene delegare a una libreria JavaScript specializzata incapsulata come estensione o Widget, e come far aggiornare l'intera schermata come un'unica unità invece che come un mucchio di elementi visuali che si ricaricano ciascuno per conto proprio.

2. Concetti chiave

Concetto	Perché è importante
dashboard	Una composizione di controlli il cui unico compito è rispondere a una domanda che porta a una decisione; se un elemento visivo non aiuta qualcuno a decidere qualcosa, è rumore.
estensione ChartJS	Un pacchetto NuGet che espone Chart.js come componente lato server: il server possiede lo stato del grafico, il browser ne rende l'aspetto, e il grafico entra nello stesso modello a eventi di tutti gli altri controlli.
etichette e DataSet	Le due metà dei dati di un grafico; le etichette danno un nome ai punti lungo l'asse e ogni DataSet porta una serie, quindi si impostano sempre insieme.
view model della dashboard	Un oggetto modellato per la visualizzazione, come <code>DashboardModel</code> , che contiene mesi, conteggi e un valore di completamento, separato sia da qualsiasi entità del database sia dal codice che lo disegna.
confine del browser	Il client è il browser e il server è dove gira il tuo codice Wisej.NET; file, PDF, HTML e tile delle mappe attraversano quella linea in una direzione o nell'altra.
Upload contro percorso locale	Upload sposta un file dal browser al server; non è la stessa cosa che leggere un percorso sul desktop, perché il server non può mai sfogliare il disco dell'utente.
refresh basato sul modello	Una sola chiamata al servizio restituisce un modello completo della dashboard e un solo metodo aggiorna tutti i controlli a partire da quello, in blocco, con un orario visibile.

3. Il filo della lezione

Parti dalla domanda, poi scegli il controllo

Ogni controllo del tab dashboard deve corrispondere alla domanda che l'utente si sta ponendo. Un grafico mostra un andamento: usa un grafico a linee per una tendenza nel tempo, un grafico a barre per confrontare categorie, e uno scatter o un bubble chart per la relazione tra due valori. Riserva i grafici a torta e a ciambella a una piccola relazione parte-tutto; con molte categorie gli spicchi diventano illeggibili e il grafico smette di rispondere a qualsiasi cosa.

Non usare un grafico perché fa bella figura. Quando conta l'informazione esatta a livello di record, la `DataGridView` è il controllo migliore, e la griglia Orders del Modulo 5 esiste già per questo. Quando all'utente serve un singolo valore di stato, per esempio la percentuale di completamento dei ticket del mese, una `ProgressBar` o un indicatore tipo gauge è più chiaro di qualsiasi grafico. Quando la domanda è geografica, risponde una mappa. Le regole di scelta sono brevi: grafici per gli andamenti, griglie per i record, gauge e barre di avanzamento per i valori singoli, mappe per la geografia.

ChartJS: opzioni, etichette e DataSet sul server

ChartJS è la scelta abituale per grafici a linee, a barre, a barre orizzontali, radar, polar area, torta, ciambella, bubble e scatter. Aggiungi l'estensione a `OperationsConsole` tramite NuGet e poi la trascini sul tab come qualsiasi altro componente. Il grafico si configura attraverso le sue opzioni, le sue `Labels` e i suoi `DataSets`; la distinzione importante è che il server possiede lo stato del componente mentre il browser ne

rende l'aspetto, ed è per questo che un clic sul grafico può sollevare un normale evento lato server e che un grafico può essere ricostruito da C# senza una riga di JavaScript.

I dati del grafico vanno tenuti separati dal suo rendering. Crea un view model come `DashboardModel` con i mesi, i conteggi di aperti e chiusi e la percentuale di completamento, e convertilo in etichette e DataSet in un unico punto. Non spargere la costruzione del grafico tra gestori di click scollegati. Le etichette fanno parte dei dati, non sono decorazione opzionale: quando mancano, il grafico può non mostrare i punti come ti aspetti, quindi il lab imposta etichette e DataSet insieme, nello stesso metodo. Configura assi, tooltip, legende, valori minimi e massimi e colori in modo consapevole, invece di accettare quello che producono i valori predefiniti.

```
private void RefreshDashboard(DashboardModel model)
{
    this.chartTickets.Labels = model.Months.ToArray();
    this.chartTickets.DataSets.Clear();
    var opened = this.chartTickets.DataSets.Add("Aperti");
    opened.Data = model.Opened.Cast<object>().ToArray();
    var closed = this.chartTickets.DataSets.Add("Chiusi");
    closed.Data = model.Closed.Cast<object>().ToArray();

    this.progressCompletion.Value = model.CompletionPercent;
    this.pdfPreview.PdfSource = model.PreviewDocumentUrl;
    this.lblLastRefreshed.Text = $"Ultimo aggiornamento {DateTime.Now:HH:mm:ss}";
}
```

