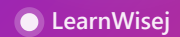


Images de thème, URL, couleur des SVG et sélecteur d'images du designer

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



Guide de la leçon

Des sources d'image qui ne sont pas des fichiers livrés à côté de la page : des images de thème nommées chargées depuis le cache de thème du client, des URL relatives et absolues, des données base64 embarquées dans le thème, l'Image Selector de Visual Studio qui écrit la chaîne de source à votre place, et le suffixe `?color=` qui recolore un SVG monochrome adapté avec un nom de couleur du thème. 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

Des sources d'image qui ne sont pas des fichiers livrés à côté de la page : des images de thème nommées chargées depuis le cache de thème du client, des URL relatives et absolues, des données base64 embarquées dans le thème, l'Image Selector de Visual Studio qui écrit la chaîne de source à votre place, et le suffixe `?color=` qui recolore un SVG monochrome adapté avec un nom de couleur du thème.

1. Le problème en production

IconDesk possède désormais un panneau de commandes et un atelier d'images, et chaque image qu'ils contiennent est un fichier livré assorti d'un chemin saisi par quelqu'un. Cela tient jusqu'à ce que l'application change de thème, soit déployée sous un autre répertoire virtuel, ou qu'on lui demande d'afficher le même symbole dans une couleur d'avertissement sur un écran et dans une couleur neutre sur un autre. Ce module fait sortir l'origine d'une image du dossier du projet pour la placer dans le thème, dans une URL ou dans le designer, afin que la chaîne portée par un contrôle signifie encore quelque chose après un changement de visuel, de thème ou de serveur.

2. Concepts clés

Concept	Pourquoi c'est important
image de thème nommée	Un nom sémantique tel qu' <code>icon-print</code> que le thème actif associe à un visuel, si bien qu'un autre thème peut fournir une autre image sans le moindre changement en C#.
cache de thème du client	Les images de thème voyagent avec le thème et sont déjà dans le navigateur quand un contrôle en demande une, ce qui fait d'un thème le bon foyer pour un petit jeu de symboles de commande et le mauvais foyer pour des milliers.
source en URL relative	Une chaîne de source d'image résolue par rapport à l'application : la forme la plus simple pour un logo déployé avec l'application.
source en URL absolue	Une adresse externe complète qui transforme l'image en dépendance envers un autre système, avec des conséquences de disponibilité, de confidentialité, de CORS et de Content Security Policy.
image base64 embarquée	Les octets de l'image écrits dans la définition de thème elle-même, si bien que le visuel n'a besoin d'aucune seconde requête, au prix d'un thème plus lourd.
Image Selector	Le sélecteur au design de Visual Studio qui parcourt les images du projet, les images de thème et les ressources des packs d'icônes référencés, puis sérialise la chaîne de source choisie dans <code>Designer.cs</code> .
suffixe <code>?color=</code>	Un suffixe sur une source d'image qui demande à Wisej.NET de recolorer un SVG monochrome compatible avec une couleur CSS ou un nom de couleur du thème.

3. Le fil de la leçon

Quatre formes qu'une image peut prendre dans un thème

Un thème Wisej.NET n'est pas seulement fait de couleurs et de bordures ; il porte aussi des définitions d'images que le framework et votre application peuvent tous deux utiliser. La documentation du Theme Builder décrit quatre formes utiles pour ces définitions. Une **URL relative** est résolue par rapport à l'application. Une **URL absolue** désigne une adresse externe complète. Des données **base64 embarquées** placent les octets de l'image dans la définition de thème, si bien que rien d'autre n'a besoin d'être récupéré. Une **image nommée** pointe vers une définition d'image, ce qui ajoute une couche d'indirection : plusieurs apparences peuvent référencer le même nom, et changer la définition une seule fois change tous les usages.

L'indirection est la partie intéressante pour IconDesk. Quand la galerie demande un nom plutôt qu'un fichier, la décision sur le visuel que désigne ce nom appartient au thème, et le thème peut être remplacé au déploiement ou par client. Quand la page demande une URL, la décision appartient à celui qui a saisi l'URL, et changer de thème ne change rien. Le lab place les deux comportements sur un même écran pour que la différence ne soit pas affaire de croyance.

Les URL absolues méritent une décision délibérée plutôt qu'une habitude. Une image récupérée sur un autre hôte est une dépendance à l'exécution : elle peut être indisponible, elle peut exiger une

authentification dont le navigateur ne dispose pas, elle peut être bloquée par une Content Security Policy, et sa politique de cache n'est pas la vôtre. Une icône qui doit toujours être présente est mieux déployée avec l'application ou empaquetée en ressource, ce que traite le Module 5.

