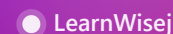


Controlli Wisej personalizzati, classi client e temi

Guida della lezione · Guida al lab / esame



Guida della lezione

Costruire controlli dall'aspetto nativo con classi server, classi widget client, appearance key e risorse incorporate. Che cosa significa per uno sviluppatore che integra con Wisej.NET — e come affrontare questo modulo.



Focus di produzione

Costruire controlli dall'aspetto nativo con classi server, classi widget client, appearance key e risorse incorporate.

1. Il problema di integrazione

Alcune integrazioni devono comportarsi come controlli Wisej.NET di prima classe: riutilizzabili, adatti ai temi, utilizzabili nel Designer e manutenibili in molte applicazioni.

2. Concetti chiave

Concetto	Come usarlo
Classe del controllo	Usa un controllo lato server quando serve un'integrazione dall'aspetto nativo.
AppearanceKey	Collega la resa grafica del controllo agli stili a livello di tema.

3. La lezione di produzione

Un controllo Wisej.NET personalizzato è la forma di integrazione più solida. Va usato quando il widget fa parte della piattaforma di prodotto, quando i temi contano o quando il controllo deve apparire e comportarsi come un controllo Wisej.NET nativo. Il pattern è simile alla creazione di un controllo nativo: una classe server espone proprietà ed eventi .NET, e una classe client riceve lo stato renderizzato e possiede l'oggetto lato browser. La classe client può estendere `wisej.web.Control` o `wisej.web.Widget` a seconda del comportamento desiderato. Può creare l'oggetto di terze parti, implementare i metodi `apply` delle proprietà, gestire i cambi di dimensione o di aspetto e tradurre le callback del fornitore in eventi Wisej.NET. La classe server può ridefinire `OnWebRender` per inviare solo le proprietà di cui il client ha bisogno. Questo approccio richiede più codice, ma offre l'architettura più pulita nel lungo periodo. È la scelta giusta per i controlli distribuiti come parte di un framework, di una libreria di estensioni o di un prodotto enterprise, dove il wrapper deve comparire in IntelliSense, funzionare nel Designer e restare stabile tra le versioni della libreria del fornitore sottostante.

Un controllo server per un'integrazione più stretta

Derivare da `Wisej.Web.Control` invece che da `Wisej.Web.Widget` cambia la natura della classe. Un Widget è un contenitore per uno script; un Control è un membro di prima classe del framework, con lo stesso comportamento di layout, focus, tema e Designer di un pulsante o di un pannello nativo. Il `SimpleGaugeControl` del lab segue questa strada, così che il gauge si possa trascinare, dimensionare, ancorare e personalizzare con i temi come qualunque altro controllo.

La classe server espone proprietà tipizzate ed eventi .NET, proprio come faceva il wrapper, ma non porta più con sé un `InitScript` né un contenitore `Options`. Dichiara invece quale classe client la implementa e invia il proprio stato tramite `OnWebRender`. I setter delle proprietà segnano il controllo come bisognoso di aggiornamento, e il framework decide quando inviare la modifica.

Scegli questa profondità quando il widget fa parte della piattaforma di prodotto, quando i temi contano o quando il controllo deve risultare nativo per chi ci costruisce le schermate. Il codice in più viene ripagato da ogni schermata che usa il controllo senza doverci pensare.

Una classe client sotto Platform

La metà browser di un controllo personalizzato è una classe JavaScript definita con il modello di classi client di Wisej.NET, basato su qooxdoo, e collocata nella cartella `/Platform` dell'applicazione, così che il framework la carichi insieme al client. La classe estende `wisej.web.Control`, oppure `wisej.web.Widget` quando si vuole il comportamento da contenitore di script, e dichiara le proprie proprietà con metodi `apply`.

Un metodo `apply` viene eseguito ogni volta che il server cambia la proprietà corrispondente, ed è lì che va la chiamata al fornitore: l'`apply` di `value` chiama il metodo del fornitore che imposta il valore, e l'`apply` di `maximum` aggiorna le opzioni del fornitore. Il costruttore della classe crea l'istanza del fornitore dentro l'elemento di contenuto del controllo, e il distruttore la elimina. I cambi di dimensione e di aspetto arrivano come eventi che la classe può gestire in un unico punto.

Tieni la classe client libera da logica applicativa. Traduce tra il framework e il fornitore, e nient'altro, così da poter essere riutilizzata da ogni controllo server che si appoggia a essa.

