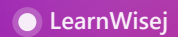


Dominio del DataGridView

Guía de la lección · Guía del laboratorio / examen



Guía de la lección

BindingSource y columnas explícitas, editores personalizados y celdas de comando, modo virtual con CellValueNeeded y una caché en el servidor, y HTML en celdas, controles en celdas y dibujo de celdas usados con responsabilidad. Qué significa esto para un desarrollador de Wisej.NET y cómo leer este módulo.



Enfoque de producción

BindingSource y columnas explícitas, editores personalizados y celdas de comando, modo virtual con CellValueNeeded y una caché en el servidor, y HTML en celdas, controles en celdas y dibujo de celdas usados con responsabilidad.

1. El problema en producción

DataGridView es el control que con más probabilidad decide si una aplicación de negocio parece madura. Pedidos, facturas, plazos, logs, usuarios y documentos acaban todos en una cuadrícula, y los usuarios juzgan la aplicación por cómo esa cuadrícula carga, se desplaza, edita e informa del estado. Wisej.NET ofrece de todo, desde el rellenado manual hasta el modo virtual y el dibujo de celdas; dominarlo consiste en elegir la técnica adecuada para el volumen de datos y la tarea del usuario.

2. Conceptos clave

Concepto	Por qué importa
BindingSource	Un intermediario en el servidor entre las filas del view model y la cuadrícula, que le da un origen de datos estable, posición actual y ordenación sin acoplar la pantalla a la base de datos.
AutoGenerateColumns	Crea una columna por cada propiedad pública del tipo enlazado; útil para un prototipo, peligroso en producción porque un cambio en el modelo reorganiza la pantalla sin avisar.
columna explícita	Una columna que defines con enlace de campo, texto de cabecera, anchura, alineación, formato, estado de solo lectura, modo de ordenación y tipo de editor, para que la interfaz siga estable cuando cambie el modelo.
modo virtual	Con <code>VirtualMode</code> activado, la cuadrícula solo conoce su número de filas y pide cada valor visible mediante <code>CellValueNeeded</code> , así que no se crea nada para las filas que nadie ve.
caché en el servidor	Un búfer del tamaño de una página que alimenta <code>CellValueNeeded</code> para que el desplazamiento no consulte la base de datos por cada celda; el evento <code>DataRead</code> es el punto de enganche documentado para construirla y mantenerla.
editor de celda personalizado	Un control asignado a la propiedad <code>Editor</code> de una columna; debe devolver su valor a la celda, mediante <code>CellBeginEdit</code> y <code>CellEndEdit</code> si no tiene un evento de cambio adecuado.
HTML en celda frente a control en celda	El HTML producido en <code>CellFormatting</code> con <code>AllowHtml</code> es markup solo de visualización y casi gratuito; un control incrustado es un control de servidor y un widget del navegador reales, con un coste por celda.

3. Desarrollo de la lección

Tres formas de rellenar, una para producción

La cuadrícula puede mostrar datos con o sin un origen de datos subyacente. El acceso directo a `Rows` y `Columns` es lo más sencillo: añades columnas y filas y eres dueño de cada valor; conviene para una tabla pequeña y estática, como una tabla de consulta de estados. El enlace con `DataSource` y `DataMember` rellena la cuadrícula a partir de una lista, un `BindingSource` o un conjunto de datos, y es el modo para los datos de negocio. El tercer modo es la generación dinámica de columnas: con `AutoGenerateColumns` en true, la cuadrícula reconstruye sus columnas cada vez que cambia el origen de datos.

Las columnas generadas sirven para un prototipo de diez minutos y son una trampa en una pantalla que se publica: añade una propiedad a `OrderRow`, renombra otra, y la cuadrícula sigue al modelo en lugar de al diseño. Por eso la cuadrícula Orders de Operations Console define cada columna explícitamente: `DataPropertyName`, `HeaderText`, `Width`, alineación y formato en `DefaultCellStyle`, `ReadOnly` donde el valor es informativo, `SortMode` y el tipo de columna que incorpora el editor adecuado. El enlace en sí sigue siendo pequeño:

```
private readonly BindingSource ordersSource = new BindingSource();

private void BindOrders(IEnumerable<OrderRow> rows)
{
    this.ordersGrid.AutoGenerateColumns = false;    // columns are designed, not discovered
    this.ordersSource.DataSource = rows.ToList();
    this.ordersGrid.DataSource = this.ordersSource;
}

private void ordersGrid_CellFormatting(object sender, DataGridViewCellFormattingEventArgs e)
{
    if (ordersGrid.Columns[e.ColumnIndex].Name != "StatusColumn") return;
    var status = System.Net.WebUtility.HtmlEncode(Convert.ToString(e.Value));
    e.Value = $"<span class='status-badge'>{status}</span>";
    e.FormattingApplied = true;
}
```

Desactiva `AutoGenerateColumns` en cuanto la pantalla sea estable y enlaza una colección de view models en lugar de una entidad: un view model tiene la forma pensada para la visualización y lleva el texto de estado, un total formateado y un ID sin filtrar detalles de persistencia.

