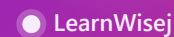


CSS, CssClass, CssStyle, états et changements de thème à l'exécution

Guide de la leçon · Guide du lab / de l'examen



Guide de la leçon

Ajouter du CSS ciblé en toute sécurité avec `CssClass` et une feuille de style applicative à portée limitée, réserver `CssStyle` aux valeurs dynamiques ponctuelles, utiliser `AppearanceKey` et des états personnalisés pour les variantes sémantiques, et changer de thème au démarrage ou pour une seule session à l'exécution. Ce que cela signifie pour un développeur Wisej.NET — et comment aborder ce module.



Enjeu en production

Ajouter du CSS ciblé en toute sécurité avec `CssClass` et une feuille de style applicative à portée limitée, réserver `CssStyle` aux valeurs dynamiques ponctuelles, utiliser `AppearanceKey` et des états personnalisés pour les variantes sémantiques, et changer de thème au démarrage ou pour une seule session à l'exécution.

1. Le problème en production

Le thème `AdaptiveOps` du Module 2 porte désormais l'identité de chaque contrôle standard d'`Adaptive Operations Console`, et les demandes qui arrivent ensuite sont toutes petites : une carte de métrique a besoin d'une surface arrondie et surélevée, un badge SLA doit être compact, une barre de progression a besoin d'une largeur calculée à partir de données en direct, la commande `Assign` doit ressembler à l'action principale, et l'équipe de nuit veut une variante sombre. Chaque demande a son raccourci tentant — une chaîne `CssStyle` ici, un `BackColor` là, le thème partagé modifié sur place — et chaque raccourci déplace discrètement une décision visuelle vers une couche qui ne la porte pas. Six mois plus tard, personne ne sait dire ce que rend le thème, ce que le CSS écrase et ce que le code rapièce. Ce module place chacune de ces cinq demandes dans le mécanisme qui la porte.

2. Concepts clés

Concept	Pourquoi c'est important
CssClass	Un ou plusieurs noms de classe ajoutés à l'élément principal du widget dans le navigateur : le point d'accroche réutilisable qu'une feuille de style applicative ciblée peut viser sans toucher au thème.
CssStyle	Une chaîne CSS écrite directement comme attribut style de l'élément principal du widget ; adaptée à une seule valeur calculée, inadaptée comme outil de style général, car les styles inline sont difficiles à inspecter, à intégrer au thème ou à confier à un designer.
feuille de style applicative ciblée	Un fichier CSS chargé à côté du thème dont les sélecteurs partent de noms de classe qui vous appartiennent, comme <code>.metric-card</code> , et n'atteignent jamais le DOM interne d'un widget Wisej standard.
étendre historique StyleSheet	Un ancien composant qui ajoute des classes CSS aux contrôles éligibles et héberge des règles dans sa propriété <code>StyleSheetSource</code> ; une aide à la migration, inutile dans le nouveau code puisque chaque contrôle dispose désormais de <code>CssClass</code> et de <code>CssStyle</code> .
AppearanceKey	Le nom d'une apparence du thème actif avec laquelle un contrôle s'affiche, pour que le C# exprime l'intention (<code>action-button</code>) et que le thème porte les couleurs, polices et métriques de chaque état.
états personnalisés	Des noms ajoutés à la collection <code>States</code> d'un contrôle, que le thème peut styler comme <code>hovered</code> ou <code>focused</code> , pour des conditions d'interface transitoires comme <code>stale</code> , <code>warning</code> ou <code>dirty</code> qu'un designer doit pouvoir voir et restyler.
thème de démarrage ou Application.Theme	Le thème global de l'application est choisi avant son exécution, dans <code>Web.config</code> ou <code>Default.json</code> ; <code>Application.Theme</code> permet à une session de travailler avec un objet de thème à l'exécution sans changer ce que voient les autres sessions.

