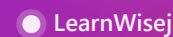


UX des données à fort volume, filtrage côté serveur et opérations par lots

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



Guide de la leçon

Grandes grilles, pagination, filtrage, tri, modifications par lots, retour à l'utilisateur. Ce que cela signifie pour une équipe Wisej.NET avancée — et comment aborder ce module.



Enjeu de production

Grandes grilles, pagination, filtrage, tri, modifications par lots, retour à l'utilisateur.

1. Le problème en production

Les utilisateurs en entreprise attendent des grilles rapides et des actions de masse fiables. Tout charger en mémoire ou refaire les bindings à l'aveugle fonctionne en démo, mais échoue avec un volume de données réaliste.

2. Concepts clés

Concept	Pourquoi c'est important
pagination côté serveur	La base de données ne renvoie que la page de lignes demandée, si bien qu'une grille peut parcourir des millions d'enregistrements sans les charger en mémoire ni dans le navigateur.
projection de recherche	Modèle de lecture allégé ne contenant que les colonnes affichées par un écran de recherche, au lieu d'entités complètes avec toutes leurs propriétés de navigation.
état de la grille	La page courante, le tri, le filtre et la sélection d'une grille, conservés sur le serveur pour survivre aux actualisations et être restaurés de façon prévisible.
raisonnement virtualisé	Concevoir chaque liste et chaque grille comme si le jeu de données était illimité, pour que les lignes soient récupérées, affichées et libérées à la demande.
commande par lots	Une opération serveur unique qui applique une modification à de nombreux enregistrements sélectionnés à la fois, avec la validation et la gestion des transactions en un seul endroit.
échec partiel	Situation où certains éléments d'un lot réussissent et d'autres échouent. L'opération doit indiquer exactement lesquels ont échoué et pourquoi, plutôt qu'une erreur tout-ou-rien.
retour réactif	Indicateurs de progression, contrôles désactivés et messages d'état qui tiennent les utilisateurs informés pendant les opérations longues au lieu de laisser l'écran figé.

3. Déroulé de la leçon

Les gros volumes sont un problème de workflow

L'objectif n'est pas d'afficher un million de lignes. L'objectif est d'aider l'utilisateur à trouver le bon enregistrement, à comprendre son statut et à agir en toute sécurité. La recherche, les filtres, les vues enregistrées, les synthèses et les actions par lots comptent plus que le nombre brut de lignes.

Projeter pour la grille

Une ligne de grille a souvent besoin d'un texte d'affichage, d'icônes de statut, d'un âge calculé, d'indicateurs de permission et de compteurs. Un modèle de projection garde la grille rapide et claire sans polluer les entités du domaine avec des détails de présentation.

Filtrer et paginer sur le serveur

Le premier écran de l'Enterprise Work Queue charge exactement une page de lignes, et ce fait définit la conception. La grille ne reçoit pas une liste ; elle reçoit un `PagedResult` contenant les lignes de la page courante et le nombre total. Les filtres, l'ordre de tri et le numéro de page sont envoyés au serveur sous la forme d'une `WorkQueueQuery`, et le service de requête les traduit en une requête de base de données qui ne renvoie que ces lignes. C'est la base de données qui filtre, car elle dispose des index et de la mémoire nécessaires.

Les vues enregistrées rendent le filtrage côté serveur exploitable. Un dispatcher qui consulte chaque matin les ordres en retard de haute priorité enregistre ce filtre une fois, et la vue enregistrée est une définition de requête stockée, pas un résultat en cache. La persistance du tri et la page courante font partie de l'état de la grille conservé sur le serveur ; une actualisation restaure donc la même vue au lieu de repartir de zéro.

La question de revue est la première du lab : combien d'enregistrements sont chargés pour le premier écran ? Si la réponse dépasse une page, la conception n'a pas encore commencé.

Des projections plutôt que des entités

Une ligne de grille a besoin de valeurs d'affichage : un numéro d'ordre de travail, un nom de client, un libellé et une icône de statut, un âge en jours et l'indication que l'utilisateur courant peut ou non le réaffecter. L'entité du domaine ne possède directement aucune de ces valeurs ; elle a des clés étrangères, des horodatages et une énumération de statut. Lier l'entité à la grille oblige l'écran à calculer les valeurs d'affichage ligne par ligne, et oblige l'entité à se doter de propriétés de présentation qu'elle ne devrait pas avoir.

