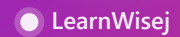


Observabilité, diagnostic, performances et profilage des sessions

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



Guide de la leçon

Santé, logs, métriques, traces, timing client/serveur, croissance de la mémoire et des sessions. Ce que cela signifie pour une équipe Wisej.NET avancée — et comment aborder ce module.



Enjeu de production

Santé, logs, métriques, traces, timing client/serveur, croissance de la mémoire et des sessions.

1. Le problème en production

Les équipes avancées doivent diagnostiquer les problèmes de production sans deviner à distance. Elles ont besoin de diagnostics sûrs, de logs structurés, de mesures de performance et de pratiques de revue des sessions et de la mémoire.

2. Concepts clés

Concept	Pourquoi c'est important
logging structuré	Écrire les entrées de log sous forme de champs nommés plutôt que de texte libre, pour qu'elles puissent être recherchées, filtrées et corrélées par des outils en production.
ID de corrélation	Identifiant unique présent dans chaque entrée de log d'une requête ou d'un job, pour qu'une action utilisateur puisse être suivie à travers les services et les serveurs.
page de diagnostic	Écran réservé aux opérateurs qui expose le nombre de sessions, la mémoire, les jobs actifs et les erreurs récentes, offrant de la visibilité sans attacher de débogueur.
health check	Endpoint léger qui indique si l'application et ses dépendances fonctionnent, utilisé par les répartiteurs de charge et la supervision.
budget de performance	Limites convenues de temps de réponse, de taille de payload et de mémoire par session, pour que les régressions soient détectées par rapport à un chiffre plutôt qu'à une impression.
mémoire de session	Mémoire serveur consommée par chaque session active. Elle doit rester faible et prévisible, car un serveur Wisej.NET héberge de nombreuses sessions à la fois.
revue de la libération des ressources	Vérifier que les timers, les gestionnaires d'événements, les DbContexts et les tâches en arrière-plan sont libérés à la fin d'un formulaire ou d'une session, pour éviter les fuites de mémoire.
gestion sûre des erreurs	Intercepter et journaliser les exceptions avec tous leurs détails sur le serveur tout en affichant un message générique aux utilisateurs, pour que les éléments internes ne soient jamais exposés.

3. Déroulé de la leçon

La production n'a pas de débogueur

Un runbook de production doit indiquer quelle version s'exécute, dans quel environnement, quels services sont sains, quelles erreurs se sont produites et quel ID de corrélation relie le signalement de l'utilisateur aux logs.

Les diagnostics doivent être sûrs

Un écran de diagnostic doit être protégé par rôle et masquer les secrets. Il peut afficher l'environnement, la version, les feature flags, le thème actif, l'état des services, le résultat du health check, l'état des files d'attente et les logs récents sans risque.

Des pages de diagnostic utiles sans exposer de secrets

La page de diagnostic est le premier endroit que consulte un ingénieur du support à l'arrivée d'un ticket. Elle doit indiquer, sur un seul écran, quelle version et quel build s'exécutent, quel environnement et quel nœud ont servi la session, quels feature flags sont actifs, si les services dépendants sont sains et combien de temps ont duré les opérations récentes. La page du lab affiche côte à côte la version, le nœud, les flags

et les budgets de performance, pour que la réponse à la plupart des premières questions se lise d'un coup d'œil.

La page est aussi une surface d'attaque ; son contenu est donc choisi par type plutôt qu'à la main. Le `DiagnosticSnapshot` du lab ne porte que des champs sûrs à afficher : aucune chaîne de connexion, aucune clé d'API, aucune donnée personnelle, aucune stack trace complète. Les valeurs potentiellement sensibles, comme le nom d'un compte de stockage, sont masquées avant d'atteindre le view model. La page est protégée par un contrôle de rôle, et son ouverture est elle-même auditée.

Une bonne question de revue consiste à se demander si la page pourrait être projetée devant un client. Si une valeur devait être cachée, elle n'a rien à faire là.

Logs structurés, corrélation et messages sûrs

Le logging structuré signifie que chaque entrée est un ensemble de champs nommés plutôt qu'une phrase. L'entrée d'une requête lente consigne le nom de l'opération, le tenant, l'utilisateur, la durée en millisecondes, la taille de page et l'ID de corrélation comme valeurs distinctes. L' `OperationTimer` du lab enveloppe chaque opération de service et écrit une telle entrée à la fin, pour que le log puisse ensuite être filtré sur toutes les opérations de plus de deux secondes dans un tenant.

