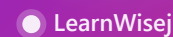


Fundamentos del diseño: Dock, Anchor, AutoSize, AutoScroll y composición del shell

Guía de la lección · Guía del laboratorio / examen



Guía de la lección

El motor de diseño predeterminado, el orden de acoplamiento y el Padding, el anclaje para regiones de tipo formulario, AutoSize y AutoScroll usados con seguridad, y la composición del shell con contenedores anidados y UserControls en lugar de manejadores de redimensionado. Qué significa esto para un desarrollador de Wisej.NET y cómo leer este módulo.



Enfoque de producción

El motor de diseño predeterminado, el orden de acoplamiento y el Padding, el anclaje para regiones de tipo formulario, AutoSize y AutoScroll usados con seguridad, y la composición del shell con contenedores anidados y UserControls en lugar de manejadores de redimensionado.

1. El problema en producción

Adaptive Operations Console ya tiene un tema y CSS con ámbito, pero su shell sigue moviendo controles a mano: un manejador **Resize** recalcula **Bounds** para la barra de herramientas, el carril de navegación, la cuadrícula de tickets y el editor de detalles cada vez que el navegador cambia de tamaño. Funciona con el ancho que probó el desarrollador, se rompe con cualquier otro y no puede sobrevivir a los perfiles de tableta y teléfono que introduce el Módulo 6. El diseño adaptable en Wisej.NET depende de contenedores y reglas de diseño, no de coordenadas: regiones que ocupan bordes, un área de trabajo que rellena lo que queda y editores que se desplazan en lugar de superponerse.

2. Conceptos clave

Concepto	Por qué importa
motor de diseño predeterminado	Cada control de Wisej.NET tiene un motor de diseño; el predeterminado admite posición absoluta, acoplamiento y anclaje, de modo que una pantalla puede combinar la colocación precisa del diseñador con regiones que se redimensionan solas.
prioridad de acoplamiento	Los hijos acoplados ocupan espacio en el orden en que aparecen en la colección de controles del padre, y por eso los mismos cinco paneles producen un shell correcto o uno con superposiciones según el orden de los hijos.
DisplayRectangle y Padding	Los controles acoplados se colocan dentro del <code>DisplayRectangle</code> del padre, que el <code>Padding</code> reduce; el motor predeterminado ignora <code>Margin</code> en los hijos acoplados, así que los espacios vienen del padding o de controles separadores.
anclaje	Un anclaje mantiene la distancia de un control a los lados seleccionados del padre, lo que encaja con regiones de tipo formulario: etiquetas arriba a la izquierda, cuadros de texto de izquierda a derecha, botones de comando abajo a la derecha.
AutoSize y diseño circular	Un contenedor que calcula su tamaño a partir de hijos acoplados o anclados a él no tiene una respuesta estable; ese bucle es la principal advertencia del módulo sobre AutoSize.
AutoScroll con barras ocultas	<code>AutoScroll</code> permite que un contenedor se desplace cuando el contenido no cabe, y <code>ScrollBars.Hidden</code> mantiene el desplazamiento táctil en los teléfonos sin dibujar barras de desplazamiento.
región UserControl	Una región del shell con su propio diseño local y una superficie pública reducida, de modo que el shell muestra, oculta, acopla o mueve la región completa según el perfil en lugar de conocer sus coordenadas internas.

