

Localiser l'interface Wisej.NET dans le designer Visual Studio

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



Guide de la leçon

La localisation dans le designer sur un conteneur Wisej.NET de premier niveau : la propriété Localizable, le sélecteur Language du designer, les propriétés qui peuvent légitimement varier d'une langue à l'autre comme Text, Size, Location, les couleurs et les images, les fichiers de ressources que le designer génère, et la règle selon laquelle les ressources du designer sont appliquées au moment de la création des contrôles. Ce que cela signifie pour un développeur qui connaît les bases de C# et les contrôles Wisej.NET et doit maintenant livrer son application dans plusieurs langues — et comment aborder ce module.



Objectif en production

La localisation dans le designer sur un conteneur Wisej.NET de premier niveau : la propriété Localizable, le sélecteur Language du designer, les propriétés qui peuvent légitimement varier d'une langue à l'autre comme Text, Size, Location, les couleurs et les images, les fichiers de ressources que le designer génère, et la règle selon laquelle les ressources du designer sont appliquées au moment de la création des contrôles.

1. Le problème en production

Le Module 1 a donné à GlobalDesk un tableau de bord dont tous les libellés viennent de `Strings.resx`, et c'est bien le bon foyer pour le texte que le code génère. L'éditeur client, lui, est différent : c'est un écran conçu, agencé à l'œil dans le designer Visual Studio. Faire passer chacun de ses libellés par un appel à une classe utilitaire oblige à retoucher le fichier du designer à la main, et cela ne résout toujours pas le vrai problème : un *Kundennummer speichern* allemand ne tient pas dans un bouton dimensionné pour *Save*. Ce module utilise le flux de localisation que Visual Studio propose déjà, de sorte que les libellés d'un écran conçu et sa géométrie peuvent varier d'une langue à l'autre, sans chaîne codée en dur et sans second fichier de mise en page.

2. Concepts clés

Concept	Pourquoi c'est important
<code>Localizable</code>	Une propriété de design sur un conteneur de premier niveau ; <code>true</code> fait basculer le designer dans le flux de localisation, pour que les valeurs puissent être stockées par culture.
Sélecteur <code>Language</code> du designer	Apparaît dès que <code>Localizable</code> vaut <code>true</code> ; la culture sélectionnée est la version que vous modifiez, si bien que chaque changement n'est enregistré que pour cette culture.
La langue neutre	La version par défaut sur laquelle toutes les cultures retombent, et la seule où l'on ajoute, supprime ou restructure des contrôles ; les vues de culture contiennent des redéfinitions, pas un second écran.
Propriétés localisables	Pas seulement <code>Text</code> : <code>Size</code> , <code>Location</code> , les couleurs et les images peuvent chacune avoir une valeur différente par culture, parce que la traduction change la géométrie autant que les mots.
Fichiers de ressources générés	Le designer écrit les valeurs localisées dans des fichiers de ressources plutôt que dans le code C# du designer, ce qui garde les valeurs propres à une langue hors du code que vous lisez et fusionnez au quotidien.
Appliquées à la création	Les valeurs localisées dans le designer sont appliquées pendant la création des contrôles : changer la culture de la session plus tard ne réécrit donc pas un conteneur qui existe déjà.
Mise en page tolérante à la traduction	Le docking, l'ancrage, <code>FlowLayoutPanel</code> , <code>TableLayoutPanel</code> , <code>FlexLayoutPanel</code> , <code>AutoSize</code> et le padding laissent le texte grandir de lui-même : il vous faut donc moins de redéfinitions de géométrie.

3. Le fil de la leçon

Activer Localizable

La localisation dans le designer s'active sur le conteneur, pas sur les contrôles individuels. Ouvrez un `Form` , une `Page` , un `UserControl` ou un `Desktop` Wisej.NET en mode conception, sélectionnez le conteneur lui-même dans la fenêtre Properties, et passez `Localizable` à `true` . Tous les contrôles que le conteneur possède sont alors couverts par le flux ; vous ne définissez jamais la propriété sur un `Button` ou un `Label` enfant.

