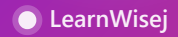


Fortgeschrittenes Design für Sessions, Zustand, Mandanten und Nebenläufigkeit

Lektionsleitfaden · Lab- und Prüfungsleitfaden



Lektionsleitfaden

Verantwortung für Sessions, Mandantenisolation, optimistische Nebenläufigkeit, Audits des statischen Zustands. Was das für ein Wisej.NET-Team auf Expertenniveau bedeutet – und wie Sie dieses Modul lesen.



Fokus: Produktivbetrieb

Verantwortung für Sessions, Mandantenisolation, optimistische Nebenläufigkeit, Audits des statischen Zustands.

1. Das Problem im Produktivbetrieb

Anspruchsvolle Anwendungen bedienen viele Benutzer, Mandanten, Browser-Tabs und Hintergrundoperationen. Fehler beim Zustand werden zu Sicherheitslücken, Datenlecks oder inkonsistenten Workflows.

2. Zentrale Konzepte

Konzept	Warum es wichtig ist
Session-Kontext	Der benutzerbezogene Serverzustand, den Wisej.NET für jede Browser-Session am Leben hält. Hierhin gehören Benutzeridentität und Arbeitszustand – niemals in statische Felder.
Mandantenkontext	Die Identität des aktuellen Mandanten, einmal pro Session aufgelöst und in Services und Datenzugriff durchgesetzt, damit ein Kunde nie die Daten eines anderen Kunden sehen kann.
Korrelations-ID	Eine eindeutige Kennung, die an einen Request oder Job angehängt und durch die Logs mitgeführt wird, sodass sich eine Benutzeraktion über Services und Server hinweg verfolgen lässt.
optimistische Nebenläufigkeit	Erkennen, dass zwei Benutzer denselben Datensatz bearbeitet haben, indem eine Version oder ein Zeitstempel verglichen wird – und den Konflikt behandeln, statt stillschweigend zu überschreiben.
tab-bezogener Zustand	Zustand, der zu einem Browser-Tab oder Fenster gehört statt zur ganzen Session, damit sich zwei geöffnete Tabs nicht gegenseitig die Arbeit verderben.
Audit des statischen Zustands	Eine Prüfung jedes statischen Felds und jedes Singletons, die bestätigt, dass darin keine benutzer- oder mandantenbezogenen Daten liegen, denn statische Member teilen sich alle Sessions auf dem Server.
Review von Race Conditions	Die Prüfung von Codepfaden, in denen Hintergrund-Tasks, Timer oder mehrere Requests auf denselben Session-Zustand zugreifen, damit Aktualisierungen korrekt synchronisiert werden.

3. Der rote Faden der Lektion

Zustand ist die Architektur

Wisej.NET hält umfangreichen serverseitigen Zustand – ein starker Vorteil. In einem anspruchsvollen System muss dieser Vorteil explizit gemacht werden. Jedes Stück Zustand braucht einen Besitzer: Anwendung, Mandant, Benutzer, Session, Browser-Tab, Workflow, Hintergrundjob oder Request.

Mandantenisolation ist kein Filter-Label

Mandantenfähige Anwendungen müssen Mandantengrenzen in Services und im Datenzugriff durchsetzen. Ein Grid-Filter ist nur eine visuelle Bedingung. Ein Command-Handler, ein Repository und eine Autorisierungsprüfung müssen alle die Mandantenidentität kennen und mandantenübergreifende Zugriffe ablehnen.

Fünf Geltungsbereiche von Zustand

Bewusstes Zustandsdesign beginnt damit, den Geltungsbereich jedes Werts zu benennen. Anwendungszustand teilen sich alle auf dem Server; er muss unveränderlich oder synchronisiert sein: Konfiguration, Referenzdaten, Feature-Flags. Mandantenzustand gehört einem Kunden und ist nur über den Mandantenkontext erreichbar. Benutzerzustand begleitet eine Person über Sessions hinweg, etwa ihre

Einstellungen. Session-Zustand lebt für eine Browser-Session in `Application.Session`: Identität, aktueller Mandant, Arbeitskontext. Tab-Zustand gehört zu einem Browserfenster, etwa ein geöffneter Arbeitsauftrag, und Request-Zustand existiert für einen einzigen Command, etwa dessen Korrelations-ID.

Jeder Geltungsbereich hat ein Zuhause. Anwendungs- und Mandantenzustand liegen in Services und in Caches mit dem Mandanten als Schlüssel. Session-Zustand liegt im Session-Kontextobjekt. Tab-Zustand liegt in der Form oder Page, der der Tab gehört. Request-Zustand liegt im Command-Kontext, der durch den Aufruf wandert. Ein Wert, der einen Geltungsbereich zu weit abgelegt ist, wird für die falschen Personen sichtbar; ein Wert, der einen Geltungsbereich zu eng abgelegt ist, geht verloren, sobald der Benutzer einen zweiten Tab öffnet.

Fragen Sie im Review bei jedem Feld: Wer kann das noch sehen, und was passiert, wenn der Benutzer denselben Bildschirm zweimal öffnet?

Ein mandantenfähiger SessionContext und CommandContext

Der `SessionContext` des Labs ist ein einzelnes Objekt, das beim Login erzeugt und in der Session gespeichert wird. Er enthält Benutzer-ID, Anzeigenamen, Rollen, Berechtigungen, Kultur und den aktuellen Mandanten – aufgelöst aus verifizierten Claims und dem Mandanten, den der Benutzer aus den für ihn freigegebenen gewählt hat. Bildschirme lesen ihn, um ihre Anzeige anzupassen. Services lesen ihn, um zu entscheiden, was sie erlauben. Nichts liest die Mandantenidentität aus einem UI-Control, denn der Wert einer Dropdown-Liste lässt sich vom Browser aus ändern.

