

UX avanzata dei flussi: procedure guidate, orchestrazione modale e compensazione

Guida della lezione · Guida al lab / esame

● LearnWisej

Guida della lezione

Flussi di business complessi, risultati tipizzati, annullamento e compensazione, flussi di lavoro riprendibili. Che cosa significa per un team Wisej.NET avanzato — e come affrontare questo modulo.



Focus di produzione

Flussi di business complessi, risultati tipizzati, annullamento e compensazione, flussi di lavoro riprendibili.

1. Il problema in produzione

I flussi di lavoro enterprise raramente stanno in un solo pulsante Save. Le schermate avanzate coordinano approvazioni, assegnazioni, validazioni, upload e verifiche esterne preservando la fiducia dell'utente.

2. Concetti chiave

Concetto	Perché è importante
orchestrazione dei flussi di lavoro	Un servizio che possiede la sequenza, le regole e lo stato di un processo di business a più passi, così il flusso non resta sepolto in diversi form.
stato del wizard	Le risposte accumulate e lo stato di validazione di un wizard a più passi, tenuti in un unico oggetto tipizzato invece che sparsi tra i controlli.
risultato tipizzato	Un valore di ritorno che modella esplicitamente gli esiti di successo, errore di validazione ed errore, così i chiamanti devono gestire ogni percorso invece di affidarsi alle eccezioni.
comando di compensazione	Un'operazione che annulla gli effetti di un passo precedente quando un passo successivo fallisce, mantenendo il sistema coerente senza un'unica gigantesca transazione.
flusso di lavoro riprendibile	Un processo che rende persistente il proprio avanzamento, così un utente può chiudere il browser e continuare più tardi, o un errore può essere recuperato dall'ultimo passo riuscito.
matrice dei percorsi di errore	Una tabella che elenca ogni passo, che cosa può andare storto lì e come risponde il flusso di lavoro, così la gestione degli errori è progettata e non improvvisata.

3. Il filo della lezione

Un flusso di lavoro non è una sequenza di form

Un wizard è solo la parte visibile di un processo di business. La domanda di progettazione avanzata è dove vive lo stato del flusso di lavoro, quale servizio possiede le transizioni e come la UI riprende dopo un'interruzione.

I risultati tipizzati mantengono onesto il codice

Le finestre di dialogo e le pagine del wizard dovrebbero restituire un'intenzione tipizzata, non flag magici sparsi tra i controlli. Questo rende le transizioni del flusso di lavoro testabili e più facili da rivedere.

Comandi tipizzati in ingresso, risultati tipizzati in uscita

L'Escalation Wizard raccoglie un motivo, gli allegati, un approvatore, una data di scadenza e le opzioni di notifica in sei pagine. Nessuna di queste pagine esegue logica di business. Riempiono un unico `EscalationCommand` e, quando l'utente conferma, un servizio di workflow lo esegue e restituisce un `WorkflowResult`. Il comando è un oggetto semplice con ogni campo tipizzato e validato, così si può costruire in uno unit test con la stessa facilità che nel wizard.

Anche il risultato è tipizzato. Dice se l'escalation è stata creata, quali validazioni sono fallite e su quale campo, se un passo esterno come la notifica è riuscito e che cosa dovrebbe fare poi l'utente. I chiamanti scelgono in base a questi esiti invece di intercettare eccezioni o esaminare stringhe. Una pagina del wizard che riceve un errore di validazione può saltare al passo giusto ed evidenziare il campo; un chiamante batch può registrare lo stesso risultato senza alcuna UI.

La prima domanda di revisione del lab è se il flusso di lavoro si può testare senza il wizard. Con un comando e un risultato tipizzati la risposta è sì, e quel test esiste prima del wizard.

Wizard e finestre modali che restano sottili

Una finestra di dialogo modale o una pagina del wizard è un form, e in questo corso i form sono sottili. La pagina raccoglie i valori, esegue la validazione dell'input necessaria per abilitare il pulsante successivo e scrive i valori nell'oggetto di stato del wizard. Non decide se l'approvatore è ammesso, se la data di scadenza è accettabile per questa priorità o se un allegato è obbligatorio. Queste sono regole di business, e vivono nel servizio di workflow, dove si applicano a ogni chiamante.