3. Desarrollo de la lección

Acoplamiento, orden de los hijos y el espacio que queda

El acoplamiento une un hijo a un borde de su padre (`DockStyle.Top` , `Bottom` , `Left` , `Right`) o hace que ocupe lo que quede (`DockStyle.Fill`). El motor reparte el espacio entre los hijos acoplados uno a uno, siguiendo su orden en la colección del padre, y cada hijo acoplado reduce el rectángulo disponible para el siguiente. Ese es todo el secreto del shell de la consola: la barra de herramientas toma una franja de arriba, la región de estado una de abajo, el carril de navegación una de la izquierda, el panel de detalles una de la derecha, y el área de trabajo rellena lo que queda. Cambia el orden y el panel de estado tapaná la última fila de la cuadrícula. En el diseñador de Visual Studio, `BringToFront` , `SendToBack` y las operaciones sobre el índice de los hijos cambian ese orden, así que cuando un shell se vea mal comprueba el orden de los hijos antes de tocar un tamaño. El código del shell del paquete es deliberadamente breve, y el laboratorio lo reproduce en el diseñador:

```
// MainPage.cs – the console shell, one Dock per region
this.toolbarPanel.Dock = DockStyle.Top;    this.toolbarPanel.Height = 56;
this.statusPanel.Dock = DockStyle.Bottom;  this.statusPanel.Height = 28;
this.navigationPanel.Dock = DockStyle.Left; this.navigationPanel.Width = 240;
this.detailsPanel.Dock = DockStyle.Right;  this.detailsPanel.Width = 360;

this.workspacePanel.Dock = DockStyle.Fill;
this.workspacePanel.MinimumSize = new System.Drawing.Size(320, 240);

// gaps between docked regions come from the container, not from Margin
this.Padding = new Padding(8);
this.workspacePanel.Padding = new Padding(12, 8, 12, 8);
```

Los hijos acoplados se colocan dentro del `DisplayRectangle` del padre, y el `Padding` reduce ese rectángulo por todos los lados. El motor de diseño predeterminado no usa `Margin` en los controles acoplados: un margen en el carril de navegación no hace nada con el espacio que tiene al lado, así que da padding al contenedor o acopla un panel separador fino. El anclaje se ocupa de los controles dentro de una región. Un anclaje conserva la distancia a los lados seleccionados del padre: las etiquetas arriba a la izquierda para que no se muevan, los cuadros de texto de izquierda a derecha para que crezcan con el panel de detalles, el editor de notas ancho arriba, izquierda y derecha, y los botones Save y Cancel abajo a la derecha. Un control sin anclaje en un eje flota y mantiene una posición proporcional, lo adecuado para centrar una tarjeta de inicio de sesión y lo inadecuado para una cuadrícula.

Contenedores en lugar de manejadores de redimensionado

El `MainPage_Resize` de la consola es la mala práctica que este módulo existe para eliminar. Cada línea que asigna `Bounds`, `Location` o `Size` en respuesta a un redimensionado es una regla que el motor de diseño podría haber aplicado por ti, y a diferencia del motor se ejecuta después de que el navegador haya pintado el diseño incorrecto, no sabe nada del padding y no se compone: en cuanto el panel de detalles se convierte en un `UserControl`, el manejador tiene que conocer también sus detalles internos. La refactorización es mecánica: enumera las regiones, decide qué borde ocupa cada una, añádelas en ese orden con `Dock` y elimina el manejador. Las coordenadas absolutas siguen siendo útiles para el trabajo preciso del diseñador dentro de un diálogo de tamaño fijo; son la herramienta equivocada para una pantalla que también debe funcionar en un teléfono.

Un shell duradero es un conjunto de regiones anidadas: barra de comandos arriba, navegación a la izquierda, área de trabajo en el centro, detalles a la derecha y estado abajo. Cada región es un `Panel`, `UserControl`, `TabControl`, `SplitContainer` o `Page` con su propio diseño local, y el anidamiento es lo que hace estable el shell: al área de trabajo no le importa cómo organiza la barra de herramientas sus botones, y a la barra de herramientas no le importa que el área de trabajo contenga tarjetas de métricas sobre una cuadrícula de tickets. Anclar muchos controles con combinaciones complicadas también es una señal: un `TableLayoutPanel` o un `FlexLayoutPanel`, ambos tratados en el Módulo 5, expresan campos alineados o regiones proporcionales con más claridad que doce ajustes de anclaje.

AutoSize, AutoScroll y las barreras de protección del tamaño

`AutoSize` es valioso cuando el contenido decide el tamaño: una etiqueta que debe ajustarse al texto traducido, un badge que crece con su número, una tarjeta que envuelve contenido dinámico;

`AutoSizeMode` decide si el control solo puede crecer o también encogerse. El peligro es el diseño circular. Si un contenedor ajusta su tamaño a partir de hijos acoplados o anclados a ese mismo contenedor, el padre pregunta a los hijos qué tamaño deben tener, los hijos preguntan al padre qué tamaño tienen, y el motor no tiene una respuesta estable. El accidente típico es un panel con `AutoSize` que además rellena a su padre. Mantén el ajuste automático de tamaño en los controles hoja y en contenedores cuyos hijos tienen tamaños fijos o dependientes del contenido.

