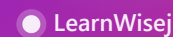


Eventos, manejadores y contratos de eventos de widget

Guía de la lección · Guía del laboratorio / examen



Guía de la lección

Convierte los callbacks del proveedor en eventos .NET estables sin trasladar la complejidad del proveedor a la aplicación. Qué significa esto para un desarrollador de integraciones con Wisej.NET — y cómo leer este módulo.



Enfoque de producción

Convierte los callbacks del proveedor en eventos .NET estables sin trasladar la complejidad del proveedor a la aplicación.

1. El problema de integración

Los widgets de terceros producen muchos eventos del navegador, pero la mayoría de las pantallas de una aplicación solo necesitan unos pocos eventos de servidor significativos con cargas útiles limpias.

2. Conceptos clave

Concepto	Cómo usarlo
fireWidgetEvent	Reenvía al servidor los eventos importantes del widget.
Contrato de carga útil	Envía al código .NET datos de evento pequeños y significativos.

3. Lección de producción

Integrar eventos no es un ejercicio mecánico de reenvío. Una cuadrícula puede producir decenas de eventos del navegador. Un gráfico puede generar callbacks de hover, clic, selección, zoom, renderizado, exportación y diseño. Enviarlos todos al servidor genera ruido y tráfico innecesario. Un buen wrapper elige los eventos que representan decisiones de negocio o cambios en el estado propiedad del servidor. Para `Wisej.Web.Widget`, `fireWidgetEvent(name, data)` es el camino directo hacia el `WidgetEvent` del servidor. En los controles personalizados, `fireEvent` y `fireDataEvent` pueden gestionarse mediante `OnWebEvent`, siempre que la configuración del servidor conecte el evento. La diferencia importa porque los controles personalizados pueden compartir nombres de eventos con los eventos integrados y deben pasar los eventos no gestionados a la implementación base. La carga útil del evento debe diseñarse a conciencia. En lugar de enviar el objeto de evento completo del proveedor, envía `rowKey`, `fieldName`, `value`, `selection` u otro objeto compacto sobre el que el servidor pueda actuar directamente. El servidor nunca debería tener que rebuscar en una estructura específica del proveedor para encontrar el único valor que le

interesa, y la carga útil debe ser lo bastante pequeña como para que lanzarla en cada interacción significativa no se convierta en un problema de rendimiento.

Conectar los eventos del proveedor en el momento adecuado

Un callback del proveedor solo puede asociarse cuando existe la instancia del proveedor, así que el lugar natural es dentro de `init`, justo después de crearla, mientras `this` sigue refiriéndose al widget de Wisej.NET. Algunas bibliotecas exponen una API de eventos en la instancia, otras aceptan callbacks en el objeto de opciones y unas pocas requieren asociarse a elementos DOM que ellas mismas crean. Cualquiera de estos enfoques funciona, siempre que el manejador capture la referencia al widget como enseñó el Módulo 2.

Las bibliotecas que recrean su instancia ante ciertos cambios de opciones necesitan volver a asociar los manejadores tras cada recreación; por eso el laboratorio mantiene la conexión en una única función a la que llaman tanto `init` como la ruta de recreación de `update`. Asociarlos solo en `init` dejaría muda la segunda instancia.

Cada asociación debería tener su desasociación en la ruta de eliminación del widget. Un manejador del proveedor que sobrevive a su widget retiene el widget en memoria y, tras suficientes cambios de pantalla, la fuga se hace visible como un navegador lento.

fireWidgetEvent hacia el servidor

Para un `Wisej.Web.Widget`, el método de cliente `fireWidgetEvent(name, data)` es el único camino que necesita una integración. El nombre es una cadena corta elegida por la integración, como `thresholdCrossed`, y los datos son cualquier objeto serializable. En el servidor, el `WidgetEvent` del widget se dispara con el nombre y los datos deserializados, y el manejador decide qué hacer según el nombre.

```
me.gauge.on('change', function (v) {
    if (v > options.threshold && !me.above) {
        me.above = true;
        me.fireWidgetEvent('thresholdCrossed', { value: v });
    }
});
```

El ejemplo muestra la disciplina importante: el cliente decide cuándo un evento es significativo y lo lanza una sola vez, en lugar de reenviar cada cambio y dejar que el servidor filtre. Una clase wrapper derivada de `Wisej.Web.Widget` puede gestionar `WidgetEvent` internamente y volver a lanzar un evento .NET tipado como `ThresholdExceeded`, de modo que el código de la aplicación nunca vea los nombres en forma de cadena.

fireEvent y fireDataEvent en los controles personalizados

Un control personalizado basado en el modelo de clases de cliente lanza los eventos de otra forma.

