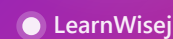


Progettazione avanzata di sessioni, stato, tenant e concorrenza

Guida della lezione · Guida al lab / esame



Guida della lezione

Titolarità della sessione, isolamento dei tenant, concorrenza ottimistica, audit dello stato statico. Che cosa significa per un team Wisej.NET avanzato — e come affrontare questo modulo.



Focus di produzione

Titolarità della sessione, isolamento dei tenant, concorrenza ottimistica, audit dello stato statico.

1. Il problema in produzione

Le applicazioni avanzate servono molti utenti, tenant, schede del browser e operazioni in background. Gli errori sullo stato diventano bug di sicurezza, fughe di dati o flussi di lavoro incoerenti.

2. Concetti chiave

Concetto	Perché è importante
contesto di sessione	Lo stato lato server per utente che Wisej.NET mantiene attivo per ogni sessione del browser. È lì che vanno l'identità dell'utente e lo stato di lavoro, mai in campi statici.
contesto del tenant	L'identità del tenant corrente, risolta una volta per sessione e imposta nei servizi e nell'accesso ai dati, così un cliente non può mai vedere i dati di un altro cliente.
correlation ID	Un identificativo univoco associato a una richiesta o a un job e riportato nei log, così una singola azione dell'utente può essere seguita attraverso servizi e server.
concorrenza ottimistica	Rilevare che due utenti hanno modificato lo stesso record confrontando una versione o un timestamp, e gestire il conflitto invece di sovrascrivere in silenzio.
stato per scheda	Stato che appartiene a una scheda o finestra del browser anziché all'intera sessione, così due schede aperte non si rovinano il lavoro a vicenda.
audit dello stato statico	Una revisione di ogni campo statico e singleton per confermare che non contenga dati per utente o per tenant, perché i campi statici sono condivisi da tutte le sessioni sul server.
revisione delle race condition	Verificare i percorsi di codice in cui task in background, timer o più richieste toccano lo stesso stato di sessione, così che gli aggiornamenti siano sincronizzati correttamente.

3. Il filo della lezione

Lo stato è l'architettura

Wisej.NET mantiene un ricco stato lato server, e questo è un vantaggio potente. In un sistema avanzato quel vantaggio deve essere esplicito. Ogni porzione di stato ha bisogno di un proprietario: applicazione, tenant, utente, sessione, scheda del browser, flusso di lavoro, job in background o richiesta.

L'isolamento dei tenant non è un'etichetta di filtro

Le applicazioni multi-tenant devono imporre i confini tra tenant nei servizi e nell'accesso ai dati. Un filtro della griglia è solo una condizione visiva. Un command handler, un repository e un controllo di autorizzazione devono tutti conoscere l'identità del tenant e rifiutare l'accesso tra tenant diversi.

Cinque ambiti di stato

Una progettazione deliberata dello stato comincia dal dare un nome all'ambito di ogni valore. Lo stato dell'applicazione è condiviso da tutti sul server e deve essere immutabile o sincronizzato: configurazione, dati di riferimento, feature flag. Lo stato del tenant appartiene a un cliente e si raggiunge solo attraverso il contesto del tenant. Lo stato dell'utente segue una persona tra le sessioni, come le preferenze. Lo stato di sessione vive in `Application.Session` per una sessione del browser: identità, tenant corrente, contesto di lavoro. Lo stato di scheda appartiene a una finestra del browser, come un ordine di lavoro aperto, e lo stato di richiesta esiste per un singolo comando, come il suo correlation ID.

Ogni ambito ha una sede. Lo stato dell'applicazione e del tenant vive in servizi e cache con chiave per tenant. Lo stato di sessione vive nell'oggetto del contesto di sessione. Lo stato di scheda vive nel form o nella pagina che possiede la scheda. Lo stato di richiesta vive nel contesto del comando che viaggia lungo la chiamata. Un valore memorizzato in un ambito troppo ampio diventa visibile alle persone sbagliate, e un valore memorizzato in un ambito troppo ristretto va perso quando l'utente apre una seconda scheda.

In revisione, chiedi di ogni campo: chi altro può vederlo, e che cosa succede quando l'utente apre due volte la stessa schermata.

Un SessionContext e un CommandContext consapevoli del tenant

Il `SessionContext` del lab è un unico oggetto creato al login e memorizzato nella sessione. Contiene l'ID utente, il nome visualizzato, i ruoli, i permessi, la cultura e il tenant corrente, risolti a partire da claims verificati e dal tenant che l'utente ha scelto tra quelli a cui ha diritto. Le schermate lo leggono per adattare ciò che mostrano. I servizi lo leggono per decidere che cosa consentire. Nulla legge l'identità del tenant da un controllo della UI, perché il valore di una tendina può essere modificato dal browser.

