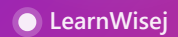


Observability, Diagnose, Performance und Session-Profiling

Lektionsleitfaden · Lab- und Prüfungsleitfaden



Lektionsleitfaden

Health, Logs, Metriken, Traces, Client- und Server-Timing, Wachstum von Speicher und Sessions. Was das für ein Wisej.NET-Team auf Expertenniveau bedeutet – und wie Sie dieses Modul lesen.



Fokus: Produktivbetrieb

Health, Logs, Metriken, Traces, Client- und Server-Timing, Wachstum von Speicher und Sessions.

1. Das Problem im Produktivbetrieb

Erfahrene Teams müssen Probleme im Produktivbetrieb diagnostizieren, ohne aus der Ferne zu raten. Dafür brauchen sie sichere Diagnosen, strukturierte Logs, Performance-Messungen und etablierte Reviews für Sessions und Speicher.

2. Zentrale Konzepte

Konzept	Warum es wichtig ist
Strukturiertes Logging	Logeinträge werden als benannte Felder statt als Freitext geschrieben, damit Tools sie im Produktivbetrieb durchsuchen, filtern und miteinander in Beziehung setzen können.
Korrelations-ID	Eine eindeutige Kennung, die jeden Logeintrag einer Anfrage oder eines Jobs begleitet, damit sich eine einzelne Benutzeraktion über Services und Server hinweg verfolgen lässt.
Diagnoseseite	Ein Bildschirm nur für den Betrieb, der Session-Anzahl, Speicher, aktive Jobs und aktuelle Fehler zeigt und so Einblick gibt, ohne einen Debugger anzuhängen.
Health Check	Ein leichtgewichtiger Endpunkt, der meldet, ob die Anwendung und ihre Abhängigkeiten funktionieren. Load Balancer und Monitoring nutzen ihn.
Performance-Budget	Vereinbarte Grenzen für Antwortzeit, Payload-Größe und Speicher pro Session, damit Regressionen an einer Zahl gemessen werden statt an einem Gefühl.
Session-Speicher	Der Serverspeicher, den jede aktive Session belegt. Er muss klein und vorhersehbar bleiben, denn ein Wisej.NET-Server hostet viele Sessions gleichzeitig.
Dispose-Review	Die Prüfung, ob Timer, Event-Handler, DbContexts und Hintergrundaufgaben freigegeben werden, wenn ein Formular oder eine Session endet, damit kein Speicherleck entsteht.
Sichere Fehlerbehandlung	Exceptions werden auf dem Server mit allen Details abgefangen und protokolliert, während Benutzer eine allgemeine Meldung sehen – so werden Interna nie preisgegeben.

3. Der rote Faden der Lektion

Im Produktivbetrieb gibt es keinen Debugger

Ein Runbook für den Produktivbetrieb sollte beantworten, welche Version läuft, in welcher Umgebung sie läuft, welche Services gesund sind, welche Fehler aufgetreten sind und welche Korrelations-ID die Meldung des Benutzers mit den Logs verbindet.

Diagnosen müssen sicher sein

Ein Diagnosebildschirm sollte durch Rollen geschützt sein und Secrets schwärzen. Er kann Umgebung, Version, Feature-Flags, aktives Theme, Servicestatus, das Ergebnis des Health Checks, den Zustand der Warteschlangen und aktuelle, unbedenkliche Logs anzeigen.

Diagnoseseiten, die helfen, ohne Secrets preiszugeben

Eine Diagnoseseite ist der erste Ort, an dem ein Support-Engineer nachsieht, wenn ein Ticket eingeht. Sie sollte auf einem Bildschirm beantworten, welche Version und welcher Build laufen, welche Umgebung und welcher Knoten die Session bedient haben, welche Feature-Flags aktiv sind, ob abhängige Services gesund sind und wie lange die letzten Operationen gedauert haben. Die Seite im Lab zeigt Version, Knoten, Flags und Performance-Budgets nebeneinander, sodass sich die meisten ersten Fragen mit einem Blick beantworten lassen.

Die Seite ist zugleich eine Angriffsfläche, deshalb wird ihr Inhalt über einen Typ festgelegt und nicht von Hand zusammengestellt. Der `DiagnosticSnapshot` im Lab enthält nur Felder, deren Anzeige unbedenklich ist: keine Connection Strings, keine API-Schlüssel, keine personenbezogenen Daten, keine vollständigen Stacktraces. Werte, die sensibel sein könnten, etwa der Name eines Speicherkontos, werden geschwärzt, bevor sie das View-Model erreichen. Die Seite liegt hinter einer Rollenprüfung, und schon das Öffnen wird auditiert.

Eine gute Review-Frage lautet, ob man die Seite vor einem Kunden per Beamer zeigen könnte. Müsste irgendein Wert abgedeckt werden, gehört er nicht auf die Seite.

Strukturierte Logs, Korrelation und sichere Meldungen

Strukturiertes Logging bedeutet, dass jeder Eintrag aus benannten Feldern besteht statt aus einem Satz. Ein Eintrag für eine langsame Abfrage hält Operationsname, Mandant, Benutzer, Dauer in Millisekunden, Seitengröße und Korrelations-ID als einzelne Werte fest. Der `OperationTimer` im Lab umschließt jede Service-Operation und schreibt nach deren Abschluss einen solchen Eintrag, sodass sich das Log später auf alle Operationen eines Mandanten filtern lässt, die länger als zwei Sekunden gedauert haben.