Faites-le une fois la mise en page neutre terminée, pas avant. Dans le lab, vous terminez d'abord le contrôle utilisateur `CustomerEditor` anglais de GlobalDesk : champs, group box, boutons Save et Cancel, docking. Actionner l'interrupteur à mi-chemin revient à prendre des décisions structurelles pendant qu'une culture est sélectionnée : une décision qui appartient à toutes les langues se retrouve enregistrée pour l'une d'elles seulement.

Modifier via le sélecteur Language

Avec `Localizable` activé, le designer expose un sélecteur `Language` sur le conteneur. Laissez-le sur l'entrée par défaut pour modifier la version neutre. Choisissez une culture comme l'allemand et la surface de conception devient cette version localisée de l'écran : elle montre toujours votre mise en page, mais chaque propriété que vous changez désormais est enregistrée comme une redéfinition pour cette seule culture.

La discipline est simple : ne changez que ce qui doit différer. Dans le lab, vous passez à l'allemand, traduisez les libellés, élargissez les deux boutons les plus longs et déplacez un label, sans toucher à rien d'autre. Puis vous revenez à la langue par défaut avant tout changement structurel, car ajouter un contrôle pendant que l'allemand est sélectionné est la façon dont une équipe se retrouve avec un écran qui possède un contrôle dans une langue et pas dans une autre.

Plus que Text : Size, Location, couleurs et images

Wisej.NET cite `Size`, `Location`, `Text`, les couleurs et les images comme des propriétés qui peuvent différer d'une langue à l'autre, et cette liste n'est pas décorative. Les noms composés allemands sont longs : un bouton dimensionné pour *Save* tronque *Speichern*. Les boîtes de dialogue françaises demandent souvent plus de hauteur. Une illustration parlante sur un marché peut n'avoir aucun sens sur un autre : une `Image` est donc elle aussi une valeur localisée légitime.

Les caractères mnémoniques relèvent de la même catégorie : c'est le libellé traduit qui décide quelle lettre d'accès rapide est disponible et parlante dans cette langue, alors ne supposez jamais que celle de l'anglais se transpose. Et si les couleurs sont localisables, les changer selon la culture est plus rare que de changer la géométrie : traitez donc une couleur propre à une langue comme une décision que vous pouvez justifier, et non comme un réflexe.

Ce que le designer écrit, et où cela va

Quand vous modifiez une vue de culture, le designer n'ajoute pas une branche à `InitializeComponent`. Il stocke les valeurs de propriétés localisées dans des fichiers de ressources générés à côté du conteneur, un jeu par culture que vous avez modifiée, ce qui garde les valeurs propres à une langue hors du code C# principal du designer. Cette séparation rend le flux vivable : votre code de designer reste lisible, les valeurs par langue sont des données, et un relecteur peut lire un fichier de ressources sans lire de C#.

Elle trace aussi une frontière pour le reste de l'application. Les ressources du designer appartiennent à un écran conçu : libellés fixes, texte des boutons, info-bulles, images statiques et géométrie par langue y ont leur place. Tout ce qui est généré dans le code, tout message produit par un service, tout texte de validation et tout ce que plusieurs écrans réutilisent relève d'un fichier de ressources d'application comme `Strings.resx`, lu via `Wisej.Resources.ResourceManager` — exactement ce que le Module 1 a construit. GlobalDesk garde les deux.

Les ressources du designer sont appliquées à la création des contrôles

C'est la règle qui surprend. Les valeurs localisées dans le designer sont appliquées pendant la construction de l'arbre des contrôles. Définir `Application.CurrentCulture` ensuite change la culture de la session, et le code qui lit des ressources à partir de cet instant adopte la nouvelle langue, mais cela ne revient pas en

arrière pour réécrire les propriétés de contrôles qui existent déjà. Un utilisateur qui change de langue et voit la moitié de l'éditeur encore en anglais a rencontré cette règle, pas un bug.