Columnas integradas, editores personalizados y celdas de comando

Los valores con tipo merecen columnas con tipo. Los tipos de columna de texto, casilla, combo, botón, enlace e imagen cubren la mayor parte de la cuadrícula Orders, y Wisej.NET añade columnas basadas en los editores de fecha, numéricos y con máscara del Módulo 2. Una celda de comando es una columna de botón o de enlace: la columna *Ship* del laboratorio lanza `CellContentClick`, y el manejador lee el ID de la fila desde el elemento enlazado y llama al servicio. Nada del pedido vive en la celda.

Cuando ningún editor integrado encaja, asigna un control a la propiedad `Editor` de la columna; el ejemplo clásico es un `MonthCalendar` en una celda de fecha de vencimiento. La advertencia importante: el editor debe actualizar correctamente el valor de la celda. La cuadrícula espera `TextChanged` o un evento de cambio equivalente; si tu control no lanza ninguno, gestiona `CellBeginEdit` para copiar el valor de la celda en el editor y `CellEndEdit` para devolver el valor editado. Prueba el viaje de ida y vuelta con Tab, Enter y un clic en otro sitio: un editor que nunca confirma el valor es el error clásico de los editores personalizados.

Modo virtual, CellValueNeeded y la caché

Los grandes volúmenes de datos necesitan otro enfoque. Cargar 100.000 pedidos porque la base de datos puede devolverlos no es una estrategia de interfaz: nadie puede revisar tantas filas, y cada una cuesta memoria en el servidor y renderizado en el navegador. Filtra primero, ofrece búsqueda y paginación en el servidor, y reserva el modo virtual para la ruta en la que el usuario de verdad debe desplazarse por un resultado muy grande. Ahí estableces `VirtualMode` en true, das a la cuadrícula su `RowCount` y sus columnas, y gestionas `CellValueNeeded` para proporcionar cada valor cuando se pide.

El evento por sí solo no basta. Un manejador que consulta la base de datos dentro de `CellValueNeeded` se ejecuta una vez por cada celda visible, así que un desplazamiento puede producir cientos de consultas. El `OrderCache` del laboratorio mantiene la página actual en el servidor: `CellValueNeeded` le pide fila y

columna, la caché atiende los aciertos desde memoria y obtiene una página entera cuando falta el dato. La documentación de Wisej.NET señala el evento `DataRead` como el lugar para construir y mantener esa caché, porque te indica qué bloque de filas está a punto de leer la cuadrícula. Perfila con volúmenes realistas, no con las veinte filas del servicio de demostración, y pregúntate qué falla primero con diez veces los datos.

HTML en celdas, controles en celdas, dibujo de celdas y la cuadrícula compuesta

El HTML en celdas es la mejor opción intermedia para enriquecer la visualización. Si un estado necesita una insignia, un icono, una barra de progreso o varios fragmentos de texto, establece `AllowHtml` y genera el markup en `CellFormatting`, como en el fragmento anterior; el estado sin procesar se queda en el modelo y el HTML es solo de visualización. Trátalo como tal: codifica el texto del usuario antes de que entre en el markup para que el nombre de un cliente nunca se convierta en HTML ejecutable, y mantén la validación de negocio fuera del manejador de formato.

Los controles en celdas son más potentes y más peligrosos. Wisej.NET puede colocar un control real dentro de una celda, incluso abarcando varias: un botón en una fila, un track bar a lo largo de columnas, una minivista previa. Cada uno es un control de servidor y un widget del navegador, así que un control en cada celda de una cuadrícula de 10.000 filas son 10.000 controles. Úsalos para la fila seleccionada, filas de resumen, un área de comandos o una cuadrícula pequeña. El dibujo de celdas está en el extremo de la escala: activa el dibujo de usuario en la celda y gestiona `CellPaint` cuando la visualización necesite un dibujo personalizado o reutilice código de dibujo .NET existente, no para estilos que ya cubren los temas, los estilos de celda o el HTML.

Por último, el `DataGridView` componible se comporta como un contenedor que aloja controles conectados, como un panel de filtro, una barra de estado, un navegador o un árbol, acoplados dentro o alrededor de él y ordenados con `BringToFront` y `SendToBack`. Así toda la API de la cuadrícula sigue visible en lugar de quedar oculta tras un UserControl; la barra de filtro y la de estado del laboratorio se construyen de esta forma.

- Columnas explícitas y un BindingSource o view model en cada pantalla de producción; AutoGenerateColumns desactivado en cuanto la pantalla es estable.
- Filtrar antes de mostrar; modo virtual y caché para los conjuntos realmente grandes.
- HTML en celdas antes que controles incrustados; controles incrustados con moderación; dibujo de celdas solo cuando los estilos, el HTML y los controles normales no bastan.
- Perfilar con datos realistas, no con datos de ejemplo.



