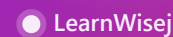


Postback, WebRequest, WebMethod y endpoints de datos

Guía de la lección · Guía del laboratorio / examen



Guía de la lección

Alimenta widgets de JavaScript que obtienen datos mediante Ajax, manteniendo las peticiones en el contexto del componente Wisej.NET correcto. Qué significa esto para un desarrollador de integraciones con Wisej.NET — y cómo leer este módulo.



Enfoque de producción

Alimenta widgets de JavaScript que obtienen datos mediante Ajax, manteniendo las peticiones en el contexto del componente Wisej.NET correcto.

1. El problema de integración

Muchos widgets avanzados, sobre todo cuadrículas, tablas dinámicas, informes, visores y componentes de subida de archivos, esperan URL o endpoints de servicio en lugar de datos enviados desde el servidor.

2. Conceptos clave

Concepto	Cómo usarlo
URL de postback	Encamina las peticiones del navegador de vuelta a la instancia de componente correcta.
WebMethod	Expone llamadas de estilo RPC para callbacks o promesas del cliente.

3. Lección de producción

Algunos widgets son más fáciles de integrar cuando obtienen los datos de una URL. Es habitual en cuadrículas de datos, tablas dinámicas, hojas de cálculo, visores, componentes de subida de archivos y bibliotecas con su propia abstracción de origen de datos. Wisej.NET proporciona una URL de postback ligada a la instancia del componente, de modo que la petición llega a la sesión correcta y al contexto de control correcto. Un manejador de postback se comporta como un pequeño servicio web. Recibe un HttpRequest y escribe un HttpResponse. La respuesta puede ser JSON para una cuadrícula, contenido binario para un archivo, datos de imagen para un visor o cualquier formato que requiera la biblioteca del proveedor. La ventaja es que el manejador vive con la instancia del componente, así que puede usar el estado del componente y los servicios de la aplicación. `[WebMethod]` tiene otra forma. Es ideal cuando el navegador necesita llamar a un método público del servidor y recibir un valor de retorno serializado. Puede ser estático o estar asociado a un contenedor de nivel superior. Suele ser más limpio para llamadas de tipo RPC, como buscar un registro, validar un valor o devolver un conjunto de resultados pequeño,

mientras que los manejadores de postback siguen siendo la mejor opción para archivos, imágenes o respuestas de datos grandes que una biblioteca de proveedor espera obtener de una URL.

Encaminar una URL a una instancia de componente

Muchos widgets avanzados obtienen sus propios datos. Una cuadrícula pide una página de filas, un visor transmite un documento, un componente de subida envía un archivo. Todos quieren una URL, y el reto es que esa URL debe llegar a la sesión de Wisej.NET correcta y, dentro de ella, a la instancia de componente correcta. `GetPostbackURL()` en el servidor, o `getPostbackUrl()` en el widget de cliente, devuelve exactamente una URL así, codificada con los identificadores que el framework necesita para entregar la petición a ese componente.

El wrapper pasa esa URL al proveedor como dirección de su origen de datos, normalmente en `init`, y puede añadir sus propios parámetros de consulta para distinguir acciones. Como la URL está ligada a una instancia, dos cuadrículas de la misma página obtienen dos URL distintas y cada petición llega al componente que es dueño de los datos.

Trata la URL como algo ligado a la sesión y de corta duración. No es la dirección de una API pública, no debe guardarse y no debe compartirse entre usuarios.

Gestionar la petición en WebRequest

Cuando llega una petición a la URL de postback, se dispara el evento `WebRequest` del componente con el `HttpRequest` y el `HttpResponse` en bruto. El manejador lee la acción y los parámetros, produce los datos, establece el content type y escribe la respuesta. JSON para una cuadrícula, un flujo PDF para un visor, una imagen para una miniatura o un CSV para una exportación son solo cuerpos distintos con content types distintos.

```
private void grid_WebRequest(object sender, WebRequestEventArgs e)
{
    var action = e.Request.QueryString["action"];
    if (action != "load") { e.Response.StatusCode = 400; return; }
    var rows = _service.LoadPage(CurrentTenant, ParsePage(e.Request));
    e.Response.ContentType = "application/json";
    e.Response.Write(JsonSerializer.Serialize(rows));
}
```

El manejador se ejecuta en el contexto del componente, así que puede leer el estado del componente y la sesión actual. Esa es toda su ventaja: la consulta de datos sabe qué tenant, qué filtros y qué permisos se aplican sin que nada de eso tenga que llegar desde el navegador.