3. Déroulé de la leçon

Quatre outils, une règle chacun

Chaque contrôle Wisej.NET expose deux points d'accroche côté client. `CssClass` accepte un ou plusieurs noms de classe ajoutés à l'élément principal du widget ; `CssStyle` accepte une chaîne CSS écrite directement comme attribut style de cet élément. Ils se côtoient dans la grille des propriétés et se comportent très différemment dans une base de code. Une classe est un nom que l'on peut rechercher, réutiliser sur vingt contrôles et restyler dans un seul fichier ; un style inline est une redéfinition anonyme qui l'emporte sur l'essentiel de ce que génère le thème et reste invisible jusqu'à ce que quelqu'un ouvre l'inspecteur du navigateur.

La règle est courte. Utilisez `CssClass` chaque fois que vous voulez un point d'accroche de style réutilisable adossé à un fichier CSS de l'application. N'utilisez `CssStyle` que pour de tout petits styles ponctuels ou des valeurs calculées dynamiquement : une largeur qui suit un pourcentage, une couleur qui arrive avec les données. Abuser de `CssStyle` rend la console difficile à inspecter, difficile à thémer et

difficile à maintenir pour un designer ; c'est pourquoi le lab autorise exactement une valeur `CssStyle` calculée et vous demande de noter pourquoi elle ne pouvait pas être une classe.

Deux mécanismes plus anciens apparaissent encore dans de vrais projets. L'extender StyleSheet ajoute des classes CSS aux contrôles éligibles et héberge des règles dans sa propriété `StyleSheetSource` ; comme chaque contrôle moderne dispose déjà de `CssClass` et de `CssStyle`, c'est une aide à la migration, pas un élément à introduire dans le nouveau code. La section Stylesheet du Theme Builder embarque des règles CSS brutes dans le thème ; elle existe pour les widgets tiers, le HTML personnalisé et les surfaces d'intégration sans appearance Wisej classique, et sur les widgets standard, les règles générées à partir des appearances et des états peuvent l'écraser.

Des classes ciblées pour l'interface propre à la console

Le lab ajoute `AdaptiveOps.css` à Adaptive Operations Console, une feuille de style que l'application charge à côté du thème. Son rôle est étroit : styler les parties de la console qui vous appartiennent, pas celles de Wisej. Une classe `.metric-card` donne aux panneaux d'indicateurs leur rayon, leur padding et leur élévation, `.metric-card .metric-title` définit le petit titre en majuscules, et `.compact-badge` transforme un label en pastille pour les marqueurs de SLA et de priorité. Chaque sélecteur part d'une classe qui vous appartient, si bien que la feuille de style ne peut pas restyler par accident un bouton ou un en-tête de grille ailleurs.

Ce qui reste hors de ce fichier compte autant que ce qui y entre. Le style des contrôles standard — couleurs de survol et d'appui des boutons, polices des en-têtes de grille, bordure de l'éditeur invalide — relève des appearances du thème du Module 2. Les rapiécer depuis le CSS implique des sélecteurs fragiles qui dépendent du DOM interne d'un widget et cassent sans bruit à la prochaine mise à jour du framework ou du thème. Le CSS des widgets tiers suit la même discipline : ciblé sur le conteneur du widget et documenté.

Avant de livrer une classe, examinez ce qui peut la mettre en échec. Un contrôle dont `BackColor`, `ForeColor` ou `Font` est défini directement dans le designer écrase à la fois le thème et votre classe ; un `CssStyle` oublié écrase tout. Quand une classe « ne marche pas », la checklist est la suivante : la classe est-elle sur l'élément, une propriété est-elle définie sur le contrôle, un style inline est-il défini, et la règle est-elle ciblée sur le bon conteneur ?

Variantes sémantiques : AppearanceKey et états personnalisés

Définir `BackColor`, `ForeColor` et `Font` sur le bouton Save lui donne l'air d'une action principale aujourd'hui et le rend intouchable demain. `AppearanceKey` est l'alternative : le contrôle s'affiche avec une appearance nommée du thème actif, et le thème porte les détails visuels de default, hovered, pressed, disabled, focused et invalid. L'appearance `action-button` du Module 2 hérite de `button` et ne redéfinit que trois états ; l'attribuer à la commande principale exprime l'intention métier en C# et laisse les pixels au thème. Le même modèle s'étend à `destructive-button`, `quiet-panel`, `kpi-card` ou `sidebar-button` dès que le concept métier est réutilisable.