Errores comunes

Dejar `AutoGenerateColumns` activado en producción. Cargar todos los registros al arrancar. Consultar la base de datos desde `CellValueNeeded` por cada celda sin caché. Incrustar controles en cada fila de una cuadrícula enorme. Mezclar la validación de negocio con el formato visual. Tratar una cuadrícula como una tabla de base de datos en lugar de como una vista orientada a tareas.



Regla de la biblioteca de controles

Elige el control nativo más sencillo que exprese la tarea del usuario antes de escribir código personalizado. Los controles nativos ya te dan soporte en tiempo de diseño, temas, eventos y estado en el servidor; recurre después a un extender o a un UserControl, y construye un widget JavaScript solo cuando ningún control nativo encaje.



Objetivos de aprendizaje

Al terminar este módulo serás capaz de:

- ✓ Enlazar un DataGridView a un BindingSource y controlar AutoGenerateColumns de forma deliberada.
- ✓ Usar columnas integradas y editores personalizados para valores con tipo y celdas de comando.
- ✓ Implementar el modo virtual, CellValueNeeded, los conceptos de DataRead/caché y los límites de rendimiento con grandes volúmenes de datos.
- ✓ Usar con responsabilidad el HTML en celdas, los controles en celdas, el dibujo de celdas y las cuadrículas compuestas.

Guía del laboratorio / examen

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



Laboratorio práctico

Construye la cuadrícula Orders de Operations Console: un DataGridView enlazado a un BindingSource con columnas explícitas y AutoGenerateColumns bajo control, una celda de estado renderizada con AllowHtml y CellFormatting, un editor personalizado o una columna de comando, una versión en modo virtual o respaldada por caché para la ruta de grandes volúmenes de datos, y una barra compuesta de filtro y estado acoplada dentro o alrededor de la cuadrícula.

Punto de partida

Abre la solución `OperationsConsole` del Módulo 4 y selecciona la página provisional DataGridView del shell. El explorador de documentos, el `CustomerEditor` y el shell de navegación ya existen; la página Orders sigue siendo una `TabPage` vacía, y basta con un pequeño `OrderService` en memoria que devuelva view models `OrderRow` para llevar a cabo cada paso de este laboratorio.

Enfoque sugerido

- ✓ Añade `ordersGrid` y un BindingSource `ordersSource`, establece `AutoGenerateColumns` en false y define cada columna explícitamente: número de pedido, cliente, fecha de vencimiento, total y estado, con texto de cabecera, anchura, alineación, formato y estado de solo lectura elegidos a propósito.
- ✓ Marca la columna de estado con `AllowHtml` y gestiona `CellFormatting` para convertir el estado sin procesar en una insignia codificada; no modifiques el valor del modelo y establece `FormattingApplied`.
- ✓ Añade o bien un editor MonthCalendar en la columna de fecha de vencimiento mediante la propiedad `Editor` de la columna (con `CellBeginEdit` / `CellEndEdit` si el valor no hace el viaje de ida y vuelta), o bien una columna de botones cuyo `CellContentClick` llame al servicio por ID de pedido.
- ✓ Construye la ruta de grandes volúmenes de datos: `VirtualMode` en true, `RowCount` desde el servicio y un `OrderCache` que atienda `CellValueNeeded` desde una página en memoria y obtenga una página nueva cuando falte el dato, usando `DataRead` para precargar el bloque que la cuadrícula está a punto de mostrar.
- ✓ Compón la cuadrícula: acopla una barra de filtro (cuadro de búsqueda, ComboBox de estado, botón Apply) encima o dentro de la cuadrícula y una barra de estado debajo que muestre el número de filas, el pedido seleccionado y la última actualización; ordénalas con `BringToFront` y `SendToBack`.

Entregables requeridos

- ✓ DataGridView Orders con columnas explícitas enlazado a un BindingSource
- ✓ Celda de estado renderizada con AllowHtml y CellFormatting

- ✓ Un editor personalizado o una columna de comando
- ✓ Versión de la cuadrícula en modo virtual o respaldada por caché para grandes volúmenes de datos
- ✓ Barra compuesta de filtro y estado acoplada dentro o alrededor de la cuadrícula

Preguntas de repaso

- ✓ Si `OrderRow` gana dos propiedades nuevas en el próximo sprint, ¿qué cambia en la pantalla con columnas generadas y qué cambia con tus columnas explícitas?
- ✓ Tu manejador de `CellValueNeeded` funciona con 200 pedidos. ¿Qué le ocurre a la base de datos cuando un usuario arrastra la barra de desplazamiento a través de 200.000, y qué parte de tu diseño lo absorbe?
- ✓ ¿Por qué la insignia de estado corresponde a `CellFormatting` y no al modelo `OrderRow`, y qué saldría mal si se colocara el nombre de un cliente en ese HTML sin codificar?



Regla de la biblioteca de controles

Elige el control nativo más sencillo que exprese la tarea del usuario antes de escribir código personalizado. Los controles nativos ya te dan soporte en tiempo de diseño, temas, eventos y estado en el servidor; recurre después a un extender o a un UserControl, y construye un widget JavaScript solo cuando ningún control nativo encaje.