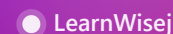


Integrazione rapida con Wisej.Web.Widget

Guida della lezione · Guida al lab / esame



Guida della lezione

Usare Wisej.Web.Widget per integrazioni una tantum, prototipi e verifiche rapide delle librerie dei fornitori. Che cosa significa per uno sviluppatore che integra con Wisej.NET — e come affrontare questo modulo.



Focus di produzione

Usare Wisej.Web.Widget per integrazioni una tantum, prototipi e verifiche rapide delle librerie dei fornitori.

1. Il problema di integrazione

Un team deve dimostrare che un widget JavaScript può essere ospitato in Wisej.NET prima di investire in un wrapper di controllo riutilizzabile.

2. Concetti chiave

Concetto	Come usarlo
Packages	Registra le risorse JavaScript e CSS richieste da un widget una tantum.
InitScript	Crea il widget del fornitore usando le opzioni inviate da .NET.

3. La lezione di produzione

Wisej.Web.Widget è la superficie di integrazione più rapida. Metti il widget su una pagina, collegli script e stili tramite Packages e scrivi un InitScript che crea o aggiorna l'oggetto di terze parti. Questo approccio è ideale quando il widget si usa una sola volta, quando l'API è ancora da esplorare o quando un prototipo va validato prima di costruire un wrapper completo. L'InitScript deve creare l'oggetto del fornitore a partire dall'elemento corretto e memorizzare l'istanza in un posto prevedibile, di solito su `this`. La funzione `update` è facoltativa, ma un prototipo di qualità da produzione dovrebbe includerla ogni volta che il lato .NET può cambiare le opzioni dopo il primo rendering. Alcuni widget dei fornitori espongono un metodo `setOptions` o `option` pulito. Altri richiedono di essere distrutti e ricreati. Il codice di integrazione dovrebbe rendere esplicito questo comportamento. L'oggetto Options è comodo e flessibile, ma è anche facile da usare male. Nei primi prototipi le opzioni in stile JSON sono accettabili, perché portano rapidamente un widget sullo schermo. Quando il wrapper viene riutilizzato, sposta le impostazioni importanti in proprietà .NET con un nome, così da poterle validare, versionare e capire senza leggere la documentazione del fornitore.

Packages per un singolo widget

Un `Wisej.Web.Widget` dichiara le proprie dipendenze tramite la collezione `Packages`. Ogni package indica una sorgente, che può essere un URL, un percorso dell'applicazione o una risorsa incorporata, e Wisej.NET la carica una volta per sessione del browser, nell'ordine della lista, prima che giri l'InitScript. Script e fogli di stile si dichiarano allo stesso modo, così il CSS del fornitore viaggia insieme al suo JavaScript e non viene dimenticato nella schermata successiva.

Nella dashboard del lab il gauge richiede uno script e un foglio di stile, mentre il knob in stile jQuery richiede jQuery, il plugin del knob e il suo foglio di stile. Due widget sulla stessa pagina possono elencare lo stesso package, e il framework lo carica una volta sola. L'ordine tra widget diversi non è garantito, quindi un widget deve elencare da sé tutto ciò di cui ha bisogno invece di contare sul fatto che un altro widget l'abbia già caricato.

Tieni i file del fornitore sotto il percorso dell'applicazione, oppure incorporali, invece di puntare a una CDN pubblica. Un disservizio o un cambio di versione della CDN non deve poter rompere una dashboard in produzione.

init crea, update risincronizza

L'InitScript definisce due funzioni sul widget. `init(options)` viene eseguita una volta, dopo che i package sono stati caricati e il contenitore esiste, e crea l'oggetto del fornitore a partire dalle Options serializzate. `update(options, old)` viene eseguita ogni volta che il server cambia le Options dopo quel primo rendering e riceve sia i nuovi valori sia i precedenti. Senza una funzione update, un'opzione modificata richiederebbe di distruggere e ricreare l'intero widget.

```
init: function (options) {
    this.gauge = new VendorGauge(this.container, options);
},
update: function (options, old) {
    if (options.value !== old.value)
        this.gauge.setValue(options.value);
    if (options.max !== old.max)
        this.gauge.setOptions({ max: options.max });
}
```