WebMethod para llamadas de estilo RPC

Cuando quien llama es el propio JavaScript de la integración y no un origen de datos del proveedor, una URL no es la forma adecuada. Un método marcado con `[WebMethod]` puede llamarse desde el cliente como una función, con argumentos tipados convertidos desde JavaScript y un valor de retorno convertido de vuelta, y el cliente puede esperar (await) el resultado. Puede ser estático o un método de instancia de un contenedor de nivel superior, como la página o la ventana.

El laboratorio implementa el mismo conjunto de datos de solo lectura de las dos formas para que el equilibrio entre ventajas e inconvenientes sea visible. La URL de postback es la adecuada cuando la biblioteca del proveedor se encarga de obtener los datos y espera una dirección a la que llamar con sus propios argumentos de paginación. El WebMethod es el adecuado cuando el código del wrapper necesita un valor, como validar un código introducido u obtener una lista de consulta, y de otro modo tendría que analizar una respuesta HTTP en bruto.

Ambos son puntos de entrada al servidor y ambos se saltan el pipeline normal de eventos, así que los dos necesitan la misma validación y autorización que recibiría un manejador de botón.

Parámetros, content types y seguridad

Cada endpoint de datos acepta entrada del navegador, y cualquiera puede automatizar el navegador con scripts. El parámetro de acción se compara con una lista fija, no se usa para construir un nombre de método. Los argumentos de paginación se analizan como enteros con un límite superior, de modo que se rechaza una petición de un millón de filas. Los valores de filtro y ordenación se asignan a columnas conocidas en lugar de pasarse a una consulta. Cualquier cosa inesperada devuelve un error de cliente y se registra con los identificadores de sesión y de correlación.

El content type se establece de forma explícita en cada respuesta, porque un cuerpo JSON servido como texto confunde a algunas bibliotecas y un cuerpo HTML servido como JSON las confunde a todas. La respuesta de la autoevaluación es `application/json` para JSON. Las respuestas de archivos establecen la cabecera de disposición y el tipo MIME real para que el navegador las trate correctamente.

- ¿Se comprueba en el manejador que el usuario actual puede ver estas filas en este tenant?
- ¿Están acotados los parámetros de paginación y filtro, y asignados a una lista permitida?
- ¿Una respuesta de error no revela nada del servidor más allá de un código de estado?



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:

- ✓ Usar `getPostbackUrl` / `GetPostbackURL`` para encaminar las peticiones del navegador a una instancia de componente.
- ✓ Gestionar las peticiones con `WebRequest`` y devolver JSON, archivos, imágenes o cualquier respuesta que espere el widget.

- ✓ Exponer métodos `[WebMethod]` para las llamadas de cliente que deban comportarse como RPC y no como peticiones HTTP en bruto.
- ✓ Añadir parámetros de acción, validación, content types y comprobaciones de seguridad a las peticiones de datos de los widgets.

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

Implementa un endpoint de postback y un endpoint WebMethod para el mismo conjunto de datos de solo lectura. Conecta un widget sencillo a cada uno y compara sus ventajas e inconvenientes.

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

- ✓ Manejador WebRequest de postback
- ✓ WebMethod con argumentos y valor de retorno
- ✓ Nota comparativa: origen de datos por URL frente a callback de cliente

Autoevaluación antes de entregar

- ✓ ¿Cuándo es mejor el postback que un WebMethod?
- ✓ ¿Por qué deben validarse los parámetros de la acción?
- ✓ ¿Qué content type deberían usar las respuestas JSON?