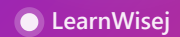


Polices d'icônes, contenu HTML et stratégie d'icônes en production

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



Guide de la leçon

Le dernier mécanisme et la politique qui les relie tous : une feuille de style de police d'icônes chargée depuis Default.html, AllowHtml sur les contrôles qui doivent interpréter le markup, des glyphes d'icônes dans les cellules d'une DataGridView avec des clics d'élément fondés sur les rôles, et une checklist de production couvrant l'accessibilité, les licences, la mise en cache, les dépendances externes et la cohérence visuelle. Ce que cela signifie pour un développeur qui connaît les bases de C# et a déjà travaillé avec les contrôles et le designer de Wisej.NET — et comment aborder ce module.



Objectif en production

Le dernier mécanisme et la politique qui les relie tous : une feuille de style de police d'icônes chargée depuis Default.html, AllowHtml sur les contrôles qui doivent interpréter le markup, des glyphes d'icônes dans les cellules d'une DataGridView avec des clics d'élément fondés sur les rôles, et une checklist de production couvrant l'accessibilité, les licences, la mise en cache, les dépendances externes et la cohérence visuelle.

1. Le problème en production

Après six modules, IconDesk sait dessiner une icône de cinq façons différentes, et toutes fonctionnent. Ce qui manque encore au projet, c'est une règle de décision : quand un nouvel écran a besoin d'un symbole Save, personne ne devrait avoir à deviner s'il vient du thème, d'un pack officiel Wisej-4 ou d'IconDesk.Icons. Ce module ajoute la dernière option, les polices d'icônes rendues en HTML, puis écrit la politique qui dit vers laquelle se tourner et ce qu'il faut vérifier avant qu'un écran ne parte.

2. Concepts clés

Concept	Pourquoi c'est important
police d'icônes	Une police web dont les glyphes sont des images au lieu de lettres, choisies par des noms de classes CSS et dessinées par le navigateur dans du contenu HTML plutôt que via une propriété d'image.
AllowHtml	L'interrupteur par contrôle qui indique à Wisej.NET d'interpréter le texte d'un contrôle comme du markup ; sans lui, les balises apparaissent en caractères littéraux, car le texte est du texte simple par défaut.
en-tête de Default.html	La page hôte de l'application, où un seul lien de feuille de style rend le CSS et les fichiers de police de la police d'icônes disponibles pour chaque écran ouvert par la session.
clic d'élément fondé sur le rôle	Un clic dans du contenu HTML peut indiquer quel élément en ligne a été atteint, si bien qu'une seule cellule de grille peut offrir une cible Edit et une cible Delete sans deux contrôles distincts.
CDN contre auto-hébergement	Pointer le lien vers un CDN public est une décision de production touchant à la disponibilité, à la confidentialité, à la Content Security Policy et aux licences ; les applications internes hébergent souvent les fichiers elles-mêmes.
politique d'icônes par couches	Une règle écrite qui donne un foyer unique à chaque type de symbole : images de thème pour les sémantiques, un pack principal pour les commandes, le pack personnalisé pour les visuels du produit, fichiers raster pour les photos.
pas de QA sur les icônes	Un parcours des écrans finis dans un thème clair et un thème sombre, à plus d'une échelle d'affichage, pour vérifier que chaque icône est toujours visible et dispose toujours d'un équivalent textuel.

3. Le fil de la leçon

Charger la feuille de style et activer le markup

Une police d'icônes ne fait pas du tout partie du système d'images de Wisej.NET. Rien n'est affecté à `Image`, rien n'est écrit dans `ImageSource`, et le serveur ne produit jamais de PNG. Le navigateur télécharge au contraire une feuille de style et un fichier de police, et un nom de classe sur un élément en ligne indique à la police quel glyphe dessiner. Cela en fait une affaire de page plutôt qu'une affaire de contrôle : le lien vers la feuille de style a sa place dans l'en-tête du `Default.html` de l'application, où il est récupéré une fois pour la session et s'applique à toutes les fenêtres et boîtes de dialogue qu'ouvre l'utilisateur.

