

UX avancée des workflows : assistants, orchestration modale et compensation

Guide de la leçon · Guide du lab / de l'examen



Guide de la leçon

Flux métier complexes, résultats typés, annulation/compensation, workflows pouvant reprendre. Ce que cela signifie pour une équipe Wisej.NET avancée — et comment aborder ce module.



Enjeu de production

Flux métier complexes, résultats typés, annulation/compensation, workflows pouvant reprendre.

1. Le problème en production

Les workflows d'entreprise tiennent rarement dans un simple bouton Save. Les écrans avancés coordonnent approbations, affectations, validations, uploads et vérifications externes tout en préservant la confiance de l'utilisateur.

2. Concepts clés

Concept	Pourquoi c'est important
orchestration de workflow	Service qui possède la séquence, les règles et l'état d'un processus métier en plusieurs étapes, pour que le flux ne soit pas enfoui dans plusieurs formulaires.
état de l'assistant	Réponses accumulées et état de validation d'un assistant en plusieurs étapes, conservés dans un seul objet typé plutôt que dispersés dans les contrôles.
résultat typé	Valeur de retour qui modélise explicitement le succès, l'échec de validation et les erreurs, pour que les appelants doivent traiter chaque chemin au lieu de s'appuyer sur les exceptions.
commande de compensation	Opération qui annule les effets d'une étape antérieure lorsqu'une étape ultérieure échoue, gardant le système cohérent sans une seule transaction géante.
workflow pouvant reprendre	Processus qui persiste sa progression pour qu'un utilisateur puisse fermer le navigateur et continuer plus tard, ou pour qu'un échec puisse être rattrapé à partir de la dernière étape réussie.
matrice des chemins d'échec	Tableau listant chaque étape, ce qui peut y mal tourner et la façon dont le workflow réagit, pour que la gestion des échecs soit conçue plutôt qu'improvisée.

3. Déroulé de la leçon

Un workflow n'est pas une suite de formulaires

Un assistant n'est que la partie visible d'un processus métier. La question de conception avancée est de savoir où vit l'état du workflow, quel service possède les transitions et comment l'UI reprend après une interruption.

Les résultats typés gardent le code honnête

Les boîtes de dialogue et les pages d'assistant doivent renvoyer une intention typée, pas des indicateurs magiques dispersés dans les contrôles. Les transitions du workflow deviennent ainsi testables et plus faciles à relire.

Des commandes typées en entrée, des résultats typés en sortie

L'Escalation Wizard recueille un motif, des pièces jointes, un approbateur, une date d'échéance et des options de notification sur six pages. Aucune de ces pages n'exécute de logique métier. Elles remplissent une seule `EscalationCommand`, et lorsque l'utilisateur confirme, un service de workflow l'exécute et renvoie un `WorkflowResult`. La commande est un objet simple dont chaque champ est typé et validé ; elle peut donc être construite dans un test unitaire aussi facilement que dans l'assistant.

Le résultat est lui aussi typé. Il indique si l'escalade a été créée, quelles validations ont échoué et sur quel champ, si une étape externe comme la notification a réussi, et ce que l'utilisateur doit faire ensuite. Les appelants font un switch sur ces issues au lieu d'intercepter des exceptions ou d'analyser des chaînes. Une page d'assistant qui reçoit un échec de validation peut sauter à la bonne étape et mettre le champ en évidence ; un appelant par lots peut journaliser le même résultat sans aucune UI.

La première question de révision du lab est de savoir si le workflow peut être testé sans l'assistant. Avec une commande et un résultat typés, la réponse est oui, et ce test existe avant même l'assistant.

Des assistants et des modales qui restent légers

Une boîte de dialogue modale ou une page d'assistant est un formulaire, et dans ce cours les formulaires sont légers. La page recueille des valeurs, effectue la validation de saisie nécessaire pour activer le bouton suivant et écrit les valeurs dans l'objet d'état de l'assistant. Elle ne décide pas si l'approbateur est autorisé, si la date d'échéance est acceptable pour cette priorité ni si une pièce jointe est obligatoire. Ce sont des règles métier, et elles vivent dans le service de workflow, où elles s'appliquent à tous les appelants.

L'objet d'état de l'assistant est la partie visible de cette discipline. C'est un objet typé unique qui contient tout ce qui a été recueilli jusqu'ici et quelles étapes sont terminées. Les pages le lisent et y écrivent, la dernière page le transforme en commande, et comme il s'agit d'un seul objet, il peut être persisté, ce qui permet au workflow de reprendre après une actualisation du navigateur. La deuxième question du lab, quel état survit à l'actualisation, trouve sa réponse dans ce que contient cet objet stocké sur le serveur.

Une modale qui appelle un repository, ou une page d'assistant qui détient un DbContext, cache de la logique métier dans un Form et sera dupliquée dès la première fois que le même workflow sera nécessaire sans l'assistant.