Une projection de recherche est une classe taillée pour l'écran, comme `WorkQueueRow`, remplie par le service de requête avec une instruction select qui calcule les valeurs d'affichage dans la base de données. Elle ne porte que les colonnes affichées par la grille ; une page de deux cents lignes pèse donc quelques kilo-octets au lieu d'un graphe d'entités. Elle porte aussi les indicateurs de permission dont la grille a besoin pour activer ou désactiver les actions, calculés une seule fois sur le serveur.

Les projections et les entités changent pour des raisons différentes. Une nouvelle colonne dans la grille modifie la projection ; une nouvelle règle métier modifie l'entité. Les garder séparées signifie qu'aucun de ces changements ne casse l'autre.

Des opérations par lots qui n'échouent jamais en silence

Réaffecter cinquante ordres de travail, c'est une action utilisateur et cinquante opérations métier. La commande par lots du lab porte les clés sélectionnées et le technicien cible, et le gestionnaire les traite une par une : il valide chacune, applique la modification et consigne le résultat par ligne. Une ligne peut échouer parce que l'ordre a été clôturé entre-temps, parce que le technicien n'a pas la certification requise ou parce qu'un autre utilisateur l'a modifié en premier. Aucun de ces cas ne doit arrêter les quarante-neuf autres.

L'échec partiel est donc le résultat attendu, pas une erreur. Le gestionnaire renvoie un rapport listant chaque ligne comme réussie, en échec ou ignorée, avec une raison. L'écran affiche la progression pendant l'exécution du lot et le rapport à la fin, et propose de relancer uniquement les lignes en échec. Le chemin d'échec du lab est exactement celui-ci : réaffecter trois ordres, l'un échoue, et l'utilisateur voit lequel et pourquoi.

- Chaque ligne est-elle validée et appliquée indépendamment, dans sa propre transaction ou dans un groupe documenté ?
- Le résultat nomme-t-il chaque ligne et son issue ?
- L'utilisateur peut-il relancer les échecs sans répéter les réussites ?

Mesurer la performance perçue

Les utilisateurs jugent un écran au temps qu'ils attendent, pas au nombre de lignes qu'il pourrait contenir. La performance perçue se mesure aux moments qu'ils remarquent : le temps entre l'ouverture de l'écran et l'apparition des premières lignes, le temps entre la modification d'un filtre et la mise à jour de la grille, le temps entre le clic sur une action par lots et la première indication de progression. Les notes de performance du lab consignent ces valeurs pour la Work Queue avec des volumes de données et un réseau réalistes.

L'optimisation suit les mesures. Un premier chargement lent signifie généralement trop de colonnes ou un index manquant pour le tri par défaut. Un changement de filtre lent signifie généralement que le filtre n'est pas traduit vers la base de données. Un lot qui semble figé a besoin d'une progression dès la première seconde, même si le travail lui-même prend une minute. Un retour réactif, un bouton désactivé, une barre de progression et une ligne d'état ne coûtent presque rien et changent la durée ressentie de l'attente.

Le raisonnement virtualisé est l'habitude derrière tout cela : concevez comme si le jeu de données n'avait pas de limite supérieure, et chaque liste, liste déroulante et export fonctionnera encore quand il sera dix fois plus volumineux.



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 écrans de listes à fort volume avec filtrage et pagination côté serveur.
- ✓ Créer des projections qui séparent l'affichage de la grille des modèles du domaine.
- ✓ Implémenter des opérations par lots sûres avec progression, validation et gestion des échecs partiels.
- ✓ Mesurer et optimiser la performance perçue.

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 écran Enterprise Work Queue avec des filtres côté serveur, des vues enregistrées, la pagination, la persistance du tri et une action de réaffectation par lots. Ajoutez une progression et un rapport de résultat par ligne pour les échecs partiels.

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

- ✓ Service de requête paginée
- ✓ Modèle de projection de recherche
- ✓ Définition de vue enregistrée
- ✓ Workflow de réaffectation par lots
- ✓ Notes de performance

Questions de révision

- ✓ Combien d'enregistrements sont chargés pour le premier écran ?
- ✓ Que se passe-t-il quand une ligne du lot échoue ?
- ✓ L'utilisateur peut-il relancer la recherche après une actualisation ?



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.