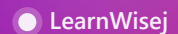


Packs d'icônes Wisej.NET et NuGet

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



Guide de la leçon

Ce qu'est réellement un pack d'icônes : un assembly .NET qui transporte des centaines de fichiers SVG en ressources embarquées, adressés individuellement via resource.wx et mis en cache par le navigateur une icône à la fois, installé depuis les packages NuGet officiels Wisej-4, parcouru dans l'explorateur d'images de Visual Studio et référencé depuis C# via le catalogue d'icônes du pack. 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

Ce qu'est réellement un pack d'icônes : un assembly .NET qui transporte des centaines de fichiers SVG en ressources embarquées, adressés individuellement via resource.wx et mis en cache par le navigateur une icône à la fois, installé depuis les packages NuGet officiels Wisej-4, parcouru dans l'explorateur d'images de Visual Studio et référencé depuis C# via le catalogue d'icônes du pack.

1. Le problème en production

Trois modules plus loin, IconDesk sait déjà afficher une image de cinq façons différentes, mais chaque icône qu'il possède a été dessinée ou téléchargée à la main. Une vraie application a besoin d'une icône Save, Delete, Search, User et Settings dès le premier jour, et de quarante autres d'ici la fin du trimestre, toutes dans le même langage visuel. Copier des fichiers SVG un par un produit un dossier de visuels sans lien, des épaisseurs de trait incohérentes et aucune traçabilité des licences.

2. Concepts clés

Concept	Pourquoi c'est important
pack d'icônes	Un assembly .NET ordinaire dont les fichiers d'icônes, le plus souvent des SVG, sont compilés en ressources embarquées ; le référencer ajoute un seul fichier à votre déploiement, quel que soit le nombre d'icônes qu'il contient.
ressource embarquée	Un fichier compilé dans l'assembly plutôt que copié à côté, si bien que les icônes ne peuvent être ni perdues, ni renommées, ni oubliées lorsque l'application est déployée sur un autre serveur.
resource.wx	Le gestionnaire Wisej.NET qui sert une ressource embarquée par HTTP ; une source d'image de pack d'icônes est une URL <code>resource.wx</code> qui nomme l'assembly du pack et le fichier qu'il contient.
catalogue d'icônes	La classe statique qu'un pack publie pour que les appelants obtiennent l'IntelliSense et un nom vérifié à la compilation au lieu d'un chemin saisi ; le namespace et le nom de classe exacts varient d'un pack à l'autre.
requête par icône	Les icônes de packs sont récupérées individuellement, à la première utilisation, puis servies depuis le cache du navigateur, et c'est ce qui rend abordable un catalogue de plusieurs milliers d'icônes.
préchargement des images de thème	Les images de thème voyagent avec le thème et sont déjà dans le cache de thème du client au démarrage de l'application : elles se comportent donc très différemment des icônes de packs au lancement.
famille d'icônes principale	Le pack unique auquel un produit s'engage pour ses commandes quotidiennes, choisi délibérément pour sa couverture, son style et sa licence plutôt qu'accumulé pack après pack.

3. Le fil de la leçon

Un pack d'icônes est un assembly, pas un dossier

La première chose à désapprendre est l'idée qu'un pack d'icônes serait une archive d'images. Un pack d'icônes Wisej.NET est un assembly .NET normal. Ses icônes, très majoritairement des SVG dans les packs officiels, y sont compilées en ressources embarquées : un pack contenant plusieurs milliers d'icônes arrive donc dans votre dossier `bin` sous la forme d'une seule DLL. Rien n'est copié à côté de `Default.html`, rien n'a besoin d'être publié avec le site, et rien ne peut être supprimé par accident pendant un déploiement.

Chaque icône reste néanmoins adressable individuellement. À l'exécution, le navigateur demande une icône à la fois via le gestionnaire `resource.wx`, qui lit cette unique ressource embarquée dans l'assembly référencé et la renvoie. La source d'image que vous affectez n'est donc qu'une chaîne : le nom du gestionnaire, l'assembly dans lequel vit la ressource, et le fichier qu'il contient. Comme c'est une réponse HTTP ordinaire, le navigateur la met en cache comme n'importe quel autre fichier : le deuxième contrôle qui utilise la même icône, et le chargement de page suivant, ne coûtent donc rien.

C'est cette livraison icône par icône qui rend abordable un pack de milliers d'icônes : référencer l'assembly n'envoie rien au navigateur, utiliser une icône envoie un fichier. Une image de thème nommée fonctionne à l'inverse, préchargée avec le thème avant le premier rendu d'écran.