Il confronto con `old` è ciò che rende update economica. All'oggetto del fornitore vengono inviati solo i valori cambiati, un'animazione non riparte per un'opzione che non c'entra, e un fornitore che richiede di ricreare il widget per una specifica opzione può farlo solo quando quell'opzione è cambiata. Scrivi la funzione update anche per un prototipo, perché il pulsante lato server del lab che cambia le opzioni è esattamente il caso che gestisce.

Serializzare le opzioni da .NET

La proprietà `Options` è un oggetto dinamico sul server. Qualunque valore le venga assegnato, compresi oggetti annidati, array e tipi anonimi, viene serializzato in JSON da Wisej.NET e arriva in JavaScript come argomento `options`. Una configurazione del gauge con un valore, un intervallo, una lista di fasce colorate e un oggetto etichetta si può costruire in C# con normali oggetti e leggere nell'InitScript come normale JavaScript.

Due regole rendono tutto prevedibile. I nomi delle proprietà vengono convertiti in camelCase durante la serializzazione, quindi un `MaxValue` di C# arriva come `maxValue`, e l'InitScript deve usare la grafia JavaScript. Va assegnato solo ciò che serve al fornitore; un'entità di database o un oggetto di dominio messi in Options vengono serializzati per intero, il che fa trapelare dati verso il browser e appesantisce ogni rendering. Costruisci un piccolo oggetto per il widget e assegna quello.

Durante update gli oggetti annidati vengono confrontati come valori interi, quindi sostituire l'array delle fasce provoca una risincronizzazione di tutte le fasce anche se è cambiato un solo colore. Struttura le opzioni in modo che i valori che cambiano spesso, come la lettura corrente, stiano al livello più alto.

Chiamare il client con Call

Alcuni comportamenti del fornitore sono metodi e non opzioni: reimpostare un'animazione, aggiornare un layout o scorrere fino a un punto. Il server li raggiunge tramite `Call`, che indica una funzione del widget client e passa argomenti serializzati come le Options. L'InitScript definisce quella funzione accanto a `init` e `update`, e la inoltra all'istanza del fornitore memorizzata sul widget.

`Call` non viene eseguita subito. Wisej.NET la mette in coda e la invia al browser con la risposta successiva, insieme a tutte le modifiche alle proprietà fatte nella stessa richiesta. Per questo un gestore di un pulsante può impostare tre opzioni e chiamare due metodi, e il browser riceve un unico blocco coerente. Significa anche che il server non può leggere un risultato da `Call`; le varianti che restituiscono un valore sono l'argomento del Modulo 6.

Ordina il lavoro in modo che lo stato sia corretto prima che il metodo venga eseguito. Imposta prima le opzioni ed emetti la chiamata dopo, perché il blocco viene applicato in ordine e il metodo del fornitore vedrà le opzioni aggiornate.



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:

- ✓ Configurare i package JavaScript e CSS per un'istanza di Widget.
- ✓ Scrivere le funzioni ``init(options)`` e, facoltativamente, ``update(options, old)``.
- ✓ Passare oggetti di opzioni complessi da .NET a JavaScript usando la serializzazione di Wisej.NET.
- ✓ Invocare metodi del client dal server con ``Call`` e capire quando la chiamata viene messa in coda.

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

Costruisci una pagina dashboard con due widget una tantum: un gauge semplice e un knob in stile jQuery. Entrambi devono aggiornarsi quando cambiano le proprietà lato server.

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

- ✓ Packages del widget configurati
- ✓ Implementazione di InitScript e update
- ✓ Pulsante lato server che modifica le opzioni del widget e chiama il client

Domande di autovalutazione

- ✓ Quando viene chiamato `update`?
- ✓ Che cosa rende Options flessibile ma soggetto a errori?
- ✓ Perché un widget potrebbe dover essere distrutto e ricreato?