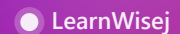


Ressources embarquées et resource.wx

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



Guide de la leçon

Livrer les images à l'intérieur de l'assembly : Build Action défini sur Embedded Resource, le gestionnaire resource.wx avec et sans qualificateur d'assembly, les extensions css, js, html, jpg, jpeg, png, gif, svg et bmp qu'il sait servir, et le fichier déployé qui peut remplacer une ressource embarquée sans recompilation. 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

Livrer les images à l'intérieur de l'assembly : Build Action défini sur Embedded Resource, le gestionnaire resource.wx avec et sans qualificateur d'assembly, les extensions css, js, html, jpg, jpeg, png, gif, svg et bmp qu'il sait servir, et le fichier déployé qui peut remplacer une ressource embarquée sans recompilation.

1. Le problème en production

Jusqu'ici, chaque ressource d'IconDesk est quelque chose que l'application a trouvé sur le disque ou demandé au thème. Cela fonctionne jusqu'à ce que quelqu'un déploie le site et oublie un dossier : le navigateur demande une icône, le serveur répond 404, et l'utilisateur voit un rectangle vide là où devrait se trouver le badge d'état. Personne ne le détecte en revue, parce que la machine du développeur a toujours le fichier.

L'embarquement fait du visuel une partie de la sortie de build : l'image voyage avec l'assembly, versionnée par le même numéro de build, impossible à perdre dans une étape de copie. C'est aussi le mécanisme sur lequel repose chaque pack d'icônes Wisej.NET, et c'est ce qui rend le Module 6 possible.

2. Concepts clés

Concept	Pourquoi c'est important
Build Action = Embedded Resource	La propriété de fichier de Visual Studio qui compile un fichier dans le manifeste de l'assembly au lieu de le copier à côté de l'exécutable, si bien que la ressource est livrée dans la DLL.
gestionnaire resource.wx	Le gestionnaire HTTP de Wisej.NET qui sert les ressources embarquées au navigateur ; ce que vous affectez est une chaîne de source d'image, jamais un objet image côté serveur.
URL de ressource non qualifiée	<code>resource.wx/company-logo.svg</code> demande à Wisej.NET de chercher dans les assemblies chargés une ressource portant ce nom, ce qui est pratique et légèrement ambigu.
URL de ressource qualifiée par l'assembly	<code>resource.wx/MyCompany.Branding/company-logo.svg</code> nomme l'assembly (sans <code>.dll</code>) pour que la résolution ne puisse pas tomber sur le fichier homonyme de quelqu'un d'autre.
types de ressources autorisés	Le gestionnaire documente exactement neuf extensions : <code>.css</code> , <code>.js</code> , <code>.html</code> , <code>.jpg</code> , <code>.jpeg</code> , <code>.png</code> , <code>.gif</code> , <code>.svg</code> et <code>.bmp</code> ; tout le reste revient en 404.
remplacement par fichier déployé	Un fichier déployé avec l'application peut être servi à la place de la ressource embarquée au même chemin, ce qui permet à une installation de rehabiller ou de corriger un visuel sans recompiler l'assembly.
nom de ressource de manifeste	Le nom que .NET stocke réellement, dérivé du nom de l'assembly, du namespace par défaut et du dossier où se trouve le fichier ; le deviner est la cause habituelle d'une URL de ressource qui ne renvoie rien.

3. Le fil de la leçon

L'embarquement est une action de build, pas un appel d'API

Il n'y a aucune méthode à appeler et aucun attribut à poser. Vous ajoutez le fichier au projet, ouvrez la fenêtre Properties, et changez Build Action de sa valeur par défaut à **Embedded Resource**. Le compilateur écrit les octets dans le manifeste de l'assembly, le fichier cesse d'apparaître dans le dossier de sortie, et dès lors la ressource fait partie du binaire. Le lab le fait cinq fois : `logo.svg`, `status-ok.svg`, `status-warning.svg`, `photo.png` et `badge.gif`, tous dans un dossier `Resources` de l'application IconDesk.

La même propriété fonctionne dans une bibliothèque de classes, et c'est le cas intéressant. Une bibliothèque dont l'unique rôle est de transporter des visuels est un sac d'icônes redistribuable : référencez-la, et chaque écran qui la référence peut adresser ces fichiers. Les packs d'icônes Wisej.NET sont exactement cela, plusieurs centaines de SVG dans un assembly, et le Module 6 vous fait construire le vôtre. Rien de la technique ne change entre une application et une bibliothèque ; seul le nom que vous devrez peut-être placer dans l'URL change.