Installer les packages officiels Wisej-4

Wisej.NET publie ses familles d'icônes sous forme de packages NuGet, et pour une application Wisej.NET 4 c'est la famille `Wisej-4-*` qu'il vous faut : `Wisej-4-FontAwesome` , `Wisej-4-MaterialDesign` , `Wisej-4-ModernUI` , `Wisej-4-ElegantIcons` , `Wisej-4-VisualStudioIcons` , `Wisej-4-BootstrapIcons` , `Wisej-4-VaadinIcons` , `Wisej-4-FeatherIcons` , `Wisej-4-CSSGGIcons` , `Wisej-4-TablerIcons` et `Wisej-4-IonicIcons` . Wisej.NET 4.1 amène aussi Lucide dans le tableau ; vérifiez sa distribution auprès du profil NuGet officiel Wisej et des notes de version 4.1 plutôt que d'installer un package tiers sans rapport qui porterait le même nom.

Ajoutez-les comme n'importe quelle dépendance, depuis l'interface NuGet de Visual Studio ou avec `dotnet add package Wisej-4-BootstrapIcons` . Gardez la ligne majeure du pack alignée sur celle du framework : une application Wisej 4 utilise des packs Wisej-4, et les exemples copiés depuis un ancien package Wisej 2 ou Wisej 3 doivent être revérifiés contre le package courant plutôt que collés tels quels. Dans le lab IconDesk, vous référencerez deux de ces packages, ce qui fait exactement deux entrées `PackageReference` dans le fichier projet et deux assemblys de plus dans le suivi des dépendances du contrôle de version.

Une étape piège presque tout le monde : compilez après avoir ajouté la référence. L'explorateur d'images au design découvre les assemblys de packs d'icônes que le projet référence effectivement et contre lesquels il a compilé : un sélecteur ouvert avant la première compilation ne listera tout simplement pas la nouvelle famille.

De l'explorateur d'images au C#

Une fois les packages en place, le sélecteur d'images de Visual Studio gagne les nouvelles familles aux côtés des images de votre projet et des images de thème, et vous y parcourez et prévisualisez les icônes au lieu de deviner des noms de fichiers. Quand vous en choisissez une, le designer écrit pour vous la source d'image dans le `Designer.cs` généré, et cette chaîne générée mérite d'être lue au moins une fois, car elle montre tout le mécanisme en une ligne.

```
// designer-generated: the picker wrote the resource.wx URL for you
buttonSave.ImageSource =
    "resource.wx/Wisej.Ext.FontAwesome/android.svg?color=Coral";

// from C#: the pack's own icon catalog, so no path is typed by hand
// (the exact catalog namespace and class vary from pack to pack)
buttonSave.ImageSource = FontAwesome.Icons.Save;

// recolour a monochrome pack SVG with a theme colour name
buttonDelete.ImageSource =
    "resource.wx/Wisej.Ext.BootstrapIcons/trash.svg?color=error";
```

Les deux formes aboutissent au même endroit. Le catalogue est celui à préférer dans le code métier : il offre l'IntelliSense, il survit à un renommage à l'intérieur du pack, et il empêche les noms de fichiers

littéraux de se disperser dans des questionnaires où personne ne peut les auditer. Le suffixe `?color=` fonctionne sur les SVG des packs exactement comme sur vos propres SVG monochromes du module précédent, et pour la même raison : un visuel plein ou multicolore l'ignorera donc. C'est un fait à découvrir dans le lab sur deux icônes réelles, pas une hypothèse à emporter en production.

Pack, image de thème, ou votre propre pack

Trois mécanismes se disputent désormais le même bouton. Utilisez une **image de thème nommée** quand l'icône représente une sémantique standard du framework et doit changer quand le thème change. Utilisez un **pack d'icônes publié** quand vous avez besoin d'une large couverture des concepts applicatifs ordinaires. Construisez un **pack d'icônes d'application**, ce qui est l'objet du Module 6, quand les symboles sont propriétaires, liés à la marque, ou constituent un sous-ensemble sélectionné que vous voulez versionner vous-même. Les mécanismes cohabitent très bien ; ce qui casse un produit, c'est de les employer tous les trois au hasard pour le même type de commande.

