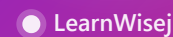


Postback, WebRequest, WebMethod ed endpoint dati

[Guida della lezione](#) · [Guida al lab / esame](#)

Guida della lezione

Alimentare widget JavaScript che recuperano i dati via Ajax, mantenendo le richieste nel contesto del componente Wisej.NET corretto. Che cosa significa per uno sviluppatore che integra con Wisej.NET — e come affrontare questo modulo.



Focus di produzione

Alimentare widget JavaScript che recuperano i dati via Ajax, mantenendo le richieste nel contesto del componente Wisej.NET corretto.

1. Il problema di integrazione

Molti widget avanzati, in particolare griglie, pivot grid, report, visualizzatori e componenti di upload, si aspettano URL o endpoint di servizio invece di dati inviati dal server.

2. Concetti chiave

Concetto	Come usarlo
URL di postback	Instrada le richieste del browser verso l'istanza di componente corretta.
WebMethod	Esponi chiamate in stile RPC per callback o promise lato client.

3. La lezione di produzione

Alcuni widget sono più facili da integrare quando recuperano i dati da un URL. È comune per griglie dati, pivot grid, fogli di calcolo, visualizzatori, componenti di upload e librerie con una propria astrazione della sorgente dati. Wisej.NET fornisce un URL di postback legato all'istanza del componente, così la richiesta arriva nella sessione giusta e nel contesto del controllo giusto. Un gestore di postback si comporta come un piccolo servizio web. Riceve una `HttpRequest` e scrive una `HttpResponse`. La risposta può essere JSON per una griglia, contenuto binario per un file, dati di immagine per un visualizzatore o qualunque formato richiesto dalla libreria del fornitore. Il vantaggio è che il gestore vive con l'istanza del componente, quindi può usare lo stato del componente e i servizi dell'applicazione. `[WebMethod]` ha una forma diversa. È ideale quando il browser deve chiamare un metodo pubblico del server e ricevere un valore restituito già convertito. Può essere statico oppure appartenere a un contenitore di primo livello. Di solito è più pulito per chiamate in stile RPC, come cercare un record, validare un valore o restituire un piccolo insieme di

risultati, mentre i gestori di postback restano la scelta migliore per file, immagini o grandi risposte di dati che una libreria del fornitore si aspetta di scaricare da un URL.

Instradare un URL verso un'istanza di componente

Molti widget avanzati recuperano da soli i propri dati. Una griglia chiede una pagina di righe, un visualizzatore riceve un documento in streaming, un componente di upload invia un file. Tutti vogliono un URL, e la difficoltà è che l'URL deve raggiungere la sessione Wisej.NET giusta e, al suo interno, l'istanza di componente giusta. `GetPostBackURL()` sul server, o `getPostbackUrl()` sul widget client, restituisce esattamente un URL di questo tipo, codificato con gli identificatori di cui il framework ha bisogno per consegnare la richiesta a quel componente.

Il wrapper passa quell'URL al fornitore come indirizzo della sorgente dati, di solito in `init`, e può aggiungere propri parametri di query per distinguere le azioni. Poiché l'URL è legato a un'istanza, due griglie sulla stessa pagina ricevono due URL diversi e ogni richiesta arriva al componente che possiede i dati.

Tratta l'URL come legato alla sessione e di breve durata. Non è l'indirizzo di un'API pubblica, non va memorizzato e non va condiviso tra utenti.

Gestire la richiesta in WebRequest

Quando una richiesta arriva all'URL di postback, scatta l'evento `WebRequest` del componente con la `HttpRequest` e la `HttpResponse` grezze. Il gestore legge l'azione e i parametri, produce i dati, imposta il content type e scrive la risposta. JSON per una griglia, uno stream PDF per un visualizzatore, un'immagine per una miniatura o un CSV per un'esportazione sono solo corpi diversi con content type diversi.

```
private void grid_WebRequest(object sender, WebRequestEventArgs e)
{
    var action = e.Request.QueryString["action"];
    if (action != "load") { e.Response.StatusCode = 400; return; }
    var rows = _service.LoadPage(CurrentTenant, ParsePage(e.Request));
    e.Response.ContentType = "application/json";
    e.Response.Write(JsonSerializer.Serialize(rows));
}
```

Il gestore gira nel contesto del componente, quindi può leggere lo stato del componente e la sessione corrente. È tutto qui il suo vantaggio: la query sui dati sa quale tenant, quali filtri e quali permessi si applicano senza che nulla di tutto questo venga passato dal browser.