Comparez avec les alternatives. Un fichier Content avec *Copy to Output Directory* reste séparé : facile à remplacer, facile à perdre. Une URL externe déplace le problème sur un autre hôte. Embarquez quand la

ressource appartient au code et doit être versionnée avec lui ; la décision est opérationnelle, pas stylistique.

L'adresser : resource.wx avec et sans nom d'assembly

Une fois le fichier embarqué, le navigateur l'atteint via le gestionnaire `resource.wx`. La valeur que vous affectez à `ImageSource` est une chaîne, et le client la résout comme une URL ; le serveur ne matérialise jamais de `System.Drawing.Image` pour elle. Cette distinction compte surtout pour le SVG, qui n'est pas du tout un format `System.Drawing` : c'est du markup que le navigateur rend, et c'est précisément pourquoi il reste net à 16, 24 ou 96 pixels.

```
// IconDesk - ResourceLab.cs
// every file below has Build Action = Embedded Resource

// unqualified: Wisej.NET searches the loaded assemblies
picStatusOk.ImageSource = "resource.wx/status-ok.svg";
picPhoto.ImageSource    = "resource.wx/photo.png";
picBadge.ImageSource     = "resource.wx/badge.gif";

// qualified with the assembly name, no ".dll" - no ambiguity
picLogo.ImageSource      = "resource.wx/IconDesk/logo.svg";

// the same URL works outside control properties, e.g. in CSS:
// .brand-mark { background-image: url('resource.wx/IconDesk/logo.svg'); }
```

Omettez le segment d'assembly et Wisej.NET cherche dans les assemblys chargés une ressource correspondante. C'est parfait tant qu'un seul assembly possède ce nom. Ajoutez une deuxième bibliothèque d'icônes, ou l'assembly de marque d'un client, et deux fichiers nommés `logo.svg` peuvent tous deux répondre. Qualifier le chemin lève la question et documente la propriété pour le prochain lecteur. Le lab vous fait qualifier exactement une source pour que la différence soit visible dans votre propre code.

Comme le gestionnaire parle du HTTP ordinaire, la même forme d'URL peut apparaître dans une feuille de style, un fragment de HTML ou un morceau de JavaScript, et pas seulement dans une propriété d'image. Notez aussi que l'attribut `[assembly: WisejResources]` relève d'une histoire plus large de découverte et de regroupement pour les scripts, styles et thèmes embarqués ; une URL de ressource fonctionne avec ou sans lui, alors n'ajoutez pas d'attributs mécaniquement juste pour afficher une image.

Ce que le gestionnaire sert et ce qu'il ne sert pas

La liste documentée est courte et mérite d'être mémorisée, car le mode de défaillance est silencieux. Le gestionnaire sert `.css`, `.js`, `.html`, `.jpg`, `.jpeg`, `.png`, `.gif`, `.svg` et `.bmp`. Six des neuf sont des images ; les trois autres existent parce qu'un assembly de bibliothèque transporte souvent une feuille de style ou un script à côté de ses visuels. Tout le reste — un fichier icône, une police, un PDF — revient en 404, aussi correctement qu'il ait été embarqué.

L'extension est donc une décision prise avant que le fichier n'entre dans le projet : un logo vectoriel devient `.svg`, une photographie `.png` ou `.jpg`, un marqueur animé `.gif`. Si quelqu'un vous remet un format que le gestionnaire ne sert pas, le convertir fait partie du travail.

Quand un fichier déployé l'emporte, et comment diagnostiquer un 404

Wisej.NET permet à un fichier déployé avec l'application de primer sur la ressource embarquée au même chemin logique. C'est délibéré et utile : une installation client peut déposer son propre `logo.svg` et l'application se rehabille sans que personne recompile ni republie l'assembly de ressources, et la même voie corrige à chaud une icône erronée. Le lab vous fait effectuer un remplacement et en capturer l'avant et l'après.

Le risque est que le répertoire de déploiement fasse désormais partie de votre frontière de confiance : quiconque peut y écrire des fichiers peut changer ce que l'application affiche. Le second risque est la dérive, car l'application en cours d'exécution ne correspond plus à l'assembly à partir duquel elle a été construite. Traitez la capacité de remplacement comme de la configuration, avec un responsable et un journal.

