

Diseño avanzado de sesiones, estado, tenants y concurrencia

Guía de la lección · Guía del laboratorio / examen

● LearnWisej

Guía de la lección

Propiedad de la sesión, aislamiento de tenants, concurrencia optimista, auditorías del estado estático. Lo que esto significa para un equipo avanzado de Wisej.NET — y cómo leer este módulo.



Enfoque de producción

Propiedad de la sesión, aislamiento de tenants, concurrencia optimista, auditorías del estado estático.

1. El problema en producción

Las aplicaciones avanzadas atienden a muchos usuarios, tenants, pestañas del navegador y operaciones en segundo plano. Los errores de estado se convierten en fallos de seguridad, fugas de datos o flujos de trabajo incoherentes.

2. Conceptos clave

Concepto	Por qué importa
contexto de sesión	El estado de servidor por usuario que Wisej.NET mantiene vivo para cada sesión del navegador. Es donde deben estar la identidad del usuario y el estado de trabajo, nunca en campos estáticos.
contexto de tenant	La identidad del tenant actual, resuelta una vez por sesión y aplicada en los servicios y el acceso a datos, para que un cliente nunca pueda ver los datos de otro cliente.
ID de correlación	Un identificador único asociado a una solicitud o a un trabajo y propagado por los registros, para poder seguir una acción del usuario a través de servicios y servidores.
concurrency optimista	Detectar que dos usuarios han editado el mismo registro comparando una versión o una marca de tiempo, y gestionar el conflicto en lugar de sobrescribir en silencio.
estado por pestaña	Estado que pertenece a una pestaña o ventana del navegador y no a toda la sesión, para que dos pestañas abiertas no estropeen el trabajo de la otra.
auditoría del estado estático	Una revisión de cada campo estático y singleton para confirmar que no contiene datos por usuario ni por tenant, porque los estáticos los comparten todas las sesiones del servidor.
revisión de condiciones de carrera	Comprobar las rutas de código en las que tareas en segundo plano, temporizadores o varias solicitudes tocan el mismo estado de sesión, para que las actualizaciones se sincronicen correctamente.

3. Desarrollo de la lección

El estado es la arquitectura

Wisej.NET mantiene un estado rico en el servidor, lo que supone una ventaja poderosa. En un sistema avanzado esa ventaja debe hacerse explícita. Cada parte del estado necesita un propietario: la aplicación, el tenant, el usuario, la sesión, la pestaña del navegador, el flujo de trabajo, el trabajo en segundo plano o la solicitud.

El aislamiento de tenants no es una etiqueta de filtro

Las aplicaciones multitenant deben aplicar los límites de tenant en los servicios y en el acceso a datos. Un filtro de cuadrícula es solo una condición visual. Un manejador de comandos, un repositorio y una comprobación de autorización deben entender la identidad del tenant y rechazar el acceso entre tenants.

Cinco ámbitos de estado

Diseñar el estado de forma deliberada empieza por nombrar el ámbito de cada valor. El estado de aplicación lo comparten todos en el servidor y debe ser inmutable o estar sincronizado: configuración, datos de referencia, feature flags. El estado de tenant pertenece a un cliente y solo se alcanza a través del contexto de tenant. El estado de usuario acompaña a una persona entre sesiones, como las preferencias. El estado de sesión vive en `Application.Session` para una sesión del navegador: identidad, tenant actual,

contexto de trabajo. El estado de pestaña pertenece a una ventana del navegador, como una orden de trabajo abierta, y el estado de solicitud existe durante un comando, como su ID de correlación.

Cada ámbito tiene su lugar. El estado de aplicación y el de tenant viven en servicios y cachés indexadas por tenant. El estado de sesión vive en el objeto de contexto de sesión. El estado de pestaña vive en el formulario o la página que posee la pestaña. El estado de solicitud vive en el contexto de comando que viaja a lo largo de la llamada. Un valor guardado en un ámbito demasiado amplio se vuelve visible para las personas equivocadas, y un valor guardado en un ámbito demasiado estrecho se pierde cuando el usuario abre una segunda pestaña.

En la revisión, pregunta por cada campo: ¿quién más puede ver esto y qué ocurre cuando el usuario abre la misma pantalla dos veces?

Un `SessionContext` y un `CommandContext` conscientes del tenant

El `SessionContext` del laboratorio es un único objeto creado al iniciar sesión y guardado en la sesión. Contiene el ID de usuario, el nombre visible, los roles, los permisos, la referencia cultural y el tenant actual, resuelto a partir de claims verificados y del tenant que el usuario eligió entre aquellos a los que tiene derecho. Las pantallas lo leen para adaptar lo que muestran. Los servicios lo leen para decidir qué permiten. Nada lee la identidad del tenant de un control de la UI, porque el valor de una lista desplegable se puede cambiar desde el navegador.

El `CommandContext` es su compañero por solicitud. Cuando un manejador inicia un comando, construye un contexto a partir del contexto de sesión y le añade un nuevo ID de correlación, una marca de tiempo y el nombre del comando. Ese contexto se pasa explícitamente a los servicios, los repositorios y los trabajos en segundo plano, de modo que cada capa conoce el tenant y el ID de correlación sin tener que volver a consultar la sesión. El trabajo en segundo plano, en particular, no puede depender de la sesión, porque la sesión puede terminar antes que el trabajo.