La compensation là où les transactions ne peuvent pas aller

Créer une escalade persiste un enregistrement, envoie une notification via un système externe et écrit une entrée d'audit. Une transaction de base de données couvre la première et la dernière opération, mais pas la notification, déjà envoyée lorsque l'écriture de l'audit échoue. Les transactions distribuées entre systèmes externes sont rarement disponibles et valent rarement leur coût ; le workflow utilise donc la compensation : pour chaque étape qui ne peut pas être annulée par un rollback, définissez une action de compensation qui l'annule ou la neutralise.

Le chemin d'échec du lab est l'échec de la notification après la réussite de la persistance. L'escalade est conservée, car elle est valide, et une commande de compensation consigne que la notification reste en attente et planifie une nouvelle tentative ou un suivi manuel. Le cas inverse, l'échec de l'audit après la notification, est compensé en consignait explicitement le trou dans l'audit et en déclenchant une alerte, car une notification envoyée ne peut pas être « désenvoyée ».

La compensation se conçoit étape par étape et se rédige avant le code. La question à se poser pour chaque étape est : à quoi le système doit-il ressembler si cette étape échoue après la réussite des précédentes ?

Une matrice pour chaque chemin d'échec

La matrice des chemins d'échec est un tableau avec une ligne par étape du workflow et une colonne par façon dont elle peut se terminer : confirmation, annulation, expiration, échec de validation et échec externe. Chaque cellule indique ce que fait le système, ce que voit l'utilisateur, ce qui est persisté et ce qui est audité. Remplir la matrice révèle les chemins que personne n'avait conçus, comme ce qui se passe quand la sélection de l'approbateur expire ou quand l'utilisateur annule après l'upload des pièces jointes.

La matrice est aussi le plan de test. Chaque cellule devient au moins un test du service de workflow, et les cellules qui impliquent l'UI deviennent des vérifications manuelles dans le script de démo. Quand un relecteur demande ce qui se passe si la vérification externe est indisponible, la réponse est une référence de cellule, pas une supposition.

- Confirmation : la commande s'exécute, le résultat est affiché, l'entrée d'audit est écrite.
- Annulation : l'état recueilli est abandonné ou conservé comme brouillon, et les fichiers uploadés sont nettoyés.
- Expiration : le workflow peut reprendre à partir de son état persisté, et rien n'est appliqué à moitié.
- Échec de validation : l'utilisateur est renvoyé à l'étape en échec avec le champ mis en évidence.
- Échec externe : la compensation de cette étape s'exécute, et l'issue est consignée.



Réflexe de production avancé

Ne vous arrêtez pas quand l'écran paraît correct. Expliquez à qui appartient l'état, les implications de sécurité, les chemins d'échec et le comportement en production avant de considérer le module comme terminé.



Objectifs d'apprentissage

À la fin de ce module, vous saurez :

- ✓ Concevoir des workflows en plusieurs étapes avec des commandes et des résultats typés.
- ✓ Utiliser des flux modaux et des assistants sans cacher la logique métier dans les Forms.
- ✓ Implémenter une compensation ou une annulation là où de véritables transactions ne sont pas possibles.
- ✓ Documenter les chemins de confirmation, d'annulation, d'expiration, d'échec de validation et d'échec externe.

Guide du lab / de l'examen

Ce que vous allez construire dans le lab pratique, l'approche suggérée et les livrables attendus.



Lab pratique

Créez un Escalation Wizard qui recueille le motif, les pièces jointes, l'approbateur, la date d'échéance et les options de notification. Implémentez l'orchestration dans un service, pas dans les pages de l'assistant, et ajoutez une compensation pour un échec de notification simulé.

Approche suggérée

- ✓ Lisez le problème en production et identifiez le risque d'architecture avant de toucher à l'UI.
- ✓ Créez ou mettez à jour d'abord l'encapsulation par un service, de commande, de modèle ou de composant concernée.
- ✓ Construisez l'écran ou le composant Wisej.NET en suivant les standards de projet établis.
- ✓ Ajoutez une démonstration du chemin d'échec, pas seulement du chemin nominal.
- ✓ Consignez ce qui a changé dans le journal d'apprentissage et mettez à jour la checklist concernée.

Livrables attendus

- ✓ Enchaînement des écrans de l'assistant
- ✓ Service de workflow
- ✓ Objets de commande/résultat typés
- ✓ Matrice des chemins d'échec
- ✓ Exemple de compensation

Questions de révision

- ✓ Le workflow peut-il être testé sans l'assistant ?
- ✓ Quel état survit à une actualisation du navigateur ?
- ✓ Que se passe-t-il si la notification réussit mais que l'écriture du journal d'audit échoue ?



Réflexe de production avancé

Ne vous arrêtez pas quand l'écran paraît correct. Expliquez à qui appartient l'état, les implications de sécurité, les chemins d'échec et le comportement en production avant de considérer le module comme terminé.