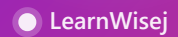


CSS, CssClass, CssStyle, estados y cambios de tema en tiempo de ejecución

Guía de la lección · Guía del laboratorio / examen



Guía de la lección

Añadir CSS específico de forma segura con `CssClass` y una hoja de estilos de la aplicación con ámbito, reservar `CssStyle` para valores dinámicos puntuales, usar `AppearanceKey` y estados personalizados para variantes semánticas, y cambiar de tema al arrancar o para una sola sesión en tiempo de ejecución. Qué significa esto para un desarrollador de Wisej.NET y cómo leer este módulo.



Enfoque de producción

Añadir CSS específico de forma segura con `CssClass` y una hoja de estilos de la aplicación con ámbito, reservar `CssStyle` para valores dinámicos puntuales, usar `AppearanceKey` y estados personalizados para variantes semánticas, y cambiar de tema al arrancar o para una sola sesión en tiempo de ejecución.

1. El problema en producción

El tema `AdaptiveOps` del Módulo 2 ya es responsable de la identidad de todos los controles estándar de `Adaptive Operations Console`, y las peticiones que llegan a continuación son todas pequeñas: una tarjeta de métricas necesita una superficie redondeada y elevada, un badge de SLA debe ser compacto, una barra de progreso necesita un ancho calculado a partir de datos en vivo, el comando `Save` debe parecer la acción principal, y el turno de noche quiere una variante oscura. Cada petición tiene un atajo tentador — aquí una cadena `CssStyle`, allí un `BackColor`, el tema compartido editado directamente— y cada atajo traslada en silencio una decisión visual a una capa que no es su responsable. Seis meses después nadie sabe decir qué renderiza el tema, qué sobrescribe el CSS y qué parchea el código. Este módulo coloca cada una de esas cinco peticiones en el mecanismo responsable.

2. Conceptos clave

Concepto	Por qué importa
CssClass	Uno o varios nombres de clase añadidos al elemento principal del widget en el navegador: el punto de enganche reutilizable al que puede apuntar una hoja de estilos de la aplicación con ámbito sin tocar el tema.
CssStyle	Una cadena CSS que se escribe directamente como atributo style del elemento principal del widget; adecuada para un único valor calculado, inadecuada como herramienta de estilo general porque los estilos en línea son difíciles de inspeccionar, de tematizar y de entregar a un diseñador.
hoja de estilos de la aplicación con ámbito	Un archivo CSS cargado junto al tema cuyos selectores parten de nombres de clase propios, como <code>.metric-card</code> , y nunca entran en el DOM interno de un widget estándar de Wisej.
extender heredado StyleSheet	Un componente antiguo que añade clases CSS a los controles compatibles y aloja reglas en su propiedad <code>StyleSheetSource</code> ; una ayuda para la migración, innecesaria en código nuevo porque todos los controles tienen ya <code>CssClass</code> y <code>CssStyle</code> .
AppearanceKey	El nombre de una appearance del tema activo con la que se renderiza un control, de modo que el C# expresa la intención (<code>action-button</code>) y el tema es responsable de los colores, fuentes y métricas de cada estado.
estados personalizados	Nombres añadidos a la colección <code>States</code> de un control a los que el tema puede dar estilo como a <code>hovered</code> o <code>focused</code> , para condiciones transitorias de la interfaz como <code>stale</code> , <code>warning</code> o <code>dirty</code> que un diseñador debe poder ver y rediseñar.
tema de arranque frente a <code>Application.Theme</code>	El tema de toda la aplicación se elige antes de que se ejecute en <code>Web.config</code> o <code>Default.json</code> ; <code>Application.Theme</code> permite que una sesión trabaje con un objeto de tema en tiempo de ejecución sin cambiar lo que ven las demás sesiones.

3. Desarrollo de la lección

Cuatro herramientas, una regla para cada una

Todos los controles de Wisej.NET exponen dos puntos de enganche del lado del cliente. `CssClass` acepta uno o varios nombres de clase que se añaden al elemento principal del widget; `CssStyle` acepta una cadena CSS que se escribe directamente como atributo `style` de ese elemento. Están uno junto al otro en la cuadrícula de propiedades y se comportan de forma muy distinta en una base de código. Una clase es un nombre que puedes buscar, reutilizar en veinte controles y rediseñar en un solo archivo; un estilo en línea es una sobrescritura anónima que gana a la mayor parte de lo que genera el tema y permanece invisible hasta que alguien abre el inspector del navegador.

La regla es breve. Usa `CssClass` siempre que quieras un punto de enganche de estilo reutilizable respaldado por un archivo CSS de la aplicación. Usa `CssStyle` solo para estilos puntuales muy pequeños o valores calculados dinámicamente: un ancho que sigue un porcentaje, un color que llega con los datos. Abusar de `CssStyle` hace que la consola sea difícil de inspeccionar, de tematizar y de mantener para un

diseñador, y por eso el laboratorio permite exactamente un valor `CssStyle` calculado y te pide anotar por qué no podía ser una clase.