Quand une image n'apparaît pas, ne devinez pas les noms de manifeste. Déroulez plutôt une liste fixe ; chaque étape élimine une cause :

- Vérifiez que Build Action vaut bien Embedded Resource, et non Content ou None.
- Vérifiez que l'extension fait partie des neuf que le gestionnaire sert.
- Vérifiez que l'assembly est réellement chargé dans le processus en cours.
- Vérifiez l'assembly et les segments de chemin de l'URL, caractère par caractère.
- Regardez si un changement de namespace par défaut, de dossier ou de nom logique a déplacé le nom de ressource de manifeste.
- Recompilez, puis demandez l'URL exacte dans le navigateur et lisez le code de statut.



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 :

- ✓ Embarquer les fichiers image pris en charge dans un assembly d'application ou de bibliothèque de classes.
- ✓ Utiliser `resource.wx` avec et sans nom d'assembly.
- ✓ Énumérer les types de ressources pris en charge par le gestionnaire de ressources embarquées de Wisej.NET.
- ✓ Expliquer comment des fichiers déployés peuvent remplacer des ressources embarquées et pourquoi cela peut être utile.

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 l'application IconDesk un dossier Resources contenant logo.svg, status-ok.svg, status-warning.svg, photo.png et badge.gif, et passez chacun d'eux en Build Action Embedded Resource. Construisez une page ResourceLab qui les affiche tous les cinq via des sources d'image resource.wx, écrivez l'une de ces sources en plaçant le nom de l'assembly de l'application devant le fichier pour que la résolution ne puisse pas être ambiguë, et laissez les autres non qualifiées. Déposez ensuite un fichier de remplacement à côté de l'application déployée pour remplacer une ressource embarquée, et capturez l'avant et l'après. Terminez en écrivant délibérément une URL de ressource mal orthographiée et en notant les étapes qui ont permis d'identifier la panne.

Point de départ

Ouvrez IconDesk tel qu'il est après le Module 4 : le panneau de commandes, la page ImageLab, la page IconGallery pilotée par le designer et la page IconCompare qui place deux packs d'icônes officiels côte à côte. Toutes les ressources vues jusqu'ici viennent du thème, d'un fichier de projet séparé, d'une URL ou d'un assembly de pack ; rien n'est encore livré dans le binaire d'IconDesk lui-même, et il n'y a pas de dossier **Resources** dans le projet.

Approche suggérée

- ✓ Ajoutez d'abord le dossier **Resources** et les cinq fichiers, puis définissez Build Action sur Embedded Resource pour chacun et recompilez avant d'écrire la moindre ligne de C#.
- ✓ Ajoutez une page **ResourceLab** avec cinq contrôles PictureBox ou Label et affectez à chaque **ImageSource** une chaîne **resource.wx**, en ne qualifiant que le logo avec le nom d'assembly **IconDesk**.
- ✓ Exécutez la page, ouvrez le panneau réseau du navigateur, et vérifiez que chaque ressource renvoie 200 et que les SVG restent nets quand vous les agrandissez.
- ✓ Publiez ou copiez l'application dans un dossier de déploiement, déposez un fichier de remplacement par-dessus une ressource embarquée, rechargez sans recompiler, et prenez une capture des deux états.
- ✓ Cassez une URL à dessein, puis déroulez la liste de dépannage dans l'ordre et notez quelle étape a révélé la cause.

Livrables attendus

- ✓ Dossier Resources avec logo.svg, status-ok.svg, status-warning.svg, photo.png et badge.gif tous marqués Embedded Resource
- ✓ Page ResourceLab affichant chaque ressource embarquée via une source d'image resource.wx
- ✓ Une source d'image qualifiée par le nom de l'assembly de l'application et les autres laissées non qualifiées

- ✓ Fichier déployé qui remplace une ressource embarquée, capturé avant et après
- ✓ Note de dépannage listant les étapes qui ont permis d'identifier une URL de ressource mal orthographiée

Questions de révision

- ✓ Deux assemblés référencés embarquent tous deux un fichier nommé `logo.svg`. Vers quoi une source non qualifiée se résout-elle, et que changeriez-vous pour que la réponse cesse de dépendre de l'ordre de chargement ?
- ✓ Votre remplacement fonctionne sur le serveur de test mais plus après la version suivante. Quelles parties du processus de déploiement ont pu le supprimer, et où le remplacement aurait-il dû être consigné ?
- ✓ Un coéquipier embarque `icon.ico` et signale que `resource.wx` est cassé. Que se passe-t-il réellement, et quelle est la plus petite correction possible ?



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.