

Sistemas en tiempo real, pipelines en segundo plano, notificaciones e importaciones

Guía de la lección · Guía del laboratorio / examen

● LearnWisej

Guía de la lección

Trabajos en segundo plano, progreso, notificaciones, cancelación, colas, server push. Lo que esto significa para un equipo avanzado de Wisej.NET — y cómo leer este módulo.



Enfoque de producción

Trabajos en segundo plano, progreso, notificaciones, cancelación, colas, push desde el servidor.

1. El problema en producción

Los sistemas avanzados de Wisej.NET suelen ejecutar importaciones, generación de informes, integraciones y notificaciones mientras los usuarios siguen trabajando. Estas canalizaciones deben ser observables, cancelables y seguras entre sesiones.

2. Conceptos clave

Concepto	Por qué importa
modelo de trabajo	Un registro persistido que describe una operación en segundo plano: su tipo, su entrada, su estado, su progreso, su propietario y su resultado, con independencia de cualquier pantalla abierta.
cola en segundo plano	Una cola duradera que acepta trabajos y los procesa fuera del ciclo de solicitud, para que el trabajo de larga duración nunca bloquee la interfaz de usuario.
Application.StartTask	La API de Wisej.NET para ejecutar trabajo en un subproceso en segundo plano que permanece vinculado al contexto de sesión, de modo que puede actualizar la UI de forma segura al terminar.
Application.Update	Envía de inmediato al navegador los cambios pendientes de la UI hechos desde un subproceso en segundo plano; así es como aparecen el progreso y las notificaciones sin que el usuario haga nada.
servicio de notificaciones	Un servicio central que entrega los resultados de los trabajos y las alertas a la sesión de usuario adecuada, tanto si la pantalla de origen sigue abierta como si no.
política de reintentos	Reglas sobre cuántas veces y con qué rapidez se reintenta un trabajo fallido, con espera progresiva, para que los errores transitorios se recuperen sin saturar los sistemas de destino.
idempotencia	Diseñar los trabajos para que ejecutarlos dos veces produzca el mismo resultado que ejecutarlos una vez, lo que hace seguros los reintentos y los mensajes duplicados.
auditoría de trabajos	Un registro de quién inició cada trabajo, cuándo se ejecutó y qué cambió, para que el trabajo en segundo plano rinda cuentas igual que el trabajo interactivo.

3. Desarrollo de la lección

Los trabajos sobreviven a las pantallas

Una importación de larga duración no debería depender de que un único manejador de clic de un botón siga vivo. La UI inicia u observa un trabajo; un servicio lo ejecuta; un almacén de trabajos registra el estado; los servicios de notificación publican las actualizaciones para las sesiones interesadas.

Envía estados significativos

Las actualizaciones en tiempo real deberían comunicar hitos significativos: en cola, iniciado, validado, importado, fallido, completado. Llamar a la actualización constantemente malgasta recursos y dificulta la depuración.

Separar la ejecución de la visualización

El Import Center del laboratorio tiene dos mitades que nunca comparten un objeto. Un servicio de trabajos ejecuta las importaciones: recibe una definición de trabajo, lee el archivo, valida y escribe las filas por lotes, y escribe su progreso y su resultado en un almacén de trabajos. La pantalla observa: inicia un trabajo

llamando al servicio y luego muestra lo que el almacén diga de él. La pantalla se puede cerrar, volver a abrir en una nueva sesión o abrirla otro usuario con el permiso adecuado, y el trabajo ni lo sabe ni le importa.

El modelo de trabajo es el contrato entre ambas mitades. Cada registro de trabajo contiene un identificador, el tipo de trabajo, quién lo inició y en qué tenant, su estado, un valor de progreso, un mensaje, marcas de tiempo y un resumen del resultado con los errores por fila. Como el registro está persistido, el estado sobrevive a un reinicio del servidor, y un ingeniero de soporte puede ver todos los trabajos del día sin preguntar a nadie.

La cola en segundo plano se sitúa entre las dos mitades. Iniciar un trabajo lo pone en la cola; un worker lo recoge. Eso es lo que hace que el subproceso que lo ejecuta sea independiente de la solicitud que lo pidió.

Actualizaciones en tiempo real con intención

Wisej.NET puede enviar cambios al navegador sin que haya una solicitud, y así es como aparece el progreso mientras el usuario espera. Una tarea en segundo plano iniciada con `Application.StartTask` conserva el contexto de sesión, modifica la barra de progreso y la etiqueta de estado, y llama a `Application.Update` para enviar esos cambios al navegador. Usado sin cuidado, el mismo mecanismo envía una actualización por cada fila importada y convierte una importación de dos minutos en una avalancha de viajes de ida y vuelta que ralentiza el servidor y el navegador.