La protección de tenant de cada servicio compara el tenant del contexto con el tenant del registro afectado y rechaza cualquier discrepancia antes de que se ejecute la lógica de negocio.

Concurrencia optimista y resolución de conflictos

Dos operadores de despacho pueden abrir la misma orden de trabajo en dos sesiones. Bloquear el registro para el primero deja al segundo esperando lo que dure una pausa para comer, así que las pantallas de edición empresariales usan concurrencia optimista. Cada registro lleva una versión (una columna de versión de fila o una marca de tiempo) que se lee al cargar el registro y se comprueba al guardarlo. Si la versión ha cambiado entretanto, otra sesión ha guardado antes y el guardado se rechaza.

El trabajo de diseño está en ese rechazo. Un simple mensaje de error deja al usuario con el trabajo perdido y sin saber por qué. El diálogo de conflicto del laboratorio explica que otra persona ha modificado el registro desde que se abrió y ofrece tres caminos: Reload descarta las modificaciones locales y muestra el registro actual, Compare muestra las dos versiones campo por campo para que el usuario pueda combinarlas, y Cancel deja la pantalla como está para que el usuario decida más tarde. La elección se registra en la pista de auditoría con el ID de correlación.

Las comprobaciones de concurrencia pertenecen al manejador de comandos, no al formulario, para que se aplique la misma regla cuando el guardado procede de una operación por lotes o de una importación.

Auditar el estado estático y compartido

Un proceso de servidor de Wisej.NET aloja todas las sesiones en la misma memoria. Por tanto, un campo estático, un servicio singleton o una caché sin clave de tenant los comparten a la vez todos los usuarios de todos los tenants. Un estático que guarda el usuario actual es la fuga clásica: la última sesión en iniciar sesión lo sobrescribe y la siguiente solicitud de otro usuario se ejecuta como esa persona. La auditoría del estado estático del laboratorio enumera cada campo estático, singleton y caché de la solución y responde a una pregunta para cada uno: ¿qué contiene y podría llegar a ser por usuario o por tenant?

La seguridad de subprocesos es la otra mitad. Las tareas en segundo plano iniciadas con `Application.StartTask`, los temporizadores y las solicitudes concurrentes desde varias pestañas pueden tocar los mismos objetos al mismo tiempo. Un diccionario compartido que se lee y se escribe sin sincronización se corrompe en silencio. La revisión de condiciones de carrera recorre cada objeto compartido y comprueba si su acceso está sincronizado o si no debería compartirse en absoluto.

- Cada estático contiene solo configuración o datos de referencia inmutables.
- Cada caché está indexada por tenant y sus entradas no se pueden leer entre tenants.
- Cada objeto mutable compartido tiene una estrategia de sincronización documentada.



Regla avanzada de producción

No te detengas cuando la pantalla parezca correcta. Explica a quién pertenece el estado, las implicaciones de seguridad, las rutas de fallo y el comportamiento en producción antes de dar el módulo por terminado.



Objetivos de aprendizaje

Al terminar este módulo serás capaz de:

- ✓ Diseñar de forma deliberada el estado de sesión, de usuario, de tenant, de pestaña y de solicitud.
- ✓ Implementar un `SessionContext` y un contexto de comando conscientes del tenant.
- ✓ Usar concurrencia optimista y resolución de conflictos en los flujos de edición.
- ✓ Auditar el estado estático y compartido en cuanto a seguridad de subprocesos y fugas entre tenants.

Guía del laboratorio / examen

Lo que construirás en el laboratorio práctico, el enfoque sugerido y los entregables requeridos.



Laboratorio práctico

Amplía EnterpriseOps con selección de tenant, servicios conscientes del tenant, IDs de correlación y concurrencia optimista en las órdenes de trabajo. Crea un diálogo de conflicto que explique las ediciones obsoletas y ofrezca los caminos Reload, Compare y Cancel.

Enfoque sugerido

- ✓ Lee el problema en producción e identifica el riesgo de arquitectura antes de tocar la UI.
- ✓ Crea o actualiza primero el límite correspondiente de servicio, comando, modelo o componente.
- ✓ Construye la pantalla o el componente Wisej.NET siguiendo los estándares de proyecto establecidos.
- ✓ Añade una demostración de la ruta de fallo, no solo del caso ideal.
- ✓ Anota lo que ha cambiado en el diario de aprendizaje y actualiza la lista de comprobación correspondiente.

Entregables requeridos

- ✓ SessionContext consciente del tenant
- ✓ CommandContext con ID de correlación
- ✓ Informe de auditoría del estado estático
- ✓ Implementación de concurrencia optimista
- ✓ Diálogo de resolución de conflictos

Preguntas de repaso

- ✓ ¿Se puede falsificar el ID de tenant desde un control de la UI?
- ✓ ¿Qué ocurre cuando la misma orden de trabajo se edita en dos sesiones?
- ✓ ¿Qué servicios se pueden compartir de forma segura?



Regla avanzada de producción

No te detengas cuando la pantalla parezca correcta. Explica a quién pertenece el estado, las implicaciones de seguridad, las rutas de fallo y el comportamiento en producción antes de dar el módulo por terminado.