Vous avez deux options honnêtes : recréer le conteneur localisé pour que ses valeurs de ressources soient appliquées sous la nouvelle culture, ou le garder et écrire du code qui réapplique vous-même les propriétés issues des ressources. Pour un écran de cette taille, recréer est plus simple, et c'est pourquoi le lab ajoute un bouton Recreate editor qui libère le contrôle et en construit un nouveau.

```
private void btnRecreateEditor_Click(object sender, EventArgs e)
{
    // Designer resources are applied while the controls are created,
    // so the new culture only reaches this screen on a rebuild.
    this.hostPanel.Controls.Remove(this.customerEditor);
    this.customerEditor.Dispose();

    this.customerEditor = new CustomerEditor { Dock = DockStyle.Fill };
    this.hostPanel.Controls.Add(this.customerEditor);
}
```

Concevoir des mises en page qui survivent à des traductions plus longues

Chaque redéfinition de géométrie que vous enregistrez est une valeur que quelqu'un devra maintenir dans toutes les langues à venir. La redéfinition la moins chère est celle dont vous n'avez jamais eu besoin : construisez donc la mise en page neutre avec des éléments qui s'adaptent. Dockez un panneau au lieu de l'épingler, ancrez un bouton au bord droit au lieu de figer sa **Location**, laissez un **FlowLayoutPanel**, un **TableLayoutPanel** ou un **FlexLayoutPanel** placer une rangée, activez **AutoSize** là où un libellé doit décider de sa propre largeur, et laissez du padding pour qu'un mot plus long ait où aller.

Puis prouvez-le. Une vue de designer n'est pas la réponse finale : exécutez GlobalDesk dans le navigateur sous la culture cible. Cherchez les libellés tronqués, les boutons qui se chevauchent, les labels qui débordent sur un voisin, un ordre de tabulation qui ne suit plus l'ordre visuel, les images, et chaque boîte de dialogue que l'écran peut ouvrir. Le lab se termine par une note listant les propriétés que vous avez redéfinies pour l'allemand et celles que vous avez délibérément laissées à la mise en page neutre, et c'est la seconde liste la plus intéressante : elle prouve que c'est votre mise en page qui fait le travail, et non vos redéfinitions.



Réflexe des couches de localisation

Décidez où vit chaque chaîne avant de la taper. Les ressources du designer appartiennent à un écran conçu et sont appliquées à la création de ses contrôles ; les ressources d'application partagées dans les .resx portent le texte que le code génère ou que plusieurs écrans réutilisent ; Resources-[LANG].resx redéfinit les libellés système de Wisej.NET. Ne stockez jamais une date, un nombre ou un montant déjà mis en forme : gardez la valeur et mettez-la en forme pour la culture active au moment de l'affichage. Et souvenez-vous que la culture est propre à la session : la langue d'un utilisateur ne doit jamais changer l'écran d'un autre.



Objectifs d'apprentissage

À la fin de ce module, vous saurez :

- ✓ Activer la localisation sur un Form, une Page, un UserControl ou un Desktop Wisej.NET.
- ✓ Utiliser le sélecteur Language du designer pour modifier des propriétés localisées.
- ✓ Expliquer quelles propriétés changent couramment d'une langue à l'autre : Text, Size, Location, Color et Images.
- ✓ Comprendre ce que le designer écrit dans les fichiers .resx.
- ✓ Concevoir des mises en page qui survivent à des traductions plus longues.

Guide du lab / de l'examen

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



Lab pratique

Localisez l'éditeur client de GlobalDesk depuis le designer Visual Studio. Terminez d'abord la mise en page neutre en anglais du contrôle utilisateur CustomerEditor, puis passez Localizable à true et utilisez le sélecteur Language du designer pour ajouter une variante allemande, en ne changeant que ce qui doit différer : les libellés, la largeur des deux boutons les plus longs et la propriété Location d'un libellé. Exécutez l'application en de-DE et comparez le rendu réel du navigateur avec la vue du designer, puis ajoutez un bouton Recreate editor qui libère et reconstruit le contrôle pour constater par vous-même pourquoi les ressources du designer ne suivent pas un changement de culture à l'exécution. Terminez par une note de lab qui liste les propriétés que vous avez redéfinies et celles que vous avez délibérément laissées à la mise en page neutre.

Point de départ

Vous ouvrez GlobalDesk tel que le Module 1 l'a laissé, où la page tableau de bord lit déjà chaque libellé depuis `Resources/Strings.resx` via la classe utilitaire `Texts` et met en forme ses valeurs d'aperçu avec `Application.CurrentCulture`. Dans ce lab, vous travaillez sur le contrôle utilisateur `CustomerEditor` qu'ouvre le bouton Customers du tableau de bord : les champs code client, nom, ville et notes avec leurs labels, un group box et les boutons Save et Cancel.

