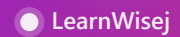


Fortgeschrittene Workflow-UX: Assistenten, modale Orchestrierung und Kompensation

Lektionsleitfaden · Lab- und Prüfungsleitfaden



Lektionsleitfaden

Komplexe Geschäftsabläufe, typisierte Ergebnisse, Undo und Kompensation, fortsetzbare Workflows. Was das für ein Wisej.NET-Team auf Expertenniveau bedeutet – und wie Sie dieses Modul lesen.



Fokus: Produktivbetrieb

Komplexe Geschäftsabläufe, typisierte Ergebnisse, Undo und Kompensation, fortsetzbare Workflows.

1. Das Problem im Produktivbetrieb

Enterprise-Workflows passen selten hinter einen einzigen Speichern-Button. Anspruchsvolle Bildschirme koordinieren Genehmigungen, Zuweisungen, Validierungen, Uploads und externe Prüfungen und erhalten dabei das Vertrauen der Benutzer.

2. Zentrale Konzepte

Konzept	Warum es wichtig ist
Workflow-Orchestrierung	Ein Service, dem Reihenfolge, Regeln und Zustand eines mehrstufigen Geschäftsprozesses gehören, damit der Ablauf nicht über mehrere Forms verstreut ist.
Assistentenzustand	Die gesammelten Antworten und der Validierungsstatus eines mehrstufigen Assistenten, gehalten in einem einzigen typisierten Objekt statt verteilt auf Controls.
typisiertes Ergebnis	Ein Rückgabewert, der Erfolg, Validierungsfehler und Fehlerfälle explizit modelliert, sodass Aufrufer jeden Pfad behandeln müssen, statt sich auf Exceptions zu verlassen.
Kompensations-Command	Eine Operation, die die Auswirkungen eines früheren Schritts rückgängig macht, wenn ein späterer Schritt scheitert, und das System ohne eine einzige riesige Transaktion konsistent hält.
fortsetzbarer Workflow	Ein Prozess, der seinen Fortschritt persistiert, sodass ein Benutzer den Browser schließen und später weitermachen oder ein Fehler ab dem letzten erfolgreichen Schritt behoben werden kann.
Matrix der Fehlerpfade	Eine Tabelle mit jedem Schritt, dem, was dort schiefgehen kann, und der Reaktion des Workflows, damit die Fehlerbehandlung entworfen statt improvisiert wird.

3. Der rote Faden der Lektion

Ein Workflow ist keine Abfolge von Forms

Ein Assistent ist nur der sichtbare Teil eines Geschäftsprozesses. Die eigentliche Designfrage lautet, wo der Workflow-Zustand liegt, welchem Service die Übergänge gehören und wie die UI nach einer Unterbrechung fortgesetzt wird.

Typisierte Ergebnisse halten den Code ehrlich

Dialoge und Seiten eines Assistenten sollten eine typisierte Absicht zurückgeben, keine magischen Flags, die über Controls verteilt sind. Das macht Workflow-Übergänge testbar und leichter zu reviewen.

Typisierte Commands hinein, typisierte Ergebnisse heraus

Der Eskalationsassistent erfasst auf sechs Seiten einen Grund, Anhänge, einen Genehmiger, ein Fälligkeitsdatum und Benachrichtigungsoptionen. Keine dieser Seiten führt Geschäftslogik aus. Sie befüllen einen einzigen `EscalationCommand`, und wenn der Benutzer bestätigt, führt ein Workflow-Service ihn aus und gibt ein `WorkflowResult` zurück. Der Command ist ein einfaches Objekt, dessen Felder alle typisiert und validiert sind; er lässt sich in einem Unit-Test genauso leicht erzeugen wie im Assistenten.

Auch das Ergebnis ist typisiert. Es sagt, ob die Eskalation angelegt wurde, welche Validierungen bei welchem Feld fehlgeschlagen sind, ob ein externer Schritt wie die Benachrichtigung erfolgreich war und was der Benutzer als Nächstes tun sollte. Aufrufer verzweigen anhand dieser Ergebnisse, statt Exceptions abzufangen oder Strings auszuwerten. Eine Seite des Assistenten, die einen Validierungsfehler erhält, kann zum richtigen Schritt springen und das Feld hervorheben; ein Aufrufer aus einer Stapelverarbeitung kann dasselbe Ergebnis ganz ohne UI protokollieren.