Un nom d'image de thème est une clé, pas un chemin

Affecter une image de thème est la ligne la plus courte du cours : définissez une propriété `ImageSource` sur le nom, et Wisej.NET la charge depuis le cache de thème côté client. Rien sur le serveur n'ouvre de fichier et aucun `System.Drawing.Image` n'est créé : c'est donc le second pipeline du Module 1, pas le premier. La chaîne est une clé de thème. Si un autre thème associe cette clé à un visuel différent, l'application change d'apparence sans une seule modification en C#, et c'est pourquoi des noms sémantiques tels qu' `icon-print` , `icon-save` ou `icon-warning` valent mieux qu'un chemin collé.

Ce comportement de préchargement est aussi une règle de dimensionnement. Les images de thème arrivent avec le thème : un thème convient donc à un petit jeu stable de symboles de commande de niveau framework. Les ressources de packs d'icônes se comportent à l'inverse : chaque icône est demandée individuellement puis mise en cache par le navigateur, ce qui passe à l'échelle sur de très grands catalogues mais fait coûter une requête à la première utilisation de chaque icône. Savoir lequel des deux vous utilisez vous dit à quoi le panneau réseau devrait ressembler, et c'est la preuve que le Module 4 vous demande de recueillir.

```
// IconGallery: three source strings, three different resolutions.
buttonPrint.ImageSource = "icon-print";           // theme key
logo.ImageSource        = "Images/company-logo.svg"; // relative URL
avatar.ImageSource      = "https://cdn.example.com/users/42.jpg"; // absolute

// Runtime choices stay semantic, never assembled from user text.
buttonStatus.ImageSource = isDanger
    ? "icon-warning?color=error"
    : "icon-info?color=activeText";
```

Laissez l'Image Selector écrire la chaîne de source

Le designer Wisej.NET dans Visual Studio offre une expérience de sélection d'images pour les propriétés de source. Il parcourt les images du projet, les images définies par le thème courant et, lorsque des assemblys de packs d'icônes sont référencés, les packs qu'il découvre automatiquement. L'explorateur de packs d'icônes affiche une liste ou une grille avec un aperçu, et une icône monochrome peut être prévisualisée dans une couleur choisie avant que vous ne validiez. Quand vous en choisissez une, le designer sérialise une source concrète telle qu'une URL `resource.wx/...` dans `Designer.cs` .

C'est bien plus qu'une aide à la saisie. Lire la ligne générée enseigne la représentation à l'exécution : choisir une icône de pack ne copie **pas** le SVG dans le formulaire. Le formulaire stocke une chaîne de source qui renvoie à l'assembly du pack, et c'est pourquoi cinq formulaires utilisant une même icône ne livrent toujours qu'un seul exemplaire du visuel. Le lab vous demande d'ouvrir `Designer.cs` et de lire ces chaînes exactement pour cette raison.

- Images du projet : choisissez-les avec le sélecteur pour qu'aucun chemin ne soit saisi deux fois.
- Images de thème : utilisez la liste des thèmes et l'aperçu pour voir le visuel du thème courant.

- Ressources de packs d'icônes : cherchez et prévisualisez plutôt que de mémoriser des noms de fichiers.

Recolorer un SVG monochrome avec ?color=

Wisej.NET peut recolorer un SVG monochrome compatible en ajoutant `?color=` à la source d'image. La valeur peut être une couleur de style CSS telle que `red` ou `#3078d4`, ou le nom d'une couleur du thème telle qu'`error` ou `activeText`. Une couleur de thème est le choix le plus solide parce qu'elle continue d'avoir du sens quand le thème change : un gris foncé codé en dur peut s'évanouir sur un thème sombre, alors qu'une couleur de thème sémantique est conçue pour s'y adapter. Le même suffixe fonctionne sur un nom d'image de thème comme sur une source de ressource `resource.wx`, parce qu'il s'applique à la chaîne de source et non à un mécanisme de stockage particulier.