L'endroit vers lequel ce lien pointe est une décision de production. Un CDN public est pratique, mais l'interface dépend alors de l'accessibilité d'un tiers, elle révèle les requêtes de vos utilisateurs à cet hôte, elle doit passer votre Content Security Policy, et elle est régie par la licence de la bibliothèque d'icônes. Pour une application interne, servir vous-même le CSS et les fichiers de police est souvent le meilleur choix : la disponibilité et la confidentialité ne dépendent alors que de votre propre déploiement.

La feuille de style chargée, rien n'apparaît encore. Les contrôles Wisej.NET traitent `Text` comme du texte simple : le markup arrive donc échappé et l'utilisateur voit des chevrons. C'est le passage d' `AllowHtml` à `true` qui change cela, et c'est délibérément par contrôle : Label, Button, cellules de DataGridView, CheckBox, RadioButton et LinkLabel sont les exemples compatibles HTML. Activer l'interrupteur déplace aussi une frontière de confiance : à partir de cet instant, le texte du contrôle est du markup ; gardez-le donc sous la responsabilité de l'application et placez les mots de l'utilisateur dans un contrôle en texte simple distinct.

Des glyphes dans une barre d'outils, une carte et une cellule de grille

Dans IconDesk, la première surface en police d'icônes est une petite barre d'outils ou une carte de synthèse. Chaque bouton ou libellé reçoit `AllowHtml = true` et une valeur de texte qui place un élément en ligne devant un mot, ce qui est là où les polices d'icônes paraissent naturelles : le glyphe s'écoule avec le libellé au lieu de siéger dans un emplacement d'image distinct, et il hérite de la taille et de la couleur de la police environnante. La même idée passe à l'échelle dans une `DataGridView` dont `AllowHtml` est activé, où une colonne d'actions compacte contient deux glyphes dans une cellule au lieu de deux contrôles hébergés.

Une cellule unique avec deux symboles ne fonctionne que si l'application peut dire lequel a été cliqué. C'est à cela que sert un attribut `role` sur l'élément en ligne : le clic indique l'élément atteint, et le gestionnaire s'y ramifie. Inspectez les données d'événement souris que votre version du framework expose réellement avant de vous y fier, et n'inventez pas un contrat que l'événement n'offre pas.

```
labelActions.AllowHtml = true;
labelActions.Text =
    "<i class='fa-solid fa-pen' role='edit'></i> " +
    "<i class='fa-solid fa-trash' role='delete'></i>";

labelActions.MouseClick += labelActions_MouseClick;
```

La compacité a un coût. Une cellule pleine de markup est plus difficile à atteindre au clavier qu'un vrai bouton, les lecteurs d'écran reçoivent un glyphe sans nom, et une icône qui ne se charge pas laisse une colonne vide. Donnez à chaque action une info-bulle ou un mot visible, et tournez-vous vers une colonne d'images ordinaire ou un contrôle hébergé dès que l'interaction dépasse le raccourci à deux icônes.

Les polices d'icônes à côté du SVG et des packs d'icônes

Aucun mécanisme ne l'emporte totalement. Le système propre à Wisej.NET repose sur les ressources et le SVG, et c'est là qu'il est le plus fort : une chaîne de source peut nommer une image de thème, un chemin `resource.wx` ou une ressource de pack ; les SVG monochromes acceptent le suffixe `?color=` avec un nom de couleur du thème ; le sélecteur d'images du designer parcourt chaque pack installé ; la mise en cache est bâtie autour de ces ressources ; et tout contrôle doté d'une propriété d'image peut les utiliser directement.

Les polices d'icônes gagnent leur place ailleurs. Elles sont familières aux développeurs web, elles s'insèrent dans un design system HTML et CSS existant, leurs catalogues sont énormes, et elles se placent en ligne avec le texte d'une manière qu'une propriété d'image ne permet pas. Si le langage de conception de

l'entreprise est déjà exprimé en classes de police, s'y accorder coûte un lien de feuille de style au lieu de redessiner chaque ressource.

