

Controlli personalizzati, estensioni, wrapper di widget e componenti riutilizzabili

Guida della lezione · Guida al lab / esame

● LearnWisej

Guida della lezione

Controlli riutilizzabili, wrapper di widget di terze parti, proprietà ed eventi personalizzati, impacchettamento delle risorse. Che cosa significa per un team Wisej.NET avanzato — e come affrontare questo modulo.



Focus di produzione

Controlli riutilizzabili, wrapper di widget di terze parti, proprietà ed eventi personalizzati, impacchettamento delle risorse.

1. Il problema in produzione

I team avanzati hanno spesso bisogno di componenti enterprise riutilizzabili o di widget JavaScript di terze parti. La sfida è esporli come controlli Wisej.NET manutenibili senza perdere la produttività del designer né la sicurezza lato server.

2. Concetti chiave

Concetto	Perché è importante
UserControl personalizzato	Un controllo composito costruito da controlli Wisej.NET esistenti che impacchetta layout e comportamento per riusarli tra le schermate.
controllo ereditato	Una sottoclasse di un controllo nativo che aggiunge valori predefiniti, validazione o comportamento mantenendo il pieno supporto a designer e temi.
wrapper di Widget	Un controllo che ospita una libreria JavaScript di terze parti dentro Wisej.NET. Dovrebbe esporre solo il valore lato browser, mentre le regole di business restano sul server.
risorse incorporate	Script, stili e immagini compilati nell'assembly del controllo, così il componente viene distribuito come un unico pacchetto senza file sparsi da rilasciare.
contratto degli eventi client/server	L'insieme concordato di eventi e forme dei payload scambiati tra il widget nel browser e il suo controllo sul server, documentato così entrambi i lati possono evolvere in sicurezza.
valori predefiniti per il designer	Valori di proprietà e attributi predefiniti sensati, così un controllo trascinato su un form funziona subito e appare coerente con il resto dell'applicazione.
API del componente	Le proprietà, i metodi e gli eventi pubblici che un controllo riutilizzabile espone. Un'API piccola e intenzionale è più facile da supportare di una che lascia trapelare dettagli di implementazione.

3. Il filo della lezione

Il riuso ha bisogno di un'API

Un controllo riutilizzabile dovrebbe essere più facile da usare correttamente che scorrettamente. Dagli proprietà, eventi, valori predefiniti e documentazione chiari. Non pretendere che ogni sviluppatore di schermate conosca il layout interno o i dettagli JavaScript.

Incapsula il comportamento, non solo l'aspetto

Un wrapper per un widget di terze parti dovrebbe tradurre chiamate C# comprensibili in termini di business in comportamento lato client, e tradurre gli eventi client in eventi server con un nome. Mantieni l'API piccola, stabile ed esplicita.

UserControl e controlli ereditati

Wisej.NET offre due modi per costruire un componente riutilizzabile a partire da controlli esistenti. Uno `UserControl` compone più controlli in un'unica superficie progettabile con il proprio layout, e si adatta a un componente come lo `StatusTimeline` del lab: un'intestazione, un elenco di voci di stato e un piè di pagina che insieme rappresentano lo storico di un ordine di lavoro. Un controllo ereditato estende un singolo controllo nativo, come una griglia che mostra sempre le colonne standard del tenant, e mantiene ogni funzionalità della sua base aggiungendo valori predefiniti e comportamento.

Entrambi si inseriscono dalla Toolbox ed entrambi compaiono nel Designer, ed è proprio questo il punto. Uno sviluppatore di schermate trascina uno `StatusTimeline` su un form, ne imposta le proprietà e ne

gestisce gli eventi senza sapere come si disegna. La prima domanda di revisione del lab, se un altro sviluppatore può usare il componente senza leggerne i dettagli interni, decide se il componente è finito.

Scegli la composizione quando il componente è una cosa nuova fatta di più parti, e l'ereditarietà quando è una versione specializzata di un controllo esistente. Mescolare le due produce componenti difficili da applicare ai temi e difficili da spiegare.

Un'API C# stabile attorno a un widget client

Il grafico del lab è un widget JavaScript di terze parti, e l'applicazione non gli parla mai direttamente. Una classe `WorkOrderChartWidget` lo incapsula, esponendo un metodo come `SetItems` che accetta un elenco di punti tipizzati e un evento server come `SegmentClicked` che trasporta l'identificativo di un segmento. Dentro il wrapper, il metodo trasforma l'elenco nella forma dei dati del fornitore e l'evento client viene tradotto dalla callback del fornitore nell'evento server con un nome.

La stabilità deriva dal mantenere l'API piccola ed espressa nei termini dell'applicazione. Un metodo chiamato secondo ciò che vuole l'applicazione, e non secondo la funzione del fornitore, sopravvive a un aggiornamento o a una sostituzione del fornitore. Il wrapper è anche il punto in cui si gestiscono i fallimenti: quando lo script del fornitore viene bloccato o non si carica, il wrapper visualizza un fallback semplice e non solleva errori nella schermata, ed è questo il fallback elegante del lab.