Approche suggérée

- ✓ Terminez la mise en page neutre en anglais de `CustomerEditor` avant de localiser quoi que ce soit : dockez le champ de notes, ancrez Save et Cancel en bas à droite, placez les paires label-éditeur dans un conteneur de mise en page comme un `TableLayoutPanel` ou un `FlexLayoutPanel`, et activez `AutoSize` sur les labels qui doivent se dimensionner eux-mêmes.
- ✓ Sélectionnez le contrôle utilisateur lui-même dans le designer, passez `Localizable` à `true` dans la fenêtre Properties, et vérifiez que le sélecteur `Language` apparaît désormais ; laissez-le sur l'entrée par défaut et recompilez une fois pour que la version neutre serve de référence.
- ✓ Passez le sélecteur `Language` à l'allemand et traduisez les libellés seulement — les labels, le titre du group box, Save et Cancel. Élargissez ensuite les deux boutons dont le texte allemand ne tient plus en modifiant leur `Size`, et déplacez un label avec sa `Location` pour que la colonne reste alignée. Revenez à la langue par défaut avant de toucher à quoi que ce soit de structurel.
- ✓ Exécutez GlobalDesk en `de-DE` et vérifiez le navigateur, pas le designer : cherchez les libellés tronqués, les contrôles qui se chevauchent, un label qui déborde sur son voisin, l'ordre de tabulation et les boîtes de dialogue que l'éditeur peut ouvrir. Consignez chaque différence trouvée entre la vue du designer et la page réelle.
- ✓ Ajoutez sur la page hôte un bouton **Recreate editor** qui retire le `CustomerEditor` courant de son conteneur, le libère, et en ajoute un fraîchement construit. Changez la culture, appuyez sur le bouton, et regardez les libellés suivre enfin ; rédigez ensuite la note qui liste chaque propriété redéfinie pour l'allemand et chacune de celles que vous avez laissées à la mise en page neutre.

Livrables attendus

- ✓ Contrôle utilisateur CustomerEditor avec Localizable à true sur une mise en page neutre en anglais complète
- ✓ Variante de langue allemande dans le designer qui redéfinit les libellés ainsi qu'au moins une Size et une Location
- ✓ Ancrage, docking ou AutoSize appliqués pour que les libellés allemands ne puissent pas être tronqués
- ✓ Bouton Recreate editor qui libère et reconstruit le conteneur après un changement de culture
- ✓ Note de lab listant les propriétés redéfinies et celles laissées à la mise en page neutre

Questions de révision

- ✓ Un coéquipier ajoute un nouveau champ téléphone à `CustomerEditor` pendant que l'allemand est sélectionné dans la liste `Language`. Que verra probablement l'utilisateur anglais, et quelle règle l'aurait évité ?
- ✓ La confirmation d'enregistrement que l'éditeur affiche après une mise à jour réussie vient toujours de `Strings.resx` plutôt que des ressources du designer. Donnez deux raisons pour lesquelles c'est le bon endroit, et nommez un libellé de cet écran qui relève au contraire vraiment du designer.
- ✓ Votre bouton Recreate editor fonctionne, mais un collègue soutient que l'écran devrait simplement réappliquer les propriétés localisées au lieu de se reconstruire. Que devrait connaître ce code que la reconstruction ignore, et dans quel cas son approche serait-elle le meilleur compromis ?



Réflexe des couches de localisation

Décidez où vit chaque chaîne avant de la taper. Les ressources du designer appartiennent à un écran conçu et sont appliquées à la création de ses contrôles ; les ressources d'application partagées dans les `.resx` portent le texte que le code génère ou que plusieurs écrans réutilisent ; `Resources-[LANG].resx` redéfinit les libellés système de Wisej.NET. Ne stockez jamais une date, un nombre ou un montant déjà mis en forme : gardez la valeur et mettez-la en forme pour la culture active au moment de l'affichage. Et souvenez-vous que la culture est propre à la session : la langue d'un utilisateur ne doit jamais changer l'écran d'un autre.