WebMethod per chiamate in stile RPC

Quando a chiamare è il JavaScript dell'integrazione stessa e non una sorgente dati del fornitore, un URL è la forma sbagliata. Un metodo marcato con `[WebMethod]` si può chiamare dal client come una funzione, con argomenti tipizzati convertiti da JavaScript e un valore restituito convertito all'indietro, e il client può attendere il risultato con `await`. Può essere statico oppure un metodo d'istanza su un contenitore di primo livello, come la pagina o la finestra.

Il lab implementa lo stesso dataset in sola lettura in entrambi i modi per rendere visibile il compromesso. L'URL di postback è giusto quando è la libreria del fornitore a gestire il recupero dei dati e si aspetta un

indirizzo da chiamare con i propri argomenti di paginazione. Il WebMethod è giusto quando il codice del wrapper ha bisogno di un valore, come validare un codice inserito o recuperare una lista di ricerca, e altrimenti dovrebbe analizzare una risposta HTTP grezza.

Entrambi sono punti di ingresso nel server, ed entrambi aggirano la normale pipeline degli eventi, quindi entrambi hanno bisogno della stessa validazione e autorizzazione che riceverebbe il gestore di un pulsante.

Parametri, content type e sicurezza

Ogni endpoint dati accetta input dal browser, e il browser può essere pilotato da script da chiunque. Il parametro action viene confrontato con una lista fissa, non usato per costruire il nome di un metodo. Gli argomenti di paginazione vengono letti come interi con un limite superiore, così una richiesta di un milione di righe viene rifiutata. I valori di filtro e ordinamento vengono mappati su colonne note invece di essere passati in una query. Tutto ciò che è inatteso restituisce un errore client e viene registrato nel log con gli identificatori di sessione e di correlazione.

Il content type va impostato esplicitamente in ogni risposta, perché un corpo JSON servito come testo confonde alcune librerie e un corpo HTML servito come JSON le confonde tutte. La risposta alla domanda di autovalutazione è `application/json` per il JSON. Le risposte con file impostano l'header disposition e il vero tipo MIME, così che il browser le gestisca correttamente.

- Il gestore verifica che l'utente corrente sia autorizzato a vedere queste righe in questo tenant?
- I parametri di paginazione e filtro sono limitati e mappati su una whitelist?
- Una risposta di errore evita di rivelare sul server qualcosa oltre al codice di stato?



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:

- ✓ Usare ``getPostbackUrl`` / ``GetPostbackURL`` per instradare le richieste del browser verso un'istanza di componente.
- ✓ Gestire le richieste con ``WebRequest`` e restituire JSON, file, immagini o qualunque risposta si aspetti il widget.
- ✓ Esporre metodi ``[WebMethod]`` per chiamate dal client che devono comportarsi come RPC invece che come richieste HTTP grezze.

- ✓ Aggiungere parametri action, validazione, content type e controlli di sicurezza attorno alle richieste di dati dei widget.

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

Implementa sia un endpoint postback sia un endpoint WebMethod per lo stesso dataset in sola lettura. Collega un widget semplice a ciascuno e confronta i compromessi.

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

- ✓ Gestore WebRequest di postback
- ✓ WebMethod con argomenti e valore restituito
- ✓ Nota di confronto: sorgente dati tramite URL vs callback lato client

Domande di autovalutazione

- ✓ Quando il postback è preferibile a un WebMethod?
- ✓ Perché i parametri action vanno validati?
- ✓ Quale content type dovrebbero usare le risposte JSON?