Die erste Wiederholungsfrage des Labs lautet, ob sich der Workflow ohne den Assistenten testen lässt. Mit typisiertem Command und Ergebnis lautet die Antwort Ja – und dieser Test existiert, bevor es den Assistenten gibt.

Assistenten und modale Dialoge, die schlank bleiben

Ein modaler Dialog oder eine Seite eines Assistenten ist eine Form, und Forms sind in diesem Kurs schlank. Die Seite erfasst Werte, führt die Eingabevalidierung durch, die zum Aktivieren des Weiter-Buttons nötig ist, und schreibt die Werte in das Zustandsobjekt des Assistenten. Sie entscheidet nicht, ob der Genehmiger zulässig ist, ob das Fälligkeitsdatum für diese Priorität akzeptabel ist oder ob ein Anhang erforderlich ist. Das sind Geschäftsregeln, und sie liegen im Workflow-Service, wo sie für jeden Aufrufer gelten.

Das Zustandsobjekt des Assistenten ist der sichtbare Teil dieser Disziplin. Es ist ein einzelnes typisiertes Objekt, das alles bisher Erfasste enthält und festhält, welche Schritte abgeschlossen sind. Die Seiten lesen daraus und schreiben hinein, die letzte Seite macht daraus den Command, und weil es ein einziges Objekt ist, lässt es sich persistieren – genau das macht den Workflow nach einem Browser-Refresh fortsetzbar. Die zweite Frage des Labs, welcher Zustand einen Refresh übersteht, beantwortet sich durch das, was in diesem Objekt steht und auf dem Server gespeichert ist.

Ein modaler Dialog, der ein Repository aufruft, oder eine Seite eines Assistenten, die einen DbContext hält, versteckt Geschäftslogik in einer Form – und diese Logik wird dupliziert, sobald derselbe Workflow zum

ersten Mal ohne Assistenten gebraucht wird.

Kompensation, wo Transaktionen nicht hinreichen

Das Anlegen einer Eskalation persistiert einen Datensatz, versendet eine Benachrichtigung über ein externes System und schreibt einen Audit-Eintrag. Eine Datenbanktransaktion deckt den ersten und den letzten Schritt ab, nicht aber die Benachrichtigung, die bereits verschickt ist, wenn das Schreiben des Audits scheitert. Verteilte Transaktionen über externe Systeme sind selten verfügbar und selten ihren Preis wert; der Workflow setzt daher auf Kompensation: Für jeden Schritt, der sich nicht zurückrollen lässt, wird eine kompensierende Aktion definiert, die ihn rückgängig macht oder neutralisiert.

Der Fehlerpfad des Labs ist eine Benachrichtigung, die scheitert, nachdem das Persistieren gelungen ist. Die Eskalation bleibt erhalten, weil sie gültig ist, und ein Kompensations-Command hält fest, dass die Benachrichtigung noch aussteht, und plant eine Wiederholung oder eine manuelle Nachverfolgung ein. Der umgekehrte Fall – das Audit scheitert nach der Benachrichtigung – wird kompensiert, indem die Audit-Lücke ausdrücklich festgehalten und ein Alarm ausgelöst wird, denn eine verschickte Benachrichtigung lässt sich nicht zurückholen.

Kompensation wird pro Schritt entworfen und schriftlich festgehalten, bevor Code entsteht. Die Frage für jeden Schritt lautet: Wie sollte das System aussehen, wenn dieser Schritt scheitert, nachdem die vorherigen gelungen sind?

Eine Matrix für jeden Fehlerpfad

Die Matrix der Fehlerpfade ist eine Tabelle mit einer Zeile pro Workflow-Schritt und einer Spalte pro möglichem Ausgang: Bestätigen, Abbrechen, Timeout, Validierungsfehler und externer Fehler. Jede Zelle beschreibt, was das System tut, was der Benutzer sieht, was persistiert und was auditiert wird. Beim Ausfüllen der Matrix zeigen sich die Pfade, die niemand entworfen hatte – etwa was passiert, wenn bei der Auswahl des Genehmigers ein Timeout auftritt oder wenn der Benutzer abbricht, nachdem Anhänge hochgeladen wurden.

