

Langues de droite à gauche et mises en page compatibles avec la localisation

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



Guide de la leçon

La prise en charge du droite à gauche dans Wisej.NET : `RightToLeft` avec `Inherit`, `No` et `Yes`, `RightToLeftLayout` qui met en miroir les enfants d'un conteneur, le paramètre d'application `rightToLeft`, des mises en page qui survivent à l'allongement du texte, l'état `rightToLeft` du thème, et les limites documentées sur `DataGridView`, `ListView` et `PropertyGrid`. 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 prise en charge du droite à gauche dans Wisej.NET : `RightToLeft` avec `Inherit`, `No` et `Yes`, `RightToLeftLayout` qui met en miroir les enfants d'un conteneur, le paramètre d'application `rightToLeft`, des mises en page qui survivent à l'allongement du texte, l'état `rightToLeft` du thème, et les limites documentées sur `DataGridView`, `ListView` et `PropertyGrid`.

1. Le problème en production

GlobalDesk sait déjà changer de langue en cours d'exécution : le sélecteur définit

`Application.CurrentCulture`, le gestionnaire `Application.CultureChanged` réapplique chaque libellé issu des ressources, et le panneau d'aperçu remet ses exemples en forme. Puis un testeur ouvre le tableau de bord en arabe. Chaque mot est traduit et chacun d'eux est au mauvais endroit : les labels se collent au bord gauche de leurs champs, la colonne de navigation se trouve là où l'œil d'un lecteur arabe arrive en dernier, et les longs libellés allemands du module précédent poussent déjà un bouton hors d'un panneau à largeur fixe.

La traduction était la moitié facile. Un écran localisé a aussi un sens de lecture et une largeur qu'il ne maîtrise pas, et ce sont là deux décisions de mise en page que l'on prend en construisant le formulaire, pas des réparations que l'on fait après coup.

2. Concepts clés

Concept	Pourquoi c'est important
RightToLeft	Une propriété de contrôle aux valeurs <code>Inherit</code> , <code>No</code> et <code>Yes</code> qui demande à un contrôle d'adapter son propre texte et le sens de ses éléments internes.
RightToLeft.Inherit	Le choix normal pour un contrôle enfant : il suit son parent, si bien que tout l'écran change de sens à partir d'une seule décision prise en haut.
RightToLeftLayout	Une autre propriété, qui met en miroir la position horizontale des enfants directs d'un conteneur quand le RTL est actif, de sorte que l'agencement visuel s'inverse pendant que votre mise en page logique reste telle que conçue.
Le paramètre d'application <code>rightToLeft</code>	Une entrée de <code>Default.json</code> ; réglée sur <code>"auto"</code> , elle fait déduire par Wisej.NET le comportement RTL de la culture de la session, au lieu de tests de nom de culture éparpillés dans vos écrans.
<code>Application.RightToLeft</code>	La même décision exprimée dans le code, pour les sessions qui doivent être basculées délibérément plutôt que par détection de la culture.
Allongement du texte	Un libellé traduit a rarement la longueur de l'anglais, et c'est pourquoi les positions en pixels fixes et les largeurs de panneaux mesurées à la main cassent en premier dans une version localisée.
Limites RTL documentées	Wisej.NET documente que <code>DataGridView</code> , <code>ListView</code> et <code>PropertyGrid</code> ne prennent pas totalement en charge le RTL : un plan de tests doit donc les nommer au lieu de supposer qu'ils se mettent en miroir.

3. Le fil de la leçon

Deux propriétés, deux travaux

La première erreur que presque tout le monde commet est de traiter le droite à gauche comme un réglage d'alignement du texte. L'arabe, l'hébreu et les autres langues de droite à gauche demandent que le texte aille dans l'autre sens *et*, en général, que l'agencement de l'écran soit mis en miroir. Wisej.NET sépare ces deux travaux en deux propriétés, et savoir laquelle vous est nécessaire constitue l'essentiel de ce module.

`RightToLeft` appartient à un seul contrôle et change ce qui se passe à l'intérieur de lui : le sens de son texte et des éléments qu'il dessine pour lui-même. Elle accepte `Inherit`, `No` ou `Yes`. Laissez-la sur `Inherit` presque partout, car un enfant suit alors son parent et tout le tableau de bord de GlobalDesk répond à une seule décision prise en haut. Mettre `Yes` sur des contrôles individuels vous donne un écran dont le sens est configuré à cinquante endroits, sans que personne ne puisse dire lequel est faux.

`RightToLeftLayout`, c'est l'autre travail. Elle met en miroir la position horizontale des enfants directs d'un conteneur quand le RTL est actif : vous continuez donc à concevoir l'éditeur client dans l'ordre qui vous parle pendant que Wisej.NET inverse l'agencement visuel. Notez les mots « enfants directs » : l'activer sur le tableau de bord met en miroir les panneaux que le tableau de bord contient, et chaque panneau décide

lui-même si ses propres contrôles sont mis en miroir. C'est pour cela que le lab l'active sur le tableau de bord et sur l'éditeur client.

Laissez le paramètre d'application décider

`Default.json` porte un paramètre `rightToLeft` à côté du paramètre `culture` que vous avez configuré au module précédent. Avec la valeur `"auto"`, Wisej.NET déduit le comportement RTL de la culture sous laquelle tourne la session : ajouter l'arabe au sélecteur suffit donc à faire suivre le sens de lecture à la langue. `Application.RightToLeft` expose la même décision au code quand une session doit être pilotée délibérément.

