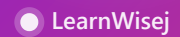


Bitmaps, SVG et PictureBox : formats, dimensionnement et chargement

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



Guide de la leçon

Deux listes de formats parce qu'il y a deux pipelines : bitmap, métafichier, icône, JPEG, GIF et PNG via `System.Drawing` et `PictureBox.Image`, face aux `jpg`, `jpeg`, `png`, `gif`, `svg` et `bmp` que `resource.wx` et les images de thème acceptent sous forme de chaînes, avec les modes de dimensionnement du `PictureBox` et les points d'entrée `Load` synchrone et `LoadAsync` asynchrone. 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

Deux listes de formats parce qu'il y a deux pipelines : bitmap, métafichier, icône, JPEG, GIF et PNG via `System.Drawing` et `PictureBox.Image`, face aux `jpg`, `jpeg`, `png`, `gif`, `svg` et `bmp` que `resource.wx` et les images de thème acceptent sous forme de chaînes, avec les modes de dimensionnement du `PictureBox` et les points d'entrée `Load` synchrone et `LoadAsync` asynchrone.

1. Le problème en production

« Quels formats d'image Wisej.NET prend-il en charge ? » est la question que toute équipe pose dès la première semaine, et la réponse honnête dépend du chemin d'API qu'emprunte l'image. Répondez par une seule liste vague et vous obtiendrez un logo étiré sur un portable haute densité, une page qui se fige pendant qu'une photo de produit se télécharge depuis un hôte lent, et un rapport de bug parce qu'un fichier `.ico` refuse de se résoudre via une chaîne de ressource. Ce module remplace la liste vague par deux listes précises, chacune attachée à la propriété de contrôle qui la consomme.

2. Concepts clés

Concept	Pourquoi c'est important
pipeline System.Drawing	La propriété <code>Image</code> contient un <code>System.Drawing.Image</code> dans la mémoire du serveur ; Wisej.NET l'envoie au navigateur sous forme de données PNG. Ses sources documentées sont les fichiers bitmap, métafichier, icône, JPEG, GIF et PNG.
pipeline des sources d'image	Une chaîne que le client résout lui-même : une URL relative ou absolue, une image de thème nommée, ou un chemin <code>resource.wx</code> . Les extensions d'image <code>resource.wx</code> documentées sont .jpg, .jpeg, .png, .gif, .svg et .bmp.
PictureBox.Image	Le point d'entrée pour des pixels que votre code C# possède déjà ou vient de générer, par exemple un graphique dessiné avec <code>Graphics</code> .
PictureBox.ImageSource	Le point d'entrée pour une image que le navigateur peut récupérer lui-même. Une courte chaîne voyage à la place des octets de l'image, et c'est pourquoi le SVG a sa place ici.
PictureBox.SizeMode	Les six valeurs <code>Normal</code> , <code>AutoSize</code> , <code>CenterImage</code> , <code>StretchImage</code> , <code>Zoom</code> et <code>Cover</code> , qui décident si une image est coupée, centrée, déformée, ajustée ou rognée.
Load et LoadAsync	<code>Load(url)</code> met à jour l'image depuis une URL de façon synchrone ; <code>LoadAsync(url)</code> effectue la même récupération de façon asynchrone, si bien qu'un hôte lent ne bloque pas l'écran.
vectoriel contre raster	Un SVG est du markup dessiné à la taille que réclame la mise en page, et un visuel monochrome adapté peut être recoloré ; un fichier raster est une grille de pixels figée qui ne fait jamais que se rééchantillonner.

3. Le fil de la leçon

Deux listes de formats, parce qu'il y a deux pipelines

Le premier objectif de ce module est d'arrêter de citer une liste unique. Pour le `PictureBox` , la documentation des contrôles Wisej.NET décrit des graphiques chargés depuis des fichiers bitmap, métafichier, icône, JPEG, GIF et PNG. C'est le monde familier du `System.Drawing.Image` : vous confiez un objet au contrôle, et le serveur le transforme en données PNG sur le réseau. Pour les ressources en chaîne, les extensions d'image `resource.wx` documentées sont .jpg, .jpeg, .png, .gif, .svg et .bmp, et les définitions d'images de thème acceptent png, gif, svg, jpg et bmp.

Les entrées intéressantes sont celles qui n'apparaissent que dans une seule liste. L'icône et le métafichier appartiennent au côté System.Drawing et ne font pas partie de la liste d'images autorisées par `resource.wx` . Le SVG, c'est l'inverse : c'est une excellente valeur de source d'image et une image de thème de premier ordre, mais ce n'est pas un format `System.Drawing.Image` au même titre qu'un Bitmap ou un PNG. Dire « Wisej.NET prend en charge le SVG » sans nommer le pipeline, c'est ce qui produit une affectation `Image` ratée plus tard dans le sprint.