Der `CommandContext` ist das Gegenstück pro Request. Wenn ein Handler einen Command startet, baut er einen Kontext aus dem Session-Kontext und ergänzt eine neue Korrelations-ID, einen Zeitstempel und den Namen des Commands. Dieser Kontext wird explizit an Services, Repositories und Hintergrundjobs übergeben, sodass jede Schicht Mandant und Korrelations-ID kennt, ohne auf die Session zurückzugreifen. Gerade Hintergrundverarbeitung kann sich nicht auf die Session verlassen, denn die Session kann enden, bevor der Job fertig ist.

Der Mandanten-Guard in jedem Service vergleicht den Mandanten im Kontext mit dem Mandanten des betroffenen Datensatzes und weist eine Abweichung zurück, bevor irgendeine Geschäftslogik läuft.

Optimistische Nebenläufigkeit und Konfliktlösung

Zwei Disponenten können denselben Arbeitsauftrag in zwei Sessions öffnen. Den Datensatz für den ersten zu sperren, blockiert den zweiten womöglich eine ganze Mittagspause lang; Bearbeitungsbildschirme im Enterprise-Umfeld nutzen deshalb optimistische Nebenläufigkeit. Jeder Datensatz trägt eine Version – eine Rowversion-Spalte oder einen Zeitstempel –, die beim Laden gelesen und beim Speichern geprüft wird. Hat sich die Version dazwischen geändert, hat eine andere Session zuerst gespeichert, und das Speichern wird abgelehnt.

Die eigentliche Designarbeit steckt in dieser Ablehnung. Eine nackte Fehlermeldung lässt den Benutzer mit verllorener Arbeit und ohne Erklärung zurück. Der Konfliktdialog des Labs erklärt, dass der Datensatz seit dem Öffnen von jemand anderem geändert wurde, und bietet drei Wege an: Reload verwirft die lokalen Änderungen und zeigt den aktuellen Datensatz, Compare zeigt beide Versionen Feld für Feld, damit der Benutzer zusammenführen kann, und Cancel lässt den Bildschirm unverändert, damit der Benutzer später entscheiden kann. Die Wahl wird mit der Korrelations-ID im Audit-Trail festgehalten.

Prüfungen auf Nebenläufigkeit gehören in den Command-Handler, nicht in die Form, damit dieselbe Regel auch greift, wenn das Speichern aus einer Stapeloperation oder einem Import kommt.

Statischen und gemeinsam genutzten Zustand auditieren

Ein Wisej.NET-Serverprozess beherbergt alle Sessions im selben Speicher. Ein statisches Feld, ein Singleton-Service oder ein Cache ohne Mandantenschlüssel wird daher gleichzeitig von allen Benutzern aller Mandanten geteilt. Ein statisches Feld mit dem aktuellen Benutzer ist das klassische Leck: Die zuletzt angemeldete Session überschreibt es, und der nächste Request eines anderen Benutzers läuft als diese Person. Das Audit des statischen Zustands im Lab listet jedes statische Feld, jeden Singleton und jeden Cache der Lösung auf und beantwortet für jeden eine Frage: Was enthält er, und könnte das jemals benutzer- oder mandantenbezogen sein?

Threadsicherheit ist die zweite Hälfte. Mit `Application.StartTask` gestartete Hintergrund-Tasks, Timer und gleichzeitige Requests aus mehreren Tabs können alle zur selben Zeit auf dieselben Objekte zugreifen. Ein gemeinsam genutztes Dictionary, das ohne Synchronisation gelesen und beschrieben wird, wird stillschweigend beschädigt. Das Review der Race Conditions geht jedes gemeinsam genutzte Objekt durch und prüft, ob der Zugriff synchronisiert ist oder ob es überhaupt nicht geteilt werden sollte.

- Jedes statische Feld enthält nur Konfiguration oder unveränderliche Referenzdaten.
- Jeder Cache hat den Mandanten als Schlüssel, und seine Einträge sind mandantenübergreifend nicht lesbar.
- Jedes gemeinsam genutzte veränderliche Objekt hat eine dokumentierte Synchronisationsstrategie.



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:

- ✓ Session-, Benutzer-, Mandanten-, Tab- und Request-Zustand bewusst entwerfen.
- ✓ einen mandantenfähigen SessionContext und einen Command-Kontext implementieren.
- ✓ optimistische Nebenläufigkeit und Konfliktlösung in Bearbeitungs-Workflows einsetzen.
- ✓ statischen und gemeinsam genutzten Zustand auf Threadsicherheit und Mandantenlecks auditieren.

Lab- und Prüfungsleitfaden

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



Praxis-Lab

Erweitern Sie EnterpriseOps um Mandantenauswahl, mandantenfähige Services, Korrelations-IDs und optimistische Nebenläufigkeit für Arbeitsaufträge. Erstellen Sie einen Konfliktdialog, der veraltete Bearbeitungen erklärt und die Wege Reload, Compare und Cancel anbietet.

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

- ✓ Mandantenfähiger SessionContext
- ✓ CommandContext mit Korrelations-ID
- ✓ Auditbericht zum statischen Zustand
- ✓ Implementierung der optimistischen Nebenläufigkeit
- ✓ Dialog zur Konfliktlösung

Wiederholungsfragen

- ✓ Lässt sich die Mandanten-ID über ein UI-Control fälschen?
- ✓ Was passiert, wenn derselbe Arbeitsauftrag in zwei Sessions bearbeitet wird?
- ✓ Welche Services können gefahrlos gemeinsam genutzt werden?



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.