`AutoScroll` es la respuesta segura cuando el contenido no cabe: el contenedor se desliza en lugar de estirar el shell. El editor de detalles de la consola tiene más campos de los que puede mostrar la pantalla de un teléfono, así que el laboratorio activa `AutoScroll` y, en anchos estrechos, establece `ScrollBars` en `Hidden`, lo que mantiene el desplazamiento táctil sin dibujar barras de desplazamiento que roben ancho. `AutoScrollMargin` añade espacio después del último control y `AutoScrollMinSize` garantiza un área desplazable mínima. `MinimumSize` y `MaximumSize` son las otras barreras de protección: un `MinimumSize` en el área de trabajo y en la cuadrícula significa que la región de relleno nunca puede colapsarse en una franja cuando el navegador se estrecha, y un `MaximumSize` evita que el panel de detalles se trague un monitor ultraancho.

Regiones UserControl que el shell puede reorganizar

El último paso del laboratorio es el que da sus frutos en el Módulo 6. Una vez que el carril de navegación y el editor de detalles son `UserControl`, cada uno con su propio acoplamiento y anclaje internos, el shell deja de conocer las coordenadas internas de cualquier editor. Conoce cinco regiones y, según el perfil, muestra, oculta, acopla o mueve regiones completas: el carril puede convertirse en una estrecha tira de iconos, el editor de detalles puede pasar bajo la cuadrícula en una tableta o abrirse en un `Form` modal en un teléfono, y ninguno de esos movimientos toca un cuadro de texto. La región expone unos pocos miembros significativos, como una propiedad `Ticket` y un evento `Saved`, y mantiene privados sus controles hijos para que un cambio de nombre interno nunca rompa la página.

La lista de verificación de diseño del módulo es breve: prefiere contenedores a manejadores de redimensionado, comprueba el orden de acoplamiento, pon `Padding` en las regiones acopladas, asigna `MinimumSize` a las regiones de relleno y deja que el contenido que no cabe se desplace. La lista de malas prácticas de refactorización del paquete es la imagen especular de esa lista, y revisar una pantalla con ella es un buen hábito antes de cada prueba de perfiles.



Lista de malas prácticas de refactorización

Código `Resize` manual que asigna `Bounds` repetidamente a muchos controles. `Dock` y `Anchor` mezclados en controles donde entran en conflicto. Coordenadas absolutas en una pantalla que debe funcionar en teléfonos. Cuadrículas sin `MinimumSize`. Paneles con `AutoSize` y acoplamiento `Fill` que crean bucles. Formularios que ocultan controles en lugar de mover el contenido de baja prioridad a una sección desplazable.



Regla del sistema visual

Usa el mecanismo de Wisej.NET responsable del problema: el tema para la identidad de los controles estándar, AppearanceKey para las variantes semánticas, CssClass para CSS específico de la aplicación con ámbito, CssStyle solo para valores dinámicos puntuales, contenedores de diseño en lugar de código de redimensionado, propiedades responsive para los valores por perfil definidos en el diseñador, y ResponsiveProfileChanged para el comportamiento que no es simplemente visual.



Objetivos de aprendizaje

Al terminar este módulo serás capaz de:

- ✓ Usar el acoplamiento y el anclaje de forma intencionada y entender cómo afecta el orden de los hijos a la prioridad de acoplamiento.
- ✓ Componer pantallas a partir de contenedores anidados en lugar de manejadores de redimensionado y actualizaciones manuales de coordenadas.
- ✓ Usar con seguridad AutoSize, AutoSizeMode, AutoScroll, ScrollBars, MinimumSize, MaximumSize, Padding y Margin.
- ✓ Crear regiones UserControl reutilizables que puedan reorganizarse entre los perfiles de escritorio, tableta y teléfono.