L'ID de corrélation est ce qui relie les éléments entre eux. Il est créé au début d'une action utilisateur, transporté dans le contexte de commande à travers chaque appel de service, transmis aux jobs en arrière-plan et écrit dans les entrées d'audit et les rapports d'erreur. Quand un utilisateur signale qu'une approbation a échoué à 14:32, le support retrouve l'ID de corrélation dans l'erreur qui lui a été affichée et le suit à travers chaque ligne de log produite par cette action.

Le message que voit l'utilisateur est différent de l'entrée de log. Il indique que l'opération n'a pas pu aboutir, fournit l'ID de corrélation et suggère la marche à suivre. Il n'affiche jamais le texte de l'exception, le SQL ni le chemin sur le serveur, car ces éléments aident davantage un attaquant que l'utilisateur.

Mesurer ce que ressentent les utilisateurs

La performance se mesure aux moments où les utilisateurs attendent. Le démarrage correspond au temps entre la première requête et une page principale utilisable. Le chargement d'un écran correspond au temps entre un clic de menu et l'affichage du formulaire. Une requête de données correspond au temps que le service passe à récupérer une page de résultats. Le rafraîchissement des bindings correspond au temps entre l'arrivée des données et leur affichage dans la grille. Le timing des jobs en arrière-plan couvre l'attente en file et l'exécution, et le timing des mises à jour mesure le temps nécessaire pour qu'une modification poussée apparaisse dans le navigateur.

Chacune de ces mesures reçoit un budget, c'est-à-dire un chiffre que l'équipe s'engage à ne pas dépasser, consigné dans le tableau des budgets de performance du lab. Le budget transforme une plainte vague en une régression mesurable. Le chemin d'échec du lab en montre l'intérêt : une requête de 2 340 ms pour un budget de 500 ms apparaît en rouge dans le tableau, et son entrée de log structurée révèle une taille de page de 5 000 lignes. Corriger la taille de page fait passer la ligne au vert, et le log le prouve.

Mesurez en production aussi bien qu'en développement, car le volume de données, la distance réseau et les sessions concurrentes modifient les chiffres. Un panneau de mesure qui ne fonctionne que sur le poste d'un développeur est une démo, pas un outil de diagnostic.

Mémoire des sessions et erreurs de libération

Chaque session active occupe de la mémoire sur le serveur tant qu'elle existe, et un serveur Wisej.NET héberge de nombreuses sessions à la fois. Un formulaire qui garde une liste de 50 000 lignes dans un champ, un rapport en cache ou un gros tampon de fichier multiplie ce coût par le nombre d'utilisateurs. La revue de la mémoire des sessions du lab liste ce que chaque session conserve, pourquoi et pour combien de temps, et remet en question chaque élément de plus de quelques mégaoctets.

Les erreurs de libération sont la seconde source de croissance. Un timer démarré dans un formulaire et jamais arrêté maintient le formulaire en vie après sa fermeture. Un gestionnaire d'événements attaché à un service partagé garde une référence vers un contrôle libéré. Un DbContext conservé dans un champ survit à l'opération pour laquelle il a été créé. Une tâche en arrière-plan qui capture un formulaire dans une closure retient tout ce que le formulaire référence. Chacun de ces cas fuit un peu, et sur une journée de travail, ce peu finit par compter.

- Quels champs de ce formulaire contiennent des collections, et sont-ils vidés à la fermeture ?
- Quels timers, abonnements et tâches ce formulaire démarre-t-il, et où sont-ils arrêtés ?
- Quels objets sont conservés pendant toute la session, et chacun doit-il l'être ?



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 pages de diagnostic qui aident l'exploitation sans exposer de secrets.
- ✓ Ajouter un logging structuré, des ID de corrélation et des messages d'erreur adaptés à l'utilisateur.
- ✓ Mesurer les temps de démarrage, de chargement des écrans, de requête de données, de rafraîchissement des bindings, des jobs en arrière-plan et des mises à jour.
- ✓ Identifier les risques de croissance de la mémoire et de la session ainsi que les erreurs de libération des ressources.

Guide du lab / de l'examen

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



Lab pratique

Ajoutez l'observabilité à EnterpriseOps : logging structuré, ID de corrélation, page de diagnostic, état de santé, panneau de mesure des performances et checklist de revue de la mémoire des sessions. Simulez une requête lente et montrez comment elle est diagnostiquée.

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

- ✓ Page de diagnostic
- ✓ Exemple de log structuré
- ✓ Propagation de l'ID de corrélation
- ✓ Tableau des budgets de performance
- ✓ Notes d'audit mémoire/session

Questions de révision

- ✓ Le support peut-il identifier la version et le tenant exacts ?
- ✓ Les diagnostics exposent-ils des secrets ?
- ✓ Quels objets sont conservés pendant toute la session ?



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.