Inviare lo stato con OnWebRender

`OnWebRender` è l'occasione in cui il server decide esattamente che cosa vede il client. L'override riceve un oggetto config dinamico, e il controllo vi scrive solo le proprietà dichiarate dalla classe client:

`config.value`, `config.minimum`, `config.maximum`, `config.caption`. I nomi sono in camelCase per corrispondere ai nomi delle proprietà client, e Wisej.NET invia solo i valori cambiati dall'ultimo rendering.

È stato intenzionale, non stato accidentale. Una proprietà che esiste sul server a beneficio del Designer, o per la logica di business, semplicemente non viene scritta nel config e non raggiunge mai il browser. Un valore che sarebbe costoso da serializzare, come un grande array di storico, può essere ridotto a ciò che serve al gauge prima di essere assegnato. La domanda di autovalutazione del lab chiede qual è il ruolo di `OnWebRender`, e la risposta è che rappresenta il contratto tra le due metà del controllo.

Gli eventi viaggiano nella direzione opposta tramite `OnWebEvent`, trattato nel Modulo 7. Insieme, i due override definiscono l'intero protocollo di comunicazione del controllo.

Appearance key, temi e design time

Un tema Wisej.NET applica lo stile ai controlli in base all'appearance key. Assegnare al controllo gauge una `AppearanceKey` come `gauge`, e aggiungere una voce con quella chiave al tema o a un mixin del tema, collega colori, font e spaziature del controllo al tema dell'applicazione. Cambiando tema, il gauge viene restilizzato insieme a tutti gli altri controlli: è la differenza tra un controllo che sembra nativo e uno che sembra incollato.

Le risorse del tema non riguardano solo i colori. Immagini e icone di cui il controllo ha bisogno si possono dichiarare come risorse del tema, così che un tema scuro fornisca le varianti scure senza modifiche al codice. La classe client legge il proprio stile dall'appearance invece di codificarlo a mano, e l'istanza del fornitore viene configurata con quei valori nei metodi apply.

Il rendering a design time è la prova finale. Il Designer istanzia il controllo e lo visualizza dal vivo, quindi la classe client deve gestire dati segnaposto, l'assenza del server e creazioni ripetute mentre lo sviluppatore lo ridimensiona. Il lab chiede uno screenshot a design time proprio per questo: se il controllo si visualizza correttamente nel Designer verrà adottato; se lancia eccezioni verrà evitato.



Regola di produzione

Tieni ben distinti la proprietà dello stato in Wisej.NET e il comportamento del widget client. Documenta il contratto client/server prima di affidarti all'integrazione in produzione.



Errore comune da evitare

Copiare nelle schermate il JavaScript delle demo senza documentare ciclo di vita, risorse, contratti degli eventi, pulizia o confini di sicurezza.



Obiettivi di apprendimento

Al termine di questo modulo sarai in grado di:

- ✓ Creare un controllo lato server derivato da `Wisej.Web.Control` quando serve un'integrazione più stretta.
- ✓ Creare una classe lato client sotto ``/Platform`` usando il modello di classi Wisej.NET/qx.
- ✓ Inviare lo stato tramite ``OnWebRender`` e mapparlo sulle proprietà o sulle opzioni del client.
- ✓ Comprendere `AppearanceKey`, le risorse del tema e gli aspetti da considerare per il rendering a design time.

Guida al lab / esame

Che cosa costruirai nel lab pratico, il flusso di lavoro consigliato, i deliverable richiesti e le domande di autovalutazione prima della consegna.



Lab pratico

Crea un controllo personalizzato `SimpleGaugeControl`` con una classe server e una classe client in ``/Platform``. Aggiungi una chiave di appearance di base e fallo visualizzare in modalità design.

Flusso di lavoro consigliato

- ✓ Apri il progetto esistente del corso di integrazione Wisej.NET ed esegilo in locale.
- ✓ Individua il componente server, il widget client, il file di risorse o l'endpoint coinvolti in questo modulo.
- ✓ Fai una piccola modifica alla volta e provala nel browser prima di andare avanti.
- ✓ Tieni il JavaScript specifico del fornitore dentro l'adapter o il wrapper invece di spargerlo tra le pagine.
- ✓ Aggiorna la UI in modo che si vedano chiaramente successo, errore, avanzamento o cambiamenti nella comunicazione.
- ✓ Raccogli screenshot o note che dimostrano che l'integrazione funziona e spiegano la scelta progettuale.

Deliverable richiesti

- ✓ Classe del controllo lato server
- ✓ Scheletro della classe client qx
- ✓ Voce di tema/appearance
- ✓ Screenshot o note a design time

Domande di autovalutazione

- ✓ Quando un controllo personalizzato vale il lavoro in più?
- ✓ Qual è il ruolo di ``OnWebRender``?
- ✓ Perché l'integrazione con il tema è importante per l'adozione in azienda?