Choisir la famille principale est une revue courte, à laquelle on peut répondre :

- Le pack couvre-t-il les concepts métier que le produit nomme réellement ?
- Ses styles en contour ou en aplat s'accordent-ils au thème ?
- Les icônes sont-elles des SVG monochromes qui acceptent le recolorage ?
- La licence amont convient-elle au mode de distribution de l'application ?
- Designers et développeurs reconnaîtront-ils la convention de nommage ?

Enfin, traitez le pack comme la dépendance qu'il est. C'est un assembly que vous livrez : consignez sa provenance et sa licence, gardez le `PackageReference` sous contrôle de version, et testez un changement de version avant la production. N'installez pas toutes les familles sous prétexte que c'est facile. Le lab se termine par une recommandation d'un paragraphe exactement pour cette raison : la preuve du panneau réseau, une requête par icône puis un accès au cache, est ce qui la rend défendable.



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 :

- ✓ Expliquer ce qu'est un pack d'icônes Wisej.NET et comment le navigateur obtient ses icônes.
- ✓ Installer et utiliser les packages NuGet officiels de packs d'icônes Wisej-4.
- ✓ Utiliser les ressources de packs d'icônes dans le designer de Visual Studio et depuis C#.

- ✓ Comparer les ressources de packs d'icônes aux images de thème et choisir le bon mécanisme pour une application.

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 IconDesk deux packages NuGet officiels de packs d'icônes Wisej-4 et construisez une page IconCompare qui montre les mêmes cinq concepts, Save, Delete, Search, User et Settings, tirés des deux familles, pour que la différence visuelle soit impossible à manquer. Choisissez au moins cinq des icônes via l'explorateur d'images de Visual Studio plutôt qu'en saisissant le chemin, affectez les autres depuis le catalogue du pack en C#, recolorez-en deux avec le suffixe `?color=`, et observez le panneau réseau pour voir les fichiers d'icônes arriver une requête à la fois, puis provenir du cache du navigateur. Concluez par une courte recommandation désignant la famille d'icônes principale du projet.

Point de départ

Ouvrez la solution IconDesk telle que vous l'avez laissée après le Module 3 : le panneau de commandes, la page ImageLab et la page IconGallery fonctionnent tous, et chaque image du projet est soit un fichier local, soit une image de thème, soit une URL que vous avez saisie vous-même. Le projet n'a encore aucune référence de pack d'icônes : le sélecteur d'images du designer n'affiche donc pour l'instant que les images du projet et du thème.

Approche suggérée

- ✓ Choisissez deux familles officielles Wisej-4 contrastées, ajoutez-les toutes deux avec l'interface NuGet ou `dotnet add package`, et compilez le projet avant d'ouvrir le moindre sélecteur.
- ✓ Ajoutez une page `IconCompare` avec deux colonnes étiquetées et une ligne par concept : Save, Delete, Search, User et Settings.
- ✓ Remplissez la première colonne depuis l'explorateur d'images de Visual Studio, puis ouvrez `IconCompare.Designer.cs` et lisez les chaînes `resource.wx` générées par le sélecteur.
- ✓ Remplissez la seconde colonne en C# depuis le catalogue d'icônes de chaque pack, et recolorez deux icônes en ajoutant le suffixe `?color=` avec un nom de couleur du thème.
- ✓ Rechargez avec le panneau réseau du navigateur ouvert, relevez les requêtes icône par icône puis les accès au cache au second chargement, et rédigez la recommandation à partir de ces preuves.

Livrables attendus

- ✓ Deux packages NuGet officiels de packs d'icônes Wisej-4 référencés par IconDesk
- ✓ Page IconCompare montrant Save, Delete, Search, User et Settings dans les deux packs
- ✓ Au moins cinq icônes choisies via l'explorateur d'images de Visual Studio plutôt que saisies à la main
- ✓ Deux icônes recolores avec le suffixe de couleur et une note indiquant quel visuel a accepté le recolorage

- ✓ Recommandation d'un paragraphe désignant la famille d'icônes principale du projet, avec les preuves de mise en cache par le navigateur qui la justifient

Questions de révision

- ✓ Référencer un pack de trois mille icônes ajoute un seul assembly au déploiement. Qu'est-ce que cela coûte au navigateur, et à quel moment ?
- ✓ L'une de vos deux icônes recolorées a ignoré le suffixe de couleur. Qu'a de particulier ce visuel, et que changeriez-vous dans les critères de sélection des packs à cause de cela ?
- ✓ Un coéquipier veut que chaque icône de commande suive le thème lorsque l'application en change. Lesquelles des cinq icônes d'IconCompare devraient devenir des images de thème, et lesquelles non ?



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.