Il `CommandContext` è il compagno per singola richiesta. Quando un handler avvia un comando, costruisce un contesto a partire dal contesto di sessione e aggiunge un nuovo correlation ID, un timestamp e il nome del comando. Quel contesto viene passato esplicitamente a servizi, repository e job in background, così ogni livello conosce il tenant e il correlation ID senza dover risalire alla sessione. L'elaborazione in background in particolare non può fare affidamento sulla sessione, perché la sessione potrebbe terminare prima del job.

La guardia del tenant in ogni servizio confronta il tenant del contesto con il tenant del record su cui si opera e rifiuta una discrepanza prima che venga eseguita qualsiasi logica di business.

Concorrenza ottimistica e risoluzione dei conflitti

Due dispatcher possono aprire lo stesso ordine di lavoro in due sessioni. Bloccare il record per il primo blocca il secondo magari per tutta una pausa pranzo, quindi le schermate di modifica enterprise usano invece la concorrenza ottimistica. Ogni record porta con sé una versione, una colonna row version o un timestamp, che viene letta quando il record viene caricato e verificata quando viene salvato. Se nel frattempo la versione è cambiata, un'altra sessione ha salvato prima, e il salvataggio viene rifiutato.

È nel rifiuto che sta il lavoro di progettazione. Un semplice messaggio di errore lascia l'utente con il lavoro perso e senza sapere perché. La finestra di dialogo per i conflitti del lab spiega che il record è stato modificato da qualcun altro da quando è stato aperto e offre tre percorsi: Reload scarta le modifiche locali e mostra il record attuale, Compare mostra entrambe le versioni campo per campo così che l'utente possa unirle, e Cancel lascia la schermata com'è perché l'utente decida più tardi. La scelta viene registrata nell'audit trail insieme al correlation ID.

I controlli di concorrenza appartengono al command handler, non al form, così la stessa regola vale quando il salvataggio arriva da un'operazione batch o da un'importazione.

Audit dello stato statico e condiviso

Un processo server Wisej.NET ospita tutte le sessioni nella stessa memoria. Un campo statico, un servizio singleton o una cache senza chiave per tenant sono quindi condivisi contemporaneamente da tutti gli utenti di tutti i tenant. Un campo statico che contiene l'utente corrente è la fuga classica: l'ultima sessione che fa login lo sovrascrive, e la richiesta successiva di un altro utente viene eseguita come quella persona. L'audit dello stato statico del lab elenca ogni campo statico, singleton e cache della soluzione e risponde a una domanda per ciascuno: che cosa contiene, e potrebbe mai essere per utente o per tenant.

La thread safety è l'altra metà. Task in background avviati con `Application.StartTask`, timer e richieste concorrenti da più schede possono toccare tutti gli stessi oggetti nello stesso momento. Un dizionario condiviso letto e scritto senza sincronizzazione si corrompe in silenzio. La revisione delle race condition passa in rassegna ogni oggetto condiviso e verifica se il suo accesso è sincronizzato o se non dovrebbe proprio essere condiviso.

- Ogni campo statico contiene solo configurazione o dati di riferimento immutabili.
- Ogni cache ha una chiave per tenant, e le sue voci non possono essere lette tra tenant diversi.
- Ogni oggetto mutabile condiviso ha una strategia di sincronizzazione documentata.



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 in modo deliberato lo stato di sessione, utente, tenant, scheda e richiesta.
- ✓ Implementare un SessionContext e un contesto di comando consapevoli del tenant.

- ✓ Usare la concorrenza ottimistica e la risoluzione dei conflitti nei flussi di modifica.
- ✓ Verificare lo stato statico e condiviso per thread safety e fughe di dati tra tenant.

Guida al lab / esame

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



Lab pratico

Estendi EnterpriseOps con la selezione del tenant, servizi consapevoli del tenant, correlation ID e concorrenza ottimistica sugli ordini di lavoro. Crea una finestra di dialogo per i conflitti che spiega le modifiche obsolete e offre i percorsi Reload, Compare e Cancel.

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

- ✓ SessionContext consapevole del tenant
- ✓ CommandContext con correlation ID
- ✓ Report dell'audit sullo stato statico
- ✓ Implementazione della concorrenza ottimistica
- ✓ Finestra di dialogo per la risoluzione dei conflitti

Domande di revisione per lo studente

- ✓ L'ID del tenant può essere falsificato da un controllo della UI?
- ✓ Che cosa succede quando lo stesso ordine di lavoro viene modificato in due sessioni?
- ✓ Quali servizi possono essere condivisi in sicurezza?



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.