La réponse honnête est donc qu'ils coexistent, et que le choix appartient à la surface plutôt qu'au projet. Les commandes portées par des contrôles dotés de propriétés d'image relèvent du pipeline de ressources ; les symboles en ligne dans du texte ou dans une cellule de grille dense sont un cas raisonnable de police d'icônes. Ce qui n'est pas acceptable, c'est la dérive, où le même concept Delete est un SVG de pack sur un écran et un glyphe de police sur le suivant.

La checklist de production et la politique d'IconDesk

Le véritable livrable du projet de synthèse est une page de politique. Une règle par couches fonctionne bien : images de thème pour les sémantiques courantes, un pack officiel Wisej-4 principal pour les commandes larges, le pack personnalisé IconDesk.Icons pour les visuels propriétaires, des fichiers raster compressés pour les photographies, et les polices d'icônes seulement là où l'intégration HTML en vaut la peine. Parce que la politique nomme un premier choix pour chaque situation, un nouvel écran est cohérent par défaut plutôt que par revue.

Autour de cela se place la checklist que le projet déroule avant qu'un écran ne parte :

- Chaque icône porteuse de sens a un mot visible, une info-bulle ou un autre équivalent textuel, et l'état n'est jamais signalé par la seule couleur.
- Toute URL externe ou référence à un CDN est explicitement approuvée, et les termes de licence de la bibliothèque sont consignés.
- Le markup de police d'icônes appartient à l'application ou est encodé ; des valeurs non fiables n'atteignent jamais un contrôle `AllowHtml`.
- Les URL de ressources sont stables pour que le cache du navigateur puisse travailler, et aucune chaîne de requête aléatoire ne le ruine.
- L'écran a été ouvert dans un thème clair et un thème sombre et à une échelle d'affichage plus élevée, et les constats sont consignés.

Cette passe de QA est le moment où les cinq mécanismes se retrouvent enfin côte à côte. Sur la page de synthèse d'IconDesk, une image de thème, une icône de pack officiel, une icône de pack personnalisé, une ressource embarquée atteinte via `resource.wx` et un SVG recoloré apparaissent tous en même temps, et changer de thème montre lesquels l'ont suivi et lesquels sont restés exactement tels qu'ils ont été dessinés. Expliquer, pour chaque image visible, où elle est stockée et quelle API l'a résolue, c'est la compétence vers laquelle ce cours a travaillé.



Nommez d'abord le mécanisme

Avant d'affecter la moindre image, dites quel mécanisme vous utilisez : un objet `System.Drawing.Image` que le serveur rend en PNG, une chaîne de source d'image que le navigateur résout, une image de thème nommée, une ressource embarquée `resource.wx`, ou une police d'icônes dessinée en HTML. N'affectez qu'une seule propriété d'image par contrôle pour que les autres ne puissent pas la réinitialiser en silence, gardez une chaîne de source parlante plutôt qu'un chemin magique collé, préférez le SVG pour les icônes d'interface afin qu'elles restent nettes à toute taille et à toute échelle, et donnez à chaque icône porteuse de sens un équivalent textuel.



Objectifs d'apprentissage

À la fin de ce module, vous saurez :

- ✓ Charger une feuille de style de police d'icônes et afficher des icônes dans les contrôles Wisej.NET avec AllowHtml.
- ✓ Utiliser les polices d'icônes dans le contenu de Label, Button et DataGridView et comprendre les clics d'élément fondés sur les rôles.
- ✓ Comparer les polices d'icônes aux ressources SVG et de packs d'icônes de Wisej.NET.
- ✓ Appliquer une checklist de production couvrant l'accessibilité, la sécurité, la mise en cache, les dépendances externes et la cohérence.

Guide du lab / de l'examen

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



Lab pratique

Terminez le projet de synthèse IconDesk. Référenciez une feuille de style de police d'icônes depuis Default.html, activez AllowHtml sur une petite barre d'outils ou une carte et affichez ses icônes sous forme de glyphes de police, puis construisez une colonne DataGridView dont les cellules contiennent des actions en police d'icônes et lisez l'élément sur lequel l'utilisateur a cliqué pour qu'Edit et Delete fassent des choses différentes. Construisez une page de synthèse qui démontre d'un seul coup tous les mécanismes vus dans le cours : une image de thème, une icône d'un pack officiel, une icône d'un pack personnalisé, un logo embarqué et un SVG recoloré. Rédigez ensuite la politique d'icônes de production du projet et effectuez une passe de QA dans un thème clair et un thème sombre.