La seconda domanda di revisione, quali dati passano dal server al client, trova risposta nei DTO del wrapper e in nient'altro. Un wrapper che invia al grafico l'entità dell'ordine di lavoro ha fatto trapelare il dominio nel browser.

Impacchettare le risorse in modo coerente

Un componente riutilizzabile dipende da script, fogli di stile, immagini e talvolta font, e ognuno di questi deve arrivare insieme al componente ovunque venga usato. Il modo affidabile è incorporarli come risorse nell'assembly del componente e referenziarli tramite il meccanismo delle risorse del framework, così aggiungere l'assembly a un progetto aggiunge tutto ciò che serve al componente. I file sparsi copiati in ogni applicazione si disallineano nel giro di un rilascio.

Il pacchetto di risorse incorporate del lab segue una sola convenzione: una cartella per componente, nomi di file con versione per le librerie del fornitore e un unico punto che elenca che cosa viene caricato e in quale ordine. Le immagini usate dal componente sono, dove possibile, sensibili al tema, così un tema scuro non mostra un'icona chiara. Il foglio di stile del componente si carica dopo quello del fornitore, così le sue sostituzioni prevalgono.

- Il componente si può aggiungere a un nuovo progetto referenziando un solo assembly?
- La versione del fornitore è visibile nei nomi delle risorse e nelle note di rilascio?
- Aggiornare il fornitore cambia una sola cartella e un solo elenco?

Proprietà, eventi e valori predefiniti di design time

La superficie pubblica di un componente è ciò che gli altri sviluppatori vedono nella finestra Proprietà e in IntelliSense. Ogni proprietà ha un tipo che corrisponde al suo significato, un attributo di valore predefinito così il Designer sa che cosa non serializzare, una categoria e una descrizione. Ogni evento ha una classe di

argomenti tipizzata che trasporta esattamente ciò di cui un gestore ha bisogno, come il segmento cliccato, e un nome che dice che cosa è successo anziché quale callback del fornitore è scattata.

Il design time è uno scenario di primo piano. Il Designer crea il componente senza servizi e senza dati, quindi il componente deve rilevare la modalità design e visualizzare dati di esempio in sicurezza. La modalità di esempio del lab mostra nel Designer una timeline e un grafico plausibili, il che permette a uno sviluppatore di schermate di disporre il form senza eseguire l'applicazione, e non chiama mai un servizio né lancia eccezioni. La terza domanda di revisione, se il componente viene visualizzato in sicurezza nel Designer, si verifica trascinandolo su un form vuoto.

Documenta l'API sul componente stesso, nei commenti di riepilogo e nelle descrizioni, perché è l'unica documentazione che la maggior parte degli utenti del componente leggerà mai.



La regola di produzione avanzata

Non fermarti quando la schermata sembra corretta. Prima di considerare completo il modulo, spiega a chi appartiene lo stato, le implicazioni di sicurezza, i percorsi di errore e il comportamento in produzione.



Obiettivi di apprendimento

Alla fine di questo modulo sarai in grado di:

- ✓ Progettare UserControl personalizzati e controlli ereditati per il riuso.
- ✓ Incapsulare un widget client dietro un'API C# stabile.
- ✓ Impacchettare script, stili, immagini e risorse in modo coerente.
- ✓ Esporre proprietà, eventi e valori predefiniti di design time che altri sviluppatori possano usare in sicurezza.

Guida al lab / esame

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



Lab pratico

Costruisci un controllo StatusTimeline riutilizzabile e un Widget di grafico o visualizzazione incapsulato per lo storico degli ordini di lavoro. Esponi proprietà C#, eventi lato server, risorse incorporate e una modalità di esempio per il design time.

Approccio suggerito

- ✓ Leggi il problema in produzione e individua il rischio architetturale prima di toccare la UI.
- ✓ Crea o aggiorna per primo l'incapsulamento in un servizio, comando, modello o componente coinvolto.
- ✓ Costruisci la schermata o il componente Wisej.NET seguendo gli standard di progetto stabiliti.
- ✓ Aggiungi una dimostrazione del percorso di errore, non solo del percorso felice.
- ✓ Annota che cosa è cambiato nel registro di apprendimento e aggiorna la checklist relativa.

Deliverable richiesti

- ✓ UserControl StatusTimeline
- ✓ Wrapper di Widget o controllo personalizzato
- ✓ Pacchetto di risorse incorporate
- ✓ Documento del contratto degli eventi
- ✓ Schermata di esempio d'uso

Domande di revisione per lo studente

- ✓ Un altro sviluppatore può usare il componente senza leggerne i dettagli interni?
- ✓ Quali dati passano dal server al client?
- ✓ Il componente viene visualizzato in sicurezza nel designer?



La regola di produzione avanzata

Non fermarti quando la schermata sembra corretta. Prima di considerare completo il modulo, spiega a chi appartiene lo stato, le implicazioni di sicurezza, i percorsi di errore e il comportamento in produzione.