Dans le lab, vous l'écrivez explicitement : la note de formats à côté de votre page ImageLab liste chaque ressource et nomme le pipeline qui l'a transportée.

Le PictureBox a deux points d'entrée, et vous en choisissez un par contrôle

Le deuxième objectif porte sur le choix de la bonne propriété. `PictureBox.Image` reçoit un `System.Drawing.Image`. `PictureBox.ImageSource` reçoit une chaîne. La règle qui garde un projet lisible est simple : si votre code C# crée ou transforme les pixels, utilisez `Image` ; si l'image est un fichier que le navigateur pourrait tout aussi bien récupérer lui-même, utilisez `ImageSource` et gardez une chaîne parlante.

```
pictureBox1.ImageSource = "Images/product.png"; // a string the client resolves
pictureBox1.SizeMode = PictureBoxSizeMode.Zoom;

pictureBox2.ImageSource = "resource.wx/IconDesk/Logo.svg";
pictureBox2.SizeMode = PictureBoxSizeMode.Cover;

// a URL the user chose: fetch it without holding the page
pictureBox3.LoadAsync(remoteJpegUrl);
```

Les points d'entrée par URL se placent à côté de ces propriétés. `Load(url)` actualise l'image depuis une URL de façon synchrone ; `LoadAsync(url)` démarre au contraire un chargement asynchrone. Votre page ImageLab utilise `LoadAsync` pour le JPEG distant précisément pour qu'un hôte tiers lent se dégrade en « l'image arrive un instant plus tard » plutôt qu'en « la page est figée ».

Six modes de dimensionnement, et ce que chacun fait à l'image

Wisej.NET documente six valeurs de `PictureBoxSizeMode`, et le troisième objectif est de choisir entre elles délibérément au lieu d'accepter ce que le designer a laissé. `Normal` dessine l'image à sa taille d'origine depuis le coin supérieur gauche et coupe ce qui ne tient pas. `AutoSize` inverse la relation : le PictureBox grandit ou rétrécit à l'image. `CenterImage` conserve la taille d'origine mais la centre, si bien que le rognage se répartit également sur tous les côtés. `StretchImage` remplit la zone, que les proportions y survivent ou non. `Zoom` met l'image à l'échelle en préservant ses proportions jusqu'à ce qu'elle tienne entièrement. `Cover` préserve lui aussi les proportions mais remplit le rectangle, en rognant les bords qui débordent.

En pratique, `Zoom` est le choix sûr pour les logos et les photos qui doivent rester entièrement visibles, `Cover` est le bon choix pour les vignettes et les bannières où remplir le rectangle compte plus que montrer chaque bord, et `StretchImage` doit être une décision prise sciemment, car c'est le mode qui déforme discrètement une marque. Le lab affiche un même visuel raster quatre fois, en `Normal`, `Zoom`, `Cover` et `StretchImage`, pour que le rognage et la déformation soient visibles côte à côte plutôt que décrits.

Le SVG pour les icônes d'interface, le raster pour les photographies

Le quatrième objectif est la règle de conception qui découle des deux pipelines. Le SVG est généralement le premier choix pour les icônes d'interface : c'est du markup vectoriel, il reste donc net à 16, 24, 32 et 48 pixels comme à tout facteur d'échelle d'affichage, un visuel monochrome simple a une empreinte réduite, et Wisej.NET peut recolorer un SVG adapté via la chaîne de source d'image. Les photographies et les

captures d'écran vont dans l'autre sens. Le PNG est le choix raster polyvalent quand la transparence ou l'exactitude des pixels compte, le JPEG convient normalement mieux aux photographies où une compression avec perte est acceptable, le GIF reste utile pour les graphiques simples ou animés, et le BMP est pris en charge mais est en général bien plus lourd qu'un format web compressé.

C'est en haute densité que la différence cesse d'être théorique. Une icône vectorielle n'a besoin d'aucune seconde copie pour une mise à l'échelle à 150 % ou 200 % ; un bitmap de 16 pixels agrandi à 32 pixels a simplement l'air agrandi. Si vous devez livrer des icônes raster, partez d'une résolution suffisante pour la plus grande taille que vous dessinerez, et testez dans le navigateur avec une échelle d'affichage augmentée.

- Glyphes de commande, de navigation et d'état : SVG via `ImageSource`.
- Photos de produits, captures d'écran et numérisations : PNG ou JPEG, généralement avec `Zoom` ou `Cover`.
- Pixels que votre propre code dessine : un `System.Drawing.Image` sur `Image`.

Le dimensionnement est une décision de conception, pas un accident

Une ressource image a des dimensions intrinsèques, mais votre interface a aussi une taille d'icône cible, et les deux ne sont pas le même nombre. Les thèmes Wisej contrôlent généralement les métriques d'icônes, et une `ImageList` peut établir une taille commune explicite et redimensionner les images que les contrôles dessinent. Pour un Label, la documentation précise que la taille d'image d'origine est utilisée à moins qu'une `ImageList` n'intervienne.