Controlli di contenuto e media al confine del browser

I controlli di contenuto risolvono problemi di contenuto. `PdfViewer` mostra un documento PDF nel browser usando le opzioni di visualizzazione supportate. `HtmlPanel` incorpora un documento HTML completo, mentre `WebBrowser` e `IFramePanel` incorporano contenuti esterni del browser. `Upload` sposta i file dal browser al server. `PictureBox` mostra i formati immagine, `ImageList` fornisce immagini condivise a liste e alberi, e `Canvas`, `Shape` e `Line` coprono il disegno personalizzato e gli elementi visivi semplici. Usali tutti con un modello chiaro del confine del browser in testa: il client è il browser, il server è dove gira il codice Wisej.NET, e ognuno di questi controlli esiste per far attraversare quella linea a un contenuto in sicurezza.

Due errori di confine compaiono in quasi ogni prima dashboard. Il primo è trattare Upload come se il server potesse sfogliare il disco locale dell'utente; il server vede soltanto i byte che il browser ha scelto di inviare, quindi il flusso è: seleziona, carica, poi elabora sul server. Il secondo è incorporare HTML arbitrario in un `HtmlPanel` senza sanificarlo o limitarlo, cosa che trasforma un'anteprima in una superficie di injection. Questi visualizzatori sono confini di contenuto, non un modo per evitare di progettare l'applicazione.

`GoogleMaps` è un buon esempio di estensione specializzata. Si controlla come un componente server, supporta tipi di mappa, marker, opzioni, forme, percorsi ed eventi come i clic sulla mappa o il trascinamento dei marker, e richiede una chiave API di Google Maps configurata sul componente. È esattamente la promessa di integrazione di Wisej.NET: una capacità esterna del browser incapsulata in un

componente lato server che partecipa ai normali eventi dell'applicazione. Nel lab basta un segnaposto per la mappa o per il contenuto; il punto è riservare lo spazio e documentare il confine.

Un solo refresh, un solo modello, niente rumore

Le dashboard hanno bisogno di regole di performance. Evita di aggiornare ogni controllo visivo per conto proprio, perché cinque controlli con cinque timer producono cinque andate e ritorno e cinque momenti in cui la schermata è in disaccordo con sé stessa. Costruisci un servizio di refresh che restituisce un modello completo della dashboard, poi aggiorna i controlli in blocco da un unico metodo come

`RefreshDashboard`. Non caricare un enorme insieme di dati per il grafico quando un aggregato risponde alla domanda: il servizio deve restituire dodici totali mensili, non la tabella grezza dei ticket.

Se la dashboard si aggiorna in tempo reale, aggiorna il numero minimo di controlli e mostra un orario, così l'utente sa quanto sono freschi i numeri. Usa stati di caricamento e disabilita le azioni non utilizzabili mentre è in corso un aggiornamento. L'abitudine che tiene insieme il modulo è una checklist breve, applicabile a qualsiasi schermata di dashboard:

- Parti dalla domanda a cui l'utente deve rispondere, e tieni i dati del grafico in un view model separato dal codice di rendering.
- Imposta insieme etichette e DataSet del grafico; usa PdfViewer, HtmlPanel, WebBrowser e IFramePanel per i confini di contenuto; usa i pattern Upload e Download per il trasferimento di file dal browser.
- Aggiorna la dashboard come un modello, non come controlli scollegati, e marca il risultato con l'orario dell'aggiornamento.



Errori tipici

Grafici a torta con molte categorie; dati del grafico aggiornati senza le etichette; Upload trattato come un modo per leggere il disco locale dell'utente; HTML arbitrario incorporato senza sanificarlo o limitarlo; e dashboard belle da vedere che però non rispondono a nessuna domanda che porti a una decisione.



La regola da libreria di controlli

Scegli il controllo nativo più semplice che esprime il compito dell'utente prima di scrivere codice personalizzato. I controlli nativi ti danno già supporto a design time, temi, eventi e stato sul server; il passo successivo è un extender o uno UserControl, e solo quando nessun controllo nativo è adatto si costruisce un widget JavaScript.