Guía del laboratorio / examen

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



Laboratorio práctico

Refactoriza el shell de Adaptive Operations Console en contenedores anidados: una barra de herramientas superior, un carril de navegación a la izquierda, un área de trabajo de relleno, un panel de detalles a la derecha y una región de estado, cada uno un Panel o UserControl con su propio diseño local. Elimina todos los manejadores Resize que asignan Bounds, corrige el orden de acoplamiento en el diseñador, usa Padding para los espacios entre regiones acopladas, asigna un MinimumSize al área de trabajo y a la cuadrícula, y haz que el editor de detalles se desplace con AutoScroll y barras de desplazamiento ocultas en anchos estrechos.

Punto de partida

Abre la solución `AdaptiveOperationsConsole` tal como la dejaste tras el Módulo 3: la `Page` principal lleva el tema AdaptiveOps, la hoja de estilos con ámbito y el cambio de tema por sesión, pero sus cinco regiones siguen siendo paneles simples cuyos `Bounds` se recalculan en `MainPage_Resize`, y el editor de detalles se superpone a la cuadrícula de tickets por debajo de unos 1100 píxeles. Ejecútala una vez con ancho de escritorio, reduce después el navegador y anota qué se recorta, se superpone o desaparece antes de cambiar nada.

Enfoque sugerido

- ✓ Enumera todos los métodos `Resize`, `Layout` o de carga que asignan `Bounds`, `Location` o `Size`, anota en tus notas del laboratorio las cinco regiones y el borde que ocupa cada una, y elimina ese código antes de tocar el diseñador.
- ✓ En el diseñador asigna `Dock` a los cinco paneles (barra de herramientas Top, estado Bottom, navegación Left, detalles Right, área de trabajo Fill) y corrige el orden de los hijos con `BringToFront` y `SendToBack` hasta que los bordes se ocupen primero y el área de trabajo tome lo que queda.
- ✓ Asigna a la página y a las regiones un `Padding` para los espacios entre las zonas acopladas, establece `MinimumSize` en el área de trabajo y en la cuadrícula de tickets, y confirma que el motor predeterminado ignora `Margin` en los hijos acoplados.
- ✓ Dentro del panel de detalles ancla las etiquetas arriba a la izquierda, los editores de izquierda a derecha, el editor de notas arriba, izquierda y derecha, y los botones de comando abajo a la derecha; activa `AutoScroll`, establece `ScrollBars` en `Hidden` para los anchos estrechos y ajusta `AutoScrollMargin`.
- ✓ Extrae `NavigationRail` y `DetailsEditor` como `UserControl` con su propio diseño local y una superficie pública reducida, vuelve a colocarlos en el shell y redimensiona el navegador del ancho de escritorio al de teléfono para verificar que nada se superpone ni se recorta.

Entregables requeridos

- ✓ Shell reconstruido con contenedores anidados y regiones de barra de herramientas, navegación, área de trabajo, detalles y estado
- ✓ Eliminado todo el código Resize manual que asignaba Bounds
- ✓ Orden de acoplamiento, Padding y MinimumSize definidos de forma intencionada en cada región
- ✓ Editor de detalles que se desplaza mediante AutoScroll con barras de desplazamiento ocultas en anchos estrechos
- ✓ Regiones de navegación y detalles extraídas a UserControls

Preguntas de repaso

- ✓ Si en el orden de los hijos el panel del área de trabajo se acoplara antes que el panel de estado, ¿qué vería el usuario en la parte inferior de la cuadrícula, y qué comando del diseñador lo corrige sin cambiar ningún tamaño?
- ✓ Has asignado un `Margin` de 12 píxeles al carril de navegación y el espacio no ha aparecido. ¿Por qué no, y qué dos mecanismos sí producen un espacio entre regiones acopladas?
- ✓ El editor de detalles es un `UserControl` con `AutoScroll`. ¿Qué miembros suyos debe conocer el shell para moverlo bajo la cuadrícula en una tableta, y cuáles no debe necesitar nunca?



Regla del sistema visual

Usa el mecanismo de Wisej.NET responsable del problema: el tema para la identidad de los controles estándar, `AppearanceKey` para las variantes semánticas, `CssClass` para CSS específico de la aplicación con ámbito, `CssStyle` solo para valores dinámicos puntuales, contenedores de diseño en lugar de código de redimensionado, propiedades `responsive` para los valores por perfil definidos en el diseñador, y `ResponsiveProfileChanged` para el comportamiento que no es simplemente visual.