Le recolorage n'est pas universel, et l'échec est silencieux. La documentation avertit qu'il peut échouer lorsque le SVG code lui-même en dur des valeurs `fill` ou `stroke`, parce que ces valeurs l'emportent. Pour une icône destinée à être colorée par Wisej.NET, retirez les couleurs codées en dur qui entrent en conflit lorsque c'est approprié ; pour un logo multicolore, n'y touchez pas. La galerie d'IconDesk doit contenir un exemplaire de chaque, pour que la note de lab puisse dire quel visuel a accepté le suffixe et lequel l'a ignoré. Gardez des chaînes de source sémantiques à l'exécution, et ne construisez jamais un chemin de ressource à partir d'un texte utilisateur non fiable.



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 :

- ✓ Utiliser des URL relatives, des URL absolues, des données base64 embarquées et des images nommées dans un thème Wisej.NET.
- ✓ Affecter directement des noms d'images de thème à `ImageSource`.
- ✓ Utiliser le sélecteur d'images au design de Visual Studio pour choisir des ressources locales, de thème et de packs d'icônes.
- ✓ Recolorer les icônes SVG adaptées avec la syntaxe `?color=` de Wisej.NET et les noms de couleurs du thème.

Guide du lab / de l'examen

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



Lab pratique

Remplissez une page IconGallery dans IconDesk à l'aide de l'Image Selector du designer plutôt qu'en saisissant les chaînes de source à la main. Affectez des images à cinq contrôles : une image de thème nommée, un SVG du projet référencé par URL relative, une URL absolue et au moins deux ressources issues d'un pack d'icônes si un pack est installé. Recolez un SVG monochrome avec le suffixe `?color=` et un nom de couleur du thème, ouvrez le Designer.cs généré et lisez les chaînes de source d'image produites par le sélecteur, puis basculez l'application entre deux thèmes Wisej et notez quelles ressources ont changé et lesquelles sont restées strictement identiques.

Point de départ

Ouvrez la solution IconDesk là où le Module 2 l'a laissée : le panneau de commandes du Module 1 et la page ImageLab avec sa grille de modes de dimensionnement, son PictureBox en `LoadAsync` et son SVG net. Ajoutez une nouvelle page nommée `IconGallery` avec cinq contrôles vides et un sélecteur de thème, pour que chaque image qu'elle porte soit affectée pendant ce lab et que rien ne soit hérité d'un module antérieur.

Approche suggérée

- ✓ Placez d'abord les cinq contrôles de la galerie et donnez à chacun un libellé nommant le mécanisme que vous comptez utiliser, avant qu'aucune image ne soit affectée.
- ✓ Affectez chaque image via l'Image Selector du designer : la liste des thèmes pour l'image de thème nommée, l'arborescence du projet pour le SVG relatif, le champ d'adresse pour l'URL absolue, et l'explorateur de packs d'icônes pour les ressources de packs.
- ✓ Ajoutez le suffixe `?color=` avec un nom de couleur du thème à un SVG monochrome, et essayez-le aussi sur une ressource multicolore pour voir le suffixe être ignoré.
- ✓ Ouvrez `Designer.cs`, copiez les lignes de source d'image générées dans la note de lab et marquez chacune comme clé de thème, URL relative, URL absolue ou source de ressource.
- ✓ Basculez l'application entre deux thèmes Wisej, capturez la galerie dans les deux, et notez quels contrôles ont suivi le thème et lesquels ne l'ont pas suivi.

Livrables attendus

- ✓ Page IconGallery avec cinq contrôles dont les images ont été affectées via l'Image Selector du designer
- ✓ Parmi elles, une image de thème nommée, un SVG relatif du projet, une URL absolue et au moins deux ressources de pack d'icônes
- ✓ SVG monochrome recoloré via le suffixe `?color=` avec un nom de couleur du thème
- ✓ Extrait de Designer.cs montrant les chaînes de source d'image générées par le sélecteur

- ✓ Note de changement de thème consignait les ressources qui ont suivi le thème et celles qui sont restées fixes

Questions de révision

- ✓ La galerie s'affiche correctement sur votre machine et ne montre rien pour l'URL absolue sur celle d'un collègue. Lequel des cinq mécanismes pourrait produire cette différence, et que vérifieriez-vous en premier ?
- ✓ Deux de vos icônes refusent de changer de couleur quand le suffixe est ajouté. Qu'y a-t-il à l'intérieur de ces fichiers, et quand les modifier est-il la mauvaise réponse ?
- ✓ Un designer demande le même symbole dans la galerie, dans une barre d'outils et dans une cellule de grille. Entre la clé de thème, l'URL relative et la ressource de pack, laquelle coûtera le moins cher à changer plus tard, et pourquoi ?



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.