En proyectos reales siguen apareciendo dos mecanismos más antiguos. El extender `StyleSheet` añade clases CSS a los controles compatibles y aloja reglas en su propiedad `StyleSheetSource`; como todos los controles modernos ya tienen `CssClass` y `CssStyle`, es una ayuda para la migración, no algo que introducir en código nuevo. La sección `Stylesheet` del Theme Builder incluye reglas CSS sin procesar dentro del tema; existe para widgets de terceros, HTML personalizado y superficies de integración sin una `appearance` normal de Wisej, y en los widgets estándar las reglas generadas a partir de `appearances` y estados pueden sobrescribirla.

Clases con ámbito para la interfaz propia de la consola

El laboratorio añade `AdaptiveOps.css` a Adaptive Operations Console, una hoja de estilos que la aplicación carga junto al tema. Su función es limitada: dar estilo a las partes de la consola que son tuyas, no de Wisej. Una clase `.metric-card` da a los paneles de KPI su radio, padding y elevación, `.metric-card .metric-title` define el título pequeño en mayúsculas, y `.compact-badge` convierte una etiqueta en una píldora para los indicadores de SLA y prioridad. Todos los selectores parten de una clase propia, así que la hoja de estilos no puede rediseñar por accidente un botón o un encabezado de cuadrícula en otro lugar.

Lo que queda fuera de ese archivo importa tanto como lo que entra. El estilo de los controles estándar — los colores de hover y pressed de los botones, las fuentes de los encabezados de cuadrícula, el borde del editor no válido — pertenece a las `appearances` del tema del Módulo 2. Parchearlos desde CSS supone selectores frágiles que dependen del DOM interno de un widget y se rompen sin avisar en la siguiente actualización del framework o del tema. El CSS de widgets de terceros sigue la misma disciplina: con ámbito limitado al contenedor del widget y documentado.

Antes de publicar una clase, revisa qué puede anularla. Un control con `BackColor`, `ForeColor` o `Font` asignados directamente en el diseñador sobrescribe tanto el tema como tu clase; un `CssStyle` olvidado lo sobrescribe todo. Cuando una clase «no funciona», la lista de comprobación es: ¿está la clase en el elemento?, ¿hay una propiedad asignada en el control?, ¿hay un estilo en línea?, ¿tiene la regla el ámbito del contenedor correcto?

Variantes semánticas: `AppearanceKey` y estados personalizados

Asignar `BackColor`, `ForeColor` y `Font` al botón Save hace que hoy parezca principal y mañana sea intocable. `AppearanceKey` es la alternativa: el control se renderiza con una `appearance` con nombre del tema activo, y el tema es responsable de los detalles visuales de default, hovered, pressed, disabled, focused e invalid. La `appearance` `action-button` del Módulo 2 hereda de `button` y sobrescribe solo tres estados; asignarla al comando principal expresa la intención de negocio en C# y deja los píxeles al tema. El mismo patrón se extiende a `destructive-button`, `quiet-panel`, `kpi-card` o `sidebar-button` siempre que el concepto de negocio sea reutilizable.

Los estados personalizados cubren las condiciones transitorias que una clave de `appearance` no puede cubrir. Todos los controles tienen una colección `States`; añadir un nombre como `stale`, `warning`, `dirty` o `compact` permite que el tema dé estilo a ese estado exactamente como da estilo a hovered o focused. El laboratorio añade un estado `stale` a las filas de la cuadrícula de tickets o al panel de estado

cuando un ticket no se ha actualizado a tiempo, y el tema decide qué aspecto tiene stale. Un estado es una condición de la interfaz que el tema puede representar visualmente, no un lugar donde guardar datos de la aplicación, y el código nunca debe volver a leer un estado para tomar una decisión.

```
// MainPage.cs – each layer owns its own decision
this.cardOpen.CssClass      = "metric-card";      // scoped app CSS
this.lblPriorityBadge.CssClass = "compact-badge";
this.btnSave.AppearanceKey  = "action-button";    // the theme owns the states

// the one computed value: clip-path, not width – the layout engine rewrites
// an inline width on every layout pass and would erase it on the next resize
this.barFill.CssStyle = $"clip-path:inset(0 {100 - slaPercent}% 0 0)";

if (ticket.IsStale)
    row.States.Add("stale");                      // themed custom state
else
    row.States.Remove("stale");
```

Temas de arranque, temas de sesión y el tema compartido

Un tema se aplica antes de que se ejecute la aplicación nombrándolo en Web.config o en Default.json. Cambiar el tema que muestra el diseñador de Visual Studio no cambia el tema en tiempo de ejecución; la consola se ejecuta con AdaptiveOps solo porque la configuración lo indica. Los archivos de tema con nombre son el lugar adecuado para la identidad de marca estable: versionados, revisados e idénticos para todos los usuarios.

Wisej.NET también expone `Application.Theme`, mediante el cual una sesión puede trabajar con un objeto de tema en tiempo de ejecución. Ahí está el cambio claro/oscuro del laboratorio: el interruptor de la barra de herramientas carga la variante oscura, la asigna a la sesión actual y recuerda la elección en `Application.Session`. El peligro sutil es que un objeto de tema se puede modificar directamente. Modificar el tema compartido por todas las sesiones porque un usuario pidió el modo oscuro cambia la consola para todo el turno de noche; el patrón seguro es un archivo de tema con nombre independiente, o una copia por sesión, nunca una edición del objeto compartido.