El `ImportWorkOrdersJob` del laboratorio publica hitos en su lugar: en cola, iniciado, validado, cada lote completado, fallido, completado. Un receptor de hitos los recibe y decide si actualizar la UI ahora o esperar, de modo que un trabajo que produce mil eventos da lugar a un puñado de actualizaciones. Cada actualización lleva un estado significativo, que es lo que el usuario realmente quiere saber.

La segunda pregunta del laboratorio, con qué frecuencia se actualiza la UI, tiene una respuesta numérica en un buen diseño, y es como mucho un número pequeño por segundo.

Modelos de estado de trabajos y de notificaciones

El estado de un trabajo es una pequeña máquina de estados: en cola, en ejecución, completado, fallido, cancelado. El progreso es un valor aparte, porque un trabajo en ejecución al noventa por ciento y otro al diez por ciento tienen el mismo estado. El almacén de estados expone el registro actual y un historial, para que la pantalla de detalle del trabajo pueda mostrar cuándo se produjo cada transición y qué mensaje la acompañó.

Las notificaciones son la forma en que el resultado llega a una persona que ya no está mirando. Un servicio de notificaciones acepta un mensaje para un usuario o un rol, lo almacena y lo entrega a cualquier sesión activa de ese usuario a través de la campana del Import Center del laboratorio. Si el usuario no ha iniciado sesión, la notificación espera hasta que lo haga. Las notificaciones también son registros, con estado de leída y no leída, así que no se pierde nada cuando un navegador se cierra en el peor momento.

Ambos modelos son conscientes del tenant. Un trabajo pertenece a un tenant, sus notificaciones pertenecen a usuarios de ese tenant y el almacén nunca devuelve un trabajo más allá del límite del tenant.

Cancelación, reintentos y sesiones cerradas

Un botón de cancelar activa un indicador; por sí solo no detiene nada. El trabajo respeta la cancelación comprobando el indicador entre lotes, terminando limpiamente el lote actual, registrando lo que se ha hecho y pasando al estado cancelado. Una cancelación que se detiene a mitad de una fila deja datos escritos a medias, lo que es peor que terminar el lote. La política de reintentos distingue los errores transitorios, como un tiempo de espera agotado al escribir una fila, que se reintentan un número limitado de veces con una pausa, de los errores terminales, como un archivo mal formado, que hacen fallar el trabajo de inmediato.

El fallo parcial se notifica por fila. Una importación de diez mil filas con doce incorrectas se completa con un resultado que enumera esas doce, y el usuario corrige y vuelve a importar esas filas, no el archivo entero. El procesamiento idempotente de filas hace que esto sea seguro: importar la misma fila dos veces genera un solo registro.

El cierre de sesión es la ruta de fallo del laboratorio. El usuario cierra el navegador a mitad de la importación, el trabajo sigue ejecutándose porque pertenece al servicio y no a la sesión, y al volver a abrir la pantalla esta encuentra el trabajo en el almacén y muestra su estado final. ¿Quién es el propietario del trabajo después de cerrar la sesión? Solo hay una respuesta: el servicio.



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 canalizaciones en segundo plano que separen la ejecución de los trabajos de la visualización del progreso en la UI.
- ✓ Usar las actualizaciones en tiempo real de forma intencionada y evitar actualizaciones excesivas del cliente.
- ✓ Implementar modelos de notificaciones y de estado de trabajos.
- ✓ Gestionar la cancelación, los reintentos, los fallos parciales y el cierre de sesión.

Guía del laboratorio / examen

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



Laboratorio práctico

Construye un Import Center con una cola de trabajos, un observador de progreso, cancelación, procesamiento de filas reintentable, notificaciones y una pantalla de detalle del trabajo. Demuestra la recuperación tras cerrar y volver a abrir la sesión.

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

- ✓ Modelo de trabajo y almacén de estados
- ✓ Abstracción de la cola en segundo plano
- ✓ UI del observador de progreso
- ✓ Panel de notificaciones
- ✓ Política de reintentos y cancelación

Preguntas de repaso

- ✓ ¿Quién es el propietario del trabajo después de cerrar la sesión?
- ✓ ¿Con qué frecuencia se actualiza la UI?
- ✓ ¿Qué errores se pueden reintentar y cuáles son terminales?



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.