Die Korrelations-ID hält die Teile zusammen. Sie wird erzeugt, wenn eine Benutzeraktion beginnt, im Command-Kontext durch jeden Serviceaufruf getragen, an Hintergrundjobs übergeben und in Audit-Einträge und Fehlerberichte geschrieben. Meldet ein Benutzer, dass eine Freigabe um 14:32 Uhr fehlgeschlagen ist, findet der Support die Korrelations-ID in der Fehlermeldung, die dem Benutzer angezeigt wurde, und verfolgt sie durch jede Logzeile, die diese Aktion erzeugt hat.

Die Meldung, die der Benutzer sieht, ist etwas anderes als der Logeintrag. Sie sagt, dass die Operation nicht abgeschlossen werden konnte, nennt die Korrelations-ID und schlägt einen nächsten Schritt vor. Sie zeigt nie den Exception-Text, das SQL oder den Serverpfad, denn diese helfen einem Angreifer mehr als dem Benutzer.

Messen, was Benutzer spüren

Performance wird dort gemessen, wo Benutzer warten. Der Start ist die Zeit von der ersten Anfrage bis zu einer nutzbaren Hauptseite. Das Laden eines Bildschirms ist die Zeit vom Klick im Menü bis zum gerenderten Formular. Eine Datenabfrage ist die Zeit, die der Service braucht, um eine Seite mit Ergebnissen zu holen. Die Binding-Aktualisierung ist die Zeit zwischen dem Eintreffen der Daten und ihrer Anzeige im Grid. Das Timing von Hintergrundjobs umfasst Wartezeit in der Warteschlange und Ausführung, und das Update-Timing erfasst, wie lange eine per Push gesendete Änderung braucht, bis sie im Browser erscheint.

Jeder dieser Punkte erhält ein Budget – eine Zahl, deren Einhaltung das Team vereinbart und die in der Tabelle mit dem Performance-Budget im Lab festgehalten wird. Das Budget verwandelt eine vage Beschwerde in eine messbare Regression. Der Fehlerpfad im Lab zeigt den Nutzen: Eine Abfrage, die 2.340 ms bei einem Budget von 500 ms gedauert hat, ist in der Tabelle rot, und ihr strukturierter Logeintrag verrät eine Seitengröße von 5.000 Zeilen. Wird die Seitengröße korrigiert, wird die Zeile grün, und das Log beweist es.

Messen Sie im Produktivbetrieb ebenso wie in der Entwicklung, denn Datenmenge, Netzwerkentfernung und gleichzeitige Sessions verändern die Zahlen. Ein Timing-Panel, das nur auf einem Entwicklerrechner funktioniert, ist eine Demo, keine Diagnose.

Session-Speicher und Fehler beim Freigeben

Jede aktive Session belegt Speicher auf dem Server, solange sie existiert, und ein Wisej.NET-Server hostet viele Sessions gleichzeitig. Ein Formular, das eine Liste mit 50.000 Zeilen in einem Feld hält, ein zwischengespeicherter Bericht oder ein großer Dateipuffer vervielfachen diese Kosten mit der Zahl der Benutzer. Das Review des Session-Speichers im Lab listet auf, was jede Session behält, warum und wie lange, und hinterfragt jeden Eintrag über einige Megabyte.

Fehler beim Freigeben sind die zweite Quelle des Wachstums. Ein Timer, der in einem Formular gestartet und nie gestoppt wird, hält das Formular nach dem Schließen am Leben. Ein Event-Handler an einem gemeinsam genutzten Service hält eine Referenz auf ein bereits freigegebenes Control. Ein DbContext in einem Feld überlebt die Operation, für die er erzeugt wurde. Eine Hintergrundaufgabe, die ein Formular in einer Closure einfängt, hält alles fest, worauf das Formular verweist. Jedes dieser Lecks ist klein, doch über einen Arbeitstag summiert sich das Kleine.

- Welche Felder dieses Formulars enthalten Collections, und werden sie beim Schließen geleert?
- Welche Timer, Abonnements und Tasks startet dieses Formular, und wo werden sie gestoppt?
- Welche Objekte bleiben für die gesamte Session erhalten, und braucht es jedes davon wirklich?



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:

- ✓ Diagnoseseiten entwerfen, die dem Betrieb helfen, ohne Secrets preiszugeben.
- ✓ strukturiertes Logging, Korrelations-IDs und für Benutzer unbedenkliche Fehlermeldungen ergänzen.
- ✓ die Zeiten für Start, Aufbau von Bildschirmen, Datenabfragen, Binding-Aktualisierung, Hintergrundjobs und Updates messen.
- ✓ Risiken durch wachsenden Speicher- und Session-Verbrauch sowie Fehler beim Freigeben (Dispose) erkennen.

Lab- und Prüfungsleitfaden

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



Praxis-Lab

Statten Sie EnterpriseOps mit Observability aus: strukturiertes Logging, Korrelations-IDs, Diagnoseseite, Health-Status, Panel für Performance-Timing und eine Review-Checkliste für den Session-Speicher. Simulieren Sie eine langsame Abfrage und zeigen Sie, wie sie diagnostiziert wird.

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

- ✓ Diagnoseseite
- ✓ Beispiel für strukturiertes Logging
- ✓ Weitergabe der Korrelations-ID
- ✓ Tabelle mit dem Performance-Budget
- ✓ Audit-Notizen zu Speicher und Sessions

Wiederholungsfragen

- ✓ Kann der Support die genaue Version und den Mandanten ermitteln?
- ✓ Gibt die Diagnose Secrets preis?
- ✓ Welche Objekte bleiben für die gesamte Session erhalten?



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.