Die Matrix ist zugleich der Testplan. Jede Zelle wird zu mindestens einem Test des Workflow-Services, und die Zellen, die die UI betreffen, werden zu manuellen Prüfungen im Demo-Skript. Fragt ein Reviewer, was passiert, wenn die externe Prüfung nicht erreichbar ist, lautet die Antwort ein Verweis auf eine Zelle, keine Vermutung.

- Bestätigen: Der Command wird ausgeführt, das Ergebnis angezeigt und der Audit-Eintrag geschrieben.
- Abbrechen: Der erfasste Zustand wird verworfen oder als Entwurf behalten, und Uploads werden bereinigt.
- Timeout: Der Workflow lässt sich aus seinem persistierten Zustand fortsetzen, und nichts ist nur halb angewendet.
- Validierungsfehler: Der Benutzer kehrt zum fehlerhaften Schritt zurück, und das Feld wird hervorgehoben.
- Externer Fehler: Die Kompensation für diesen Schritt läuft, und das Ergebnis wird festgehalten.



Expertenregel für den Produktivbetrieb

Hören Sie nicht auf, sobald der Bildschirm korrekt aussieht. Erklären Sie, wem der Zustand gehört, welche Sicherheitsfolgen es gibt, welche Fehlerpfade existieren und wie sich das System im Produktivbetrieb verhält, bevor Sie das Modul als abgeschlossen betrachten.



Lernziele

Am Ende dieses Moduls können Sie:

- ✓ mehrstufige Workflows mit typisierten Commands und Ergebnissen entwerfen.
- ✓ modale Abläufe und Assistenten einsetzen, ohne Geschäftslogik in Forms zu verstecken.
- ✓ Kompensation oder Undo implementieren, wo echte Transaktionen nicht möglich sind.
- ✓ die Pfade für Bestätigen, Abbrechen, Timeout, Validierungsfehler und externe Fehler dokumentieren.

Lab- und Prüfungsleitfaden

Was Sie im Praxis-Lab bauen, das vorgeschlagene Vorgehen und die geforderten Ergebnisse.



Praxis-Lab

Erstellen Sie einen Eskalationsassistenten, der Grund, Anhänge, Genehmiger, Fälligkeitsdatum und Benachrichtigungsoptionen erfasst. Implementieren Sie die Orchestrierung in einem Service, nicht in den Seiten des Assistenten, und ergänzen Sie eine Kompensation für einen simulierten Benachrichtigungsfehler.

Vorgeschlagenes Vorgehen

- ✓ Lesen Sie das Problem im Produktivbetrieb und benennen Sie das Architekturrisiko, bevor Sie die UI anfassen.
- ✓ Legen Sie zuerst die betroffene Grenze für Service, Command, Modell oder Komponente an oder aktualisieren Sie sie.
- ✓ Erstellen Sie den Wisej.NET-Bildschirm oder die Komponente nach den etablierten Projektstandards.
- ✓ Demonstrieren Sie auch einen Fehlerpfad, nicht nur den Happy Path.
- ✓ Halten Sie die Änderungen im Lernprotokoll fest und aktualisieren Sie die zugehörige Checkliste.

Geforderte Ergebnisse

- ✓ Bildschirmablauf des Assistenten
- ✓ Workflow-Service
- ✓ Typisierte Command-/Result-Objekte
- ✓ Matrix der Fehlerpfade
- ✓ Kompensationsbeispiel

Wiederholungsfragen

- ✓ Lässt sich der Workflow ohne den Assistenten testen?
- ✓ Welcher Zustand übersteht einen Browser-Refresh?
- ✓ Was passiert, wenn die Benachrichtigung gelingt, das Audit-Logging aber scheitert?



Expertenregel für den Produktivbetrieb

Hören Sie nicht auf, sobald der Bildschirm korrekt aussieht. Erklären Sie, wem der Zustand gehört, welche Sicherheitsfolgen es gibt, welche Fehlerpfade existieren und wie sich das System im Produktivbetrieb verhält, bevor Sie das Modul als abgeschlossen betrachten.