Point de départ

Ouvrez la solution IconDesk telle que le Module 6 l'a laissée : le panneau de commandes, les pages ImageLab et IconGallery, deux packs d'icônes officiels Wisej-4, le dossier Resources embarqué servi via `resource.wx`, et la bibliothèque de classes IconDesk.Icons avec son catalogue `AppIcons`. Rien dans le projet n'utilise encore de police d'icônes, et le `Default.html` du projet a toujours un en-tête vide à part le titre.

Approche suggérée

- ✓ Décidez d'abord si la police d'icônes est référencée depuis un CDN ou hébergée à côté de l'application, et consignez les raisons de disponibilité, de confidentialité, de Content Security Policy et de licence de ce choix avant d'ajouter le lien à `Default.html`.
- ✓ Ajoutez une petite barre d'outils ou une carte, activez `AllowHtml` sur chaque Label ou Button, et vérifiez qu'un contrôle sans l'interrupteur affiche le markup en texte littéral, afin d'avoir vu les deux comportements.
- ✓ Activez `AllowHtml` sur la grille, construisez la colonne d'actions avec deux éléments en ligne portant des rôles distincts, puis inspectez les données d'événement souris qu'expose votre version de Wisej.NET et ramifiez Edit et Delete sur l'élément atteint.
- ✓ Construisez la page de synthèse pour que chacun des cinq mécanismes apparaisse une fois, étiqueté avec l'endroit où l'image est stockée et l'API qui la résout, et gardez un seul contrôle par propriété d'image.
- ✓ Effectuez la passe de QA : basculez entre un thème clair et un thème sombre, essayez une échelle d'affichage plus élevée, vérifiez les info-bulles et l'accès au clavier sur les actions sans texte, et notez tout ce qui a changé ou disparu.

Livrables attendus

- ✓ Feuille de style de police d'icônes référencée depuis Default.html et une barre d'outils ou une carte en AllowHtml affichant des glyphes de police

- ✓ Colonne DataGridView dont les cellules affichent des actions en police d'icônes et réagissent à l'élément cliqué
- ✓ Page de synthèse IconDesk démontrant une image de thème, une icône d'un pack officiel, une icône d'un pack personnalisé, une ressource embarquée et un SVG recoloré
- ✓ Politique d'icônes de production écrite couvrant l'accessibilité, les licences, la mise en cache et les dépendances externes
- ✓ Passe de QA effectuée dans un thème clair et un thème sombre avec les constats consignés

Questions de révision

- ✓ Le lien de la feuille de style est dans `Default.html` et les noms de classes sont corrects, mais un Label affiche les balises brutes alors que le Button voisin affiche le glyphe. Qu'est-ce qui diffère entre les deux contrôles, et pourquoi le réglage est-il par contrôle plutôt que par application ?
- ✓ Votre colonne d'actions fonctionne, et un collègue demande à placer le texte de la note du client dans la même cellule pour pouvoir le styler. Que lui répondez-vous, et que construiriez-vous à la place ?
- ✓ Sur la page de synthèse, lesquelles des cinq images changent quand l'application change de thème et lesquelles restent exactement telles qu'elles ont été dessinées ? Que cela vous dit-il sur l'endroit où chacune est stockée ?



Nommez d'abord le mécanisme

Avant d'affecter la moindre image, dites quel mécanisme vous utilisez : un objet `System.Drawing.Image` que le serveur rend en PNG, une chaîne de source d'image que le navigateur résout, une image de thème nommée, une ressource embarquée `resource.wx`, ou une police d'icônes dessinée en HTML. N'affectez qu'une seule propriété d'image par contrôle pour que les autres ne puissent pas la réinitialiser en silence, gardez une chaîne de source parlante plutôt qu'un chemin magique collé, préférez le SVG pour les icônes d'interface afin qu'elles restent nettes à toute taille et à toute échelle, et donnez à chaque icône porteuse de sens un équivalent textuel.