Obiettivi di apprendimento

Alla fine di questo modulo sarai in grado di:

- ✓ Scegliere un controllo grafico, gauge, di avanzamento, documento, media o contenuto in base al bisogno informativo dell'utente.
- ✓ Usare DataSet, etichette, opzioni e tipi di grafico di ChartJS per dashboard gestionali.

- ✓ Usare PdfViewer, HtmlPanel, PictureBox, Upload, ProgressBar, Canvas e GoogleMaps tenendo presenti i confini browser/server.
- ✓ Progettare dashboard che si aggiornano dai dati del server senza diventare rumore visivo.

Guida al lab / esame

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



Lab pratico

Costruisci il tab dashboard di Operations Console: un grafico ChartJS a linee o a barre le cui etichette e DataSet arrivano da un view model della dashboard, una ProgressBar o un indicatore tipo gauge per un valore di stato, un'anteprima di contenuto con PdfViewer o HtmlPanel, un flusso di Upload, una mappa o un segnaposto di contenuto, e un unico metodo di refresh che aggiorna tutti i controlli della dashboard a partire da un solo modello.

Punto di partenza

Apri la soluzione `OperationsConsole` che ti sei portato dietro dai Moduli 1 a 5 e seleziona la pagina Dashboard dello shell a TabControl; è ancora il segnaposto del Modulo 3. Tieni il lavoro di questo modulo in un namespace o in una cartella dedicati, per esempio `OperationsConsole.Dashboard`, così che il capstone possa riutilizzarlo, e aggiungi il pacchetto dell'estensione ChartJS tramite NuGet prima di aprire il designer.

Approccio suggerito

- ✓ Scrivi la domanda a cui la dashboard risponde ("come sta andando il flusso dei ticket, e quanto siamo vicini all'obiettivo del mese?"), poi definisci un record `DashboardModel` con mesi, conteggi di aperti e chiusi, una percentuale di completamento e un riferimento al documento di anteprima, e un `DashboardService` che lo restituisce sotto forma di aggregati.
- ✓ Inserisci nel tab un controllo ChartJS chiamato `chartTickets`, scegli linee o barre per la tendenza e imposta deliberatamente assi, legenda e colori; riempi `Labels` e `DataSets` insieme a partire dal modello, mai da un gestore di click.
- ✓ Aggiungi una `ProgressBar` o un indicatore tipo gauge chiamato `progressCompletion` per il singolo valore di completamento, e un `PdfViewer` o un `HtmlPanel` chiamato `pdfPreview` o `htmlPreview` per l'anteprima del contenuto, limitando qualsiasi HTML che incorpori.
- ✓ Aggiungi un controllo `Upload` con i tipi di file ammessi e un limite di dimensione, gestisci sul server il suo evento di caricamento completato, salva il file tramite un servizio e poi aggiorna l'anteprima con il risultato; riserva un pannello segnaposto per la mappa o per il contenuto e annota in un commento che serve la chiave API di GoogleMaps.
- ✓ Scrivi un unico metodo `RefreshDashboard(DashboardModel)` che aggiorna il grafico, l'indicatore, l'anteprima e un'etichetta `lblLastRefreshed`, chiamato da un `btnRefresh_Click` sottile che mostra uno stato di caricamento, disabilita il pulsante di aggiornamento e gestisce il percorso di errore.

Deliverable richiesti

- ✓ Tab dashboard con un grafico ChartJS a linee o a barre
- ✓ Etichette e DataSet del grafico costruiti da un view model della dashboard

- ✓ ProgressBar o indicatore tipo gauge per un valore di stato
- ✓ Anteprima di contenuto con PdfViewer o HtmlPanel più un flusso di Upload
- ✓ Un unico metodo di refresh che aggiorna tutti i controlli della dashboard

Domande di autovalutazione

- ✓ Quale controllo del tuo tab dashboard fa il lavoro più importante, e quale domanda resterebbe senza risposta se lo togliessi?
- ✓ Che cosa si romperebbe per primo se l'insieme di dati dei ticket diventasse dieci volte più grande, e perché restituire aggregati dal servizio protegge il grafico?
- ✓ Dove si trova il file caricato in ciascun passo del flusso di Upload, e perché il server non può mai aprire un percorso digitato dall'utente?



La regola da libreria di controlli

Scegli il controllo nativo più semplice che esprime il compito dell'utente prima di scrivere codice personalizzato. I controlli nativi ti danno già supporto a design time, temi, eventi e stato sul server; il passo successivo è un extender o uno UserControl, e solo quando nessun controllo nativo è adatto si costruisce un widget JavaScript.