L'oggetto di stato del wizard è la parte visibile di questa disciplina. È un unico oggetto tipizzato che contiene tutto ciò che è stato raccolto finora e quali passi sono completati. Le pagine lo leggono e ci scrivono, l'ultima pagina lo trasforma nel comando e, poiché è un solo oggetto, può essere reso persistente, che è ciò che rende il flusso di lavoro riprendibile dopo un aggiornamento del browser. La seconda domanda del lab, quale stato sopravvive a un aggiornamento, trova risposta in ciò che si trova in quell'oggetto ed è memorizzato sul server.

Una finestra modale che chiama un repository, o una pagina del wizard che tiene un DbContext, ha nascosto logica di business in un Form, e verrà duplicata la prima volta che servirà lo stesso flusso di lavoro senza il wizard.

Compensazione dove le transazioni non arrivano

Creare un'escalation salva un record, invia una notifica tramite un sistema esterno e scrive una voce di audit. Una transazione di database copre il primo e l'ultimo passo, ma non la notifica, che è già stata inviata quando la scrittura dell'audit fallisce. Le transazioni distribuite tra sistemi esterni sono raramente disponibili e raramente valgono il loro costo, quindi il flusso di lavoro usa invece la compensazione: per ogni passo che non può essere annullato con un rollback, definisci un'azione di compensazione che lo annulla o lo neutralizza.

Il percorso di errore del lab è la notifica che fallisce dopo che il salvataggio è riuscito. L'escalation viene mantenuta, perché è valida, e un comando di compensazione registra che la notifica è in sospeso e pianifica un retry o un follow-up manuale. Il caso inverso, l'audit che fallisce dopo la notifica, si compensa registrando esplicitamente la lacuna nell'audit e generando un avviso, perché una notifica inviata non si può ritirare.

La compensazione si progetta passo per passo e si mette per iscritto prima del codice. La domanda per ogni passo è come dovrebbe apparire il sistema se questo passo fallisce dopo che i precedenti sono riusciti.

Una matrice per ogni percorso di errore

La matrice dei percorsi di errore è una tabella con una riga per ogni passo del flusso di lavoro e una colonna per ogni modo in cui può concludersi: conferma, annullamento, timeout, errore di validazione ed errore esterno. Ogni cella dice che cosa fa il sistema, che cosa vede l'utente, che cosa viene reso persistente e che cosa viene sottoposto ad audit. Compilare la matrice fa emergere i percorsi che nessuno aveva progettato, come che cosa succede quando la selezione dell'approvatore va in timeout o quando l'utente annulla dopo aver caricato gli allegati.

La matrice è anche il piano di test. Ogni cella diventa almeno un test del servizio di workflow, e le celle che coinvolgono la UI diventano verifiche manuali nello script della demo. Quando un revisore chiede che cosa succede se la verifica esterna non è disponibile, la risposta è il riferimento a una cella, non un'ipotesi.

- Conferma: il comando viene eseguito, il risultato viene mostrato, la voce di audit viene scritta.
- Annullamento: lo stato raccolto viene scartato o conservato come bozza, e gli upload vengono ripuliti.
- Timeout: il flusso di lavoro può essere ripreso dal suo stato persistente, e nulla resta applicato a metà.
- Errore di validazione: l'utente torna al passo che ha fallito con il campo evidenziato.
- Errore esterno: viene eseguita la compensazione per quel passo e l'esito viene registrato.



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 flussi di lavoro a più passi con comandi e risultati tipizzati.
- ✓ Usare flussi modali e wizard senza nascondere la logica di business nei Form.
- ✓ Implementare compensazione o annullamento dove le vere transazioni non sono possibili.
- ✓ Documentare i percorsi di conferma, annullamento, timeout, errore di validazione ed errore esterno.

Guida al lab / esame

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



Lab pratico

Crea un Escalation Wizard che raccoglie motivo, allegati, approvatore, data di scadenza e opzioni di notifica. Implementa l'orchestrazione in un servizio, non nelle pagine del wizard, e aggiungi la compensazione per un errore di notifica simulato.

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

- ✓ Flusso delle schermate del wizard
- ✓ Servizio di workflow
- ✓ Oggetti comando e risultato tipizzati
- ✓ Matrice dei percorsi di errore
- ✓ Esempio di compensazione

Domande di revisione per lo studente

- ✓ Il flusso di lavoro si può testare senza il wizard?
- ✓ Quale stato sopravvive a un aggiornamento del browser?
- ✓ Che cosa succede se la notifica riesce ma la scrittura del log di audit fallisce?



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.