Les états personnalisés couvrent les conditions transitoires qu'une clé d'appearance ne peut pas exprimer. Chaque contrôle possède une collection `States` ; y ajouter un nom comme `stale`, `warning`, `dirty` ou `compact` permet au thème de styler cet état exactement comme il stylise hovered ou focused. Le lab ajoute un état stale aux lignes de la grille des tickets ou au panneau d'état quand un ticket n'a pas été mis

à jour à temps, et c'est le thème qui décide de l'aspect de stale. Un état est une condition d'interface que le thème peut représenter visuellement, pas un endroit où conserver des données applicatives, et le code ne doit jamais relire un état pour prendre une décision.

```
// 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");
```

Thèmes de démarrage, thèmes de session et thème partagé

Un thème est appliqué avant l'exécution de l'application en le nommant dans Web.config ou dans Default.json. Changer le thème affiché dans le designer de Visual Studio ne change pas le thème d'exécution ; la console s'exécute avec AdaptiveOps uniquement parce que la configuration le demande. Les fichiers de thème nommés sont le bon endroit pour une identité visuelle stable : versionnés, relus et identiques pour chaque utilisateur.

Wisej.NET expose aussi `Application.Theme`, grâce auquel une session peut travailler avec un objet de thème à l'exécution. C'est là que se trouve le basculement clair/sombre du lab : le bouton de la barre d'outils charge la variante sombre, l'attribue à la session courante et mémorise le choix dans `Application.Session`. Le danger subtil est qu'un objet de thème peut être modifié sur place. Modifier le thème partagé par toutes les sessions parce qu'un utilisateur a demandé le mode sombre change la console pour toute l'équipe de nuit ; le modèle sûr est un fichier de thème nommé distinct, ou une copie par session, jamais une modification de l'objet partagé.

La règle de décision : des fichiers de thème nommés pour l'identité visuelle stable, des changements de thème à l'exécution pour la personnalisation, les accents par tenant, les démos et les aperçus contrôlés. Prouvez la frontière avec deux sessions de navigateur côte à côte, l'une sombre et l'autre claire, avant de considérer le livrable comme terminé.



Pièges courants

Des chaînes `CssStyle` copiées sur chaque bouton jusqu'à ce que plus personne ne sache ce que rend le thème. Des sélecteurs qui plongent dans le DOM interne d'un widget standard. Un `BackColor`, `ForeColor` ou `Font` défini dans le designer qui écrase sans bruit le thème et la classe. L'usage de `States` comme stockage de données. Un bouton de mode sombre qui modifie le thème partagé pour toutes les sessions.



Réflexe du système visuel

Utilisez le mécanisme Wisej.NET qui porte le problème : le thème pour l'identité des contrôles standard, AppearanceKey pour les variantes sémantiques, CssClass pour le CSS ciblé propre à l'application, CssStyle uniquement pour les valeurs dynamiques ponctuelles, les conteneurs de mise en page plutôt que du code de redimensionnement, les propriétés responsives pour les valeurs de profil définies dans le designer, et ResponsiveProfileChanged pour le comportement qui n'est pas simplement visuel.



Objectifs d'apprentissage

À la fin de ce module, vous serez capable de :

- ✓ Utiliser correctement CssClass, CssStyle, le CSS personnalisé et les modèles de l'étendre historique StyleSheet.
- ✓ Définir des classes ciblées pour l'interface propre à l'application tout en gardant le style des contrôles standard dans les thèmes.
- ✓ Utiliser AppearanceKey et des états personnalisés pour créer des variantes sémantiques de contrôles.
- ✓ Appliquer des thèmes de démarrage et comprendre la différence entre les changements de thème propres à une session et les changements de thème globaux.

Guide du lab / de l'examen

Ce que vous allez construire dans le lab pratique, l'approche suggérée et les livrables attendus.



Lab pratique

Ajoutez à Adaptive Operations Console une feuille de style `AdaptiveOps.css` à portée limitée, avec des classes pour les cartes de métriques et les badges compacts, appliquez-les via `CssClass`, et n'utilisez `CssStyle` que pour une seule valeur calculée, par exemple une largeur de progression. Attribuez l'`AppearanceKey` `action-button` à la commande principale, ajoutez un état personnalisé `stale` aux lignes de la grille des tickets ou à un panneau d'état, et implémentez dans la barre d'outils un basculement de thème clair/sombre qui change le thème pour la session courante uniquement, sans modifier le thème partagé.