Le dernier livrable boucle la boucle avec le Module 1 : un gestionnaire de rapport parcourt chaque `PictureBox` de la page `ImageLab` et indique si son image est arrivée par `Image` ou par `ImageSource`. Ce gestionnaire est le moyen le plus rapide de trouver un contrôle sur lequel quelqu'un a défini les deux propriétés et où une seule l'emporte.



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 :

- ✓ Identifier les formats d'image que Wisej.NET accepte via le `PictureBox`, les thèmes et `resource.wx`.
- ✓ Utiliser correctement `PictureBox.Image` et `PictureBox.ImageSource`.
- ✓ Choisir les modes de dimensionnement du `PictureBox`, dont `Normal`, `AutoSize`, `CenterImage`, `StretchImage`, `Zoom` et `Cover`.

- ✓ Expliquer pourquoi le SVG est généralement préféré pour les icônes d'interface redimensionnables et pourquoi les photos sont généralement des images raster.

Guide du lab / de l'examen

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



Lab pratique

Construisez dans IconDesk une page ImageLab qui rend les deux pipelines visibles côte à côte. Affichez un même visuel raster quatre fois avec `SizeMode` défini sur Normal, Zoom, Cover et `StretchImage` pour que les différences de rognage et de déformation sautent aux yeux, ajoutez un `PictureBox` qui récupère un JPEG distant avec `LoadAsync` et garde la page réactive pendant son arrivée, et placez un SVG affecté via `ImageSource` à 16, 24, 32 et 48 pixels pour prouver qu'il reste net. Terminez par un gestionnaire qui indique, pour chaque `PictureBox` de la page, si son image vient d'`Image` ou d'`ImageSource`.

Point de départ

Ouvrez la solution IconDesk créée au Module 1, où le panneau de commandes porte déjà quatre icônes de boutons par quatre mécanismes différents. Ajoutez au projet une nouvelle page nommée `ImageLab` et gardez à portée de main un visuel raster, un JPEG de type photographie hébergé sur un hôte distant et une icône SVG monochrome avant de commencer à poser des contrôles.

Approche suggérée

- ✓ Disposez sur ImageLab quatre contrôles `PictureBox` de taille identique et donnez-leur le même visuel raster, puis définissez `SizeMode` sur Normal, Zoom, Cover et `StretchImage` et étiquetez chacun avec le mode qu'il illustre.
- ✓ Ajoutez un cinquième `PictureBox` et appelez `LoadAsync` avec l'URL du JPEG distant depuis le chemin de chargement de la page, en laissant un espace réservé ou une légende visible pour que l'utilisateur voie que la page est restée utilisable pendant que l'image était en route.
- ✓ Placez quatre autres contrôles `PictureBox` dimensionnés à 16, 24, 32 et 48 pixels, affectez le même SVG à chacun via `ImageSource`, et comparez-les au visuel raster mis à l'échelle aux quatre mêmes tailles.
- ✓ Écrivez un gestionnaire `btnReport_Click` qui parcourt les contrôles de la page et, pour chaque `PictureBox`, écrit une ligne nommant le contrôle et indiquant si son image vient d'`Image` ou d'`ImageSource`.
- ✓ Rédigez la note de formats : listez toutes les ressources utilisées par la page, indiquez lesquelles transitent par `System.Drawing` et lesquelles restent une chaîne de ressource, et consignez le mode de dimensionnement que vous défendriez en revue pour un logo, une vignette et une photographie.

Livrables attendus

- ✓ Page ImageLab affichant un même visuel raster dans les modes de dimensionnement Normal, Zoom, Cover et `StretchImage`
- ✓ `PictureBox` qui récupère un JPEG distant avec `LoadAsync` sans bloquer la page
- ✓ Un SVG affecté via `ImageSource` et rendu à 16, 24, 32 et 48 pixels

- ✓ Gestionnaire de rapport qui nomme, pour chaque PictureBox, si l'image vient d'Image ou d'ImageSource
- ✓ Note de formats listant les ressources du lab qui passent par System.Drawing et celles qui restent des chaînes de ressource

Questions de révision

- ✓ Votre visuel raster est plus large que haut et la zone est carrée. Lequel de Zoom et de Cover perd une partie de l'image, lequel laisse de l'espace vide, et lequel défendriez-vous pour le logo d'un client ?
- ✓ L'hôte du JPEG distant est inaccessible pendant dix secondes. Que vit l'utilisateur de votre page avec `LoadAsync`, et qu'aurait-il vécu avec le `Load` synchrone ?
- ✓ Votre gestionnaire de rapport indique que l'image d'un contrôle vient d' `ImageSource`, mais le fichier designer affecte aussi `Image` sur le même contrôle. Laquelle est réellement dessinée, et comment le prouveriez-vous plutôt que de le deviner ?



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.