`fireEvent(name)` lanza un evento sin carga útil y `fireDataEvent(name, data)` lanza uno con carga útil. Como son eventos normales del lado del cliente, el servidor solo los recibe cuando la clase de servidor del control ha declarado su interés, lo que el laboratorio describe como conectar el evento en la configuración del servidor.

En el servidor, los eventos llegan a `OnWebEvent`, que recibe el nombre y los datos del evento. La sobrescritura gestiona los nombres que le corresponden y debe pasar todo lo demás a la implementación base, porque un control personalizado comparte el pipeline de eventos con los eventos integrados que su clase base ya gestiona, como el foco y el cambio de tamaño. Tragarse un evento desconocido rompe un comportamiento que no tiene nada que ver con la integración.

La diferencia entre ambos mecanismos es la autoevaluación del módulo. `WidgetEvent` es un único evento que lo recoge todo en un Widget; `OnWebEvent` es el punto de entrada general de eventos de un control, compartido con el framework, y por eso necesita la llamada a la base.

Cargas útiles en las que el código .NET puede confiar

La carga útil de un evento es un contrato, y el laboratorio lo registra en una tabla: nombre del evento, cada campo con su tipo, cuándo se dispara el evento y en qué evento .NET se convierte. El `pointClicked` del gráfico lleva el nombre de una serie, un valor x y un valor y. El `valueChanged` del mando giratorio lleva solo el nuevo valor. Nada en ninguna de las dos cargas útiles es un objeto del proveedor, un nodo DOM o una fila de datos completa.

Las cargas útiles pequeñas son estables entre actualizaciones del proveedor, porque se construyen a partir de unos pocos valores que extrae el wrapper, no de la forma del argumento de evento del proveedor. También son seguras: el servidor trata cada campo como no fiable, lo valida y busca por sí mismo cualquier dato sensible. Una clave de fila en una carga útil es una clave de búsqueda, nunca la prueba de que el usuario puede editar esa fila.

- ¿Contiene la carga útil solo primitivos y objetos pequeños?
- ¿Podría producir la misma carga útil otra biblioteca de proveedor?
- ¿Se valida cada campo en el servidor antes de usarlo?



Regla de producción

Mantén claras la propiedad del estado en Wisej.NET y el comportamiento del widget de cliente. Documenta el contrato cliente/servidor antes de confiar en la integración en producción.



Error habitual que debes evitar

Copiar JavaScript de demostración en las pantallas sin documentar el ciclo de vida, los recursos, los contratos de eventos, la limpieza ni los límites de seguridad.



Objetivos de aprendizaje

Al terminar este módulo serás capaz de:

- ✓ Conectar los eventos del proveedor durante la inicialización o después de crear la instancia.
- ✓ Usar `fireWidgetEvent` para reenviar eventos en `Wisej.Web.Widget`.
- ✓ Usar `fireEvent` y `fireDataEvent` con eventos conectados en los controles personalizados.

- ✓ Diseñar cargas útiles de eventos pequeñas, estables y significativas para el código .NET de la aplicación.

Guía del laboratorio / examen

Lo que construirás en el laboratorio práctico, el flujo de trabajo recomendado, los entregables requeridos y la autoevaluación antes de entregar.



Laboratorio práctico

Añade eventos de servidor para el cruce de umbral del indicador, el cambio de valor del mando giratorio y el clic en un punto del gráfico. Normaliza todas las cargas útiles de los eventos en DTO sencillos.

Flujo de trabajo recomendado

- ✓ Abre el proyecto existente del curso de integración de Wisej.NET y ejecútalo en local.
- ✓ Identifica el componente de servidor, el widget de cliente, el archivo de recursos o el endpoint que intervienen en este módulo.
- ✓ Haz un solo cambio pequeño cada vez y pruébalo en el navegador antes de continuar.
- ✓ Mantén el JavaScript específico del proveedor dentro del adaptador o wrapper en lugar de repartirlo por las páginas.
- ✓ Actualiza la interfaz para que el estudiante vea con claridad el éxito, los errores, el progreso o los cambios de comunicación.
- ✓ Toma capturas de pantalla o notas que demuestren que la integración funciona y expliquen la decisión de diseño.

Entregables requeridos

- ✓ Tres manejadores de eventos de cliente
- ✓ Tres manejadores de eventos de servidor
- ✓ Tabla del contrato de cargas útiles

Autoevaluación antes de entregar

- ✓ ¿Por qué no reenviar todos los eventos del proveedor?
- ✓ ¿Cuál es la diferencia entre `WidgetEvent`` y `OnWebEvent``?
- ✓ ¿Qué hace que la carga útil de un evento sea estable?