Point de départ

Ouvrez Adaptive Operations Console telle que l'a laissée le Module 2 : le shell ancré avec les panneaux barre d'outils, navigation, espace de travail, détails et état, le thème `AdaptiveOps` sélectionné dans `Web.config` ou `Default.json`, ses tokens nommés et l'apparence `action-button`. L'espace de travail contient déjà quatre panneaux de métriques, une barre de progression SLA et la grille des tickets, mais ils sont encore stylés avec les apparences par défaut ou avec des valeurs `BackColor`, `ForeColor` et `CssStyle` improvisées, et la barre d'outils n'a pas de basculement de thème.

Approche suggérée

- ✓ Créez `AdaptiveOps.css`, chargez-le avec l'application à côté du thème, et n'écrivez que des sélecteurs enracinés dans des classes qui vous appartiennent : `.metric-card`, `.metric-card .metric-title` et `.compact-badge`. Appliquez-les via `CssClass` aux quatre panneaux de métriques et aux labels de SLA et de priorité.
- ✓ Identifiez l'unique valeur qui doit être calculée à l'exécution, la largeur de progression SLA, définissez-la via `CssStyle` dans une seule méthode, et écrivez un commentaire expliquant pourquoi une classe ne pouvait pas l'exprimer. Supprimez tous les autres `CssStyle` et toutes les redéfinitions `BackColor`, `ForeColor` ou `Font` restées sur ces contrôles.
- ✓ Attribuez `AppearanceKey = "action-button"` à la commande principale (`btnSave`) et vérifiez que les aspects `hovered`, `pressed` et `disabled` proviennent du thème, sans la moindre ligne de code.
- ✓ Dans le Theme Builder, ajoutez un état `stale` à l'apparence de la ligne de ticket ou du panneau d'état, stylé à partir du token `warning`, puis ajoutez et retirez `"stale"` via `States` quand la dernière mise à jour d'un ticket dépasse le seuil. Gardez le seuil dans les données et l'aspect dans le thème.
- ✓ Ajoutez à la barre d'outils un bouton de thème qui charge un second fichier de thème nommé, `AdaptiveOps-Dark`, l'attribue à `Application.Theme` pour la session courante et mémorise le choix dans `Application.Session`. Vérifiez avec deux sessions de navigateur que le thème partagé reste intact.

Livrables attendus

- ✓ AdaptiveOps.css à portée limitée avec les classes metric-card et compact-badge appliquées via CssClass
- ✓ CssStyle utilisé pour exactement une valeur ponctuelle calculée, avec la raison documentée
- ✓ Commande principale utilisant l'AppearanceKey action-button
- ✓ État personnalisé stale ajouté via States et stylé dans le thème
- ✓ Basculement de thème clair/sombre limité à la session qui laisse le thème partagé intact

Questions de révision

- ✓ Un designer demande un rayon d'angle plus petit sur toutes les cartes de métriques. Quel fichier change, et combien d'endroits auriez-vous dû modifier si le rayon avait été défini avec CssStyle ?
- ✓ Pourquoi le bouton Assign n'a-t-il besoin d'aucun code de survol ou d'appui, et que perdriez-vous exactement en remplaçant son AppearanceKey par des valeurs BackColor et ForeColor correspondant à la palette ?
- ✓ Deux personnes sont connectées et l'une d'elles passe en mode sombre. Décrivez ce que voit chacune, et désignez la ligne de votre code qui le garantit.



Réflexe du système visuel

Utilisez le mécanisme Wisej.NET qui porte le problème : le thème pour l'identité des contrôles standard, AppearanceKey pour les variantes sémantiques, CssClass pour le CSS ciblé propre à l'application, CssStyle uniquement pour les valeurs dynamiques ponctuelles, les conteneurs de mise en page plutôt que du code de redimensionnement, les propriétés responsives pour les valeurs de profil définies dans le designer, et ResponsiveProfileChanged pour le comportement qui n'est pas simplement visuel.