La regla de decisión: archivos de tema con nombre para la identidad de marca estable, cambios de tema en tiempo de ejecución para personalización, acentos por tenant, demos y vistas previas controladas. Demuestra el límite con dos sesiones del navegador una junto a otra, una oscura y otra clara, antes de dar por terminado el entregable.



Errores habituales

Cadenas `CssStyle` copiadas en cada botón hasta que nadie sabe qué renderiza el tema. Selectores que entran en el DOM interno de un widget estándar. `BackColor`, `ForeColor` o `Font` asignados en el diseñador que sobrescriben sin avisar el tema y la clase. Usar `States` como almacén de datos. Un interruptor de modo oscuro que edita el tema compartido para todas las sesiones.



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 correctamente CssClass, CssStyle, CSS personalizado y los patrones del extender heredado StyleSheet.
- ✓ Definir clases con ámbito para la interfaz específica de la aplicación manteniendo en los temas el estilo de los controles estándar.
- ✓ Usar AppearanceKey y estados personalizados para crear variantes semánticas de controles.
- ✓ Aplicar temas de arranque y entender la diferencia entre los cambios de tema por sesión y los cambios de tema globales.

Guía del laboratorio / examen

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



Laboratorio práctico

Añade a Adaptive Operations Console una hoja de estilos AdaptiveOps.css con ámbito, con clases para tarjetas de métricas y badges compactos, aplícalas mediante CssClass y usa CssStyle solo para un valor calculado, como el ancho de una barra de progreso. Asigna la AppearanceKey action-button al comando principal, añade un estado personalizado stale a las filas de la cuadrícula de tickets o a un panel de estado, e implementa en la barra de herramientas un cambio de tema claro/oscuro que modifique el tema solo para la sesión actual, sin alterar el tema compartido.

Punto de partida

Abre Adaptive Operations Console del Módulo 2: el shell acoplado con los paneles de barra de herramientas, navegación, área de trabajo, detalles y estado, el tema AdaptiveOps seleccionado en Web.config o Default.json, sus tokens con nombre y la appearance `action-button`. El área de trabajo ya contiene cuatro paneles de métricas, una barra de progreso de SLA y la cuadrícula de tickets, pero siguen teniendo appearances predeterminadas o valores improvisados de `BackColor`, `ForeColor` y `CssStyle`, y la barra de herramientas no tiene cambio de tema.

Enfoque sugerido

- ✓ Crea `AdaptiveOps.css`, cárgalo con la aplicación junto al tema y escribe solo selectores anclados en clases propias: `.metric-card`, `.metric-card .metric-title` y `.compact-badge`. Aplícalos mediante `CssClass` a los cuatro paneles de métricas y a las etiquetas de SLA y prioridad.
- ✓ Localiza el único valor que debe calcularse en tiempo de ejecución, el ancho de progreso del SLA, asígnalo mediante `CssStyle` en un solo método y escribe un comentario que explique por qué una clase no podía expresarlo. Elimina cualquier otro `CssStyle` y toda sobrescritura de `BackColor`, `ForeColor` o `Font` que quede en esos controles.
- ✓ Asigna `AppearanceKey = "action-button"` al comando principal (`btnSave`) y confirma que los aspectos hovered, pressed y disabled vienen del tema sin ningún código.
- ✓ En el Theme Builder añade un estado `stale` a la appearance de la fila de tickets o del panel de estado, con estilo basado en el token warning; después añade y quita `"stale"` mediante `States` cuando la última actualización de un ticket supere el umbral. Mantén el umbral en los datos y el aspecto en el tema.
- ✓ Añade a la barra de herramientas un interruptor de tema que cargue un segundo archivo de tema con nombre, AdaptiveOps-Dark, lo asigne a `Application.Theme` para la sesión actual y recuerde la elección en `Application.Session`. Verifica con dos sesiones del navegador que el tema compartido queda intacto.

Entregables requeridos

- ✓ AdaptiveOps.css con ámbito y las clases metric-card y compact-badge aplicadas mediante CssClass
- ✓ CssStyle usado exactamente para un valor calculado puntual, con el motivo documentado
- ✓ Comando principal con la AppearanceKey action-button
- ✓ Estado personalizado stale añadido mediante States y con estilo definido en el tema
- ✓ Cambio de tema claro/oscuro solo para la sesión que deja intacto el tema compartido

Preguntas de repaso

- ✓ Un diseñador pide un radio de esquina más pequeño en todas las tarjetas de métricas. ¿Qué archivo cambia, y cuántos lugares habrías tenido que editar si el radio se hubiera asignado con CssStyle ?
- ✓ ¿Por qué el botón Assign no necesita código de hover ni de pressed, y qué perderías exactamente al sustituir su AppearanceKey por valores de BackColor y ForeColor que coincidan con la paleta?
- ✓ Hay dos personas conectadas y una de ellas cambia al modo oscuro. Describe lo que ve cada una y señala la línea de tu código que lo garantiza.



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.