L'alternative, c'est le motif à éviter : un test de nom de culture dans chaque écran, puis une deuxième liste de préfixes quand l'hébreu arrive, puis un écran que quelqu'un a oublié. Un seul paramètre d'application remplace tout cela, et c'est pourquoi vos contrôles peuvent rester sur `Inherit`. Le sens de lecture devient un état de session, exactement comme `Application.CurrentCulture` : un testeur qui passe à l'arabe ne doit jamais mettre en miroir l'écran d'un collègue.

Concevoir une mise en page capable de se mettre en miroir

La mise en miroir ne rend bien que si la mise en page était souple au départ. Un contrôle posé à une `Location` mesurée à la main dans un panneau de taille fixe se retrouve, une fois mis en miroir, à une position que personne n'a conçue, et un libellé 40 % plus long en allemand déborde tout simplement. Les outils sont ceux de la mise en page Wisej.NET ordinaire : le docking pour qu'une région possède un bord, l'ancrage pour qu'un contrôle grandisse avec son conteneur, `AutoSize` là où un label doit pouvoir être aussi long que sa traduction, un conteneur de mise en page là où une rangée de boutons doit pouvoir se réorganiser, et des tailles minimales raisonnables pour que rien ne s'effondre.

Il y a une exception délibérée, et le lab vous demande d'écrire le commentaire qui l'explique. Certaines chaînes ne sont pas de la prose : un code client, un numéro de série, le contenu d'un éditeur de code. Ceux-là se lisent de gauche à droite dans toutes les cultures, et les inverser transforme un identifiant correct en identifiant déroutant. C'est à cela que sert `RightToLeft.No` — à utiliser avec parcimonie, toujours avec une raison inscrite à côté.

```
private void ConfigureDirection()
{
    // Default.json carries rightToLeft "auto", so Application.RightToLeft
    // already follows the session culture. Children stay on Inherit.
    this.RightToLeft = RightToLeft.Inherit;
    this.RightToLeftLayout = true;

    // The customer code is a technical identifier, not prose:
    // it must keep reading left to right in every culture.
    txtCustomerCode.RightToLeft = RightToLeft.No;
}
```

Tester l'allongement et les contrôles mis en miroir

Aucun des deux problèmes n'est visible dans le code : le module se termine donc par une liste de tests plutôt que par une fonctionnalité. Faites passer chaque écran localisé par au moins cinq cas :

- Un texte court dans la langue neutre, c'est-à-dire l'état dans lequel tout a été conçu.
- Une langue aux traductions nettement plus longues, comme l'allemand.
- Une fenêtre de navigateur étroite, là où l'ancrage et les tailles minimales prouvent leur utilité.
- Une culture RTL comme l'arabe, en vérifiant l'agencement et pas seulement l'alignement.
- Du contenu mixte, comme un label en arabe à côté d'un code produit en caractères latins.

Un fichier de ressources pseudo-localisé rend les cas d'allongement honnêtes. Copiez les clés neutres, puis allongez chaque valeur et encadrez-la de crochets pour qu'un libellé devienne quelque chose comme `[[Save changes ~~~~]]`. Tout ce qui se lit encore en anglais courant n'a jamais été une clé de ressource, et tout ce qui est tronqué vous dit quel contrôle a besoin d'ancrage. Les thèmes Wisej.NET comprennent eux aussi un état `rightToLeft` : les icônes et le style asymétrique peuvent donc s'adapter sans second thème ; ce sont les règles de thème personnalisées et les widgets personnalisés qui peuvent encore demander une mise en miroir écrite à la main.

Les contrôles aux limites documentées

Enfin, la partie honnête. Wisej.NET documente que `DataGridView`, `ListView` et `PropertyGrid` ne prennent pas totalement en charge le droite à gauche. GlobalDesk affiche ses contacts dans une `DataGridView` : le lab vous demande donc de consigner ce que cette grille fait réellement en arabe, et non ce que vous attendiez. L'ordre des colonnes, l'alignement des en-têtes et le côté de la barre de défilement sont les points à regarder.

Ce n'est pas une raison d'éviter ces contrôles ; c'est une raison de les nommer dans le plan de tests et de les vérifier sur la version et la configuration exactes que vous livrez. Une mise en page localisée ne doit jamais tronquer un libellé traduit, pousser un contrôle hors de la zone visible, dépendre de la longueur des mots anglais, incruster une flèche directionnelle dans une image sans variante RTL, ni forcer un identifiant dans le mauvais sens de lecture — chacun de ces défauts coûte moins cher à corriger sur le premier écran que sur le centième.



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 :

- ✓ Utiliser correctement `RightToLeft` et `RightToLeftLayout` dans Wisej.NET.
- ✓ Expliquer le paramètre d'application `rightToLeft="auto"`.
- ✓ Construire une mise en page qui se met en miroir comme il faut pour les cultures RTL.

- ✓ Tester l'allongement du texte et les contrôles mis en miroir.
- ✓ Identifier les contrôles Wisej dont la prise en charge du RTL comporte des limites documentées.

Guide du lab / de l'examen

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



Lab pratique

Faites en sorte que GlobalDesk survive à une culture de droite à gauche. Ajoutez l'arabe au sélecteur de langue et mettez `rightToLeft` sur `auto` dans `Default.json`, laissez les contrôles enfants sur `RightToLeft.Inherit`, et activez `RightToLeftLayout` sur le tableau de bord et l'éditeur client pour que leurs enfants soient mis en miroir. Gardez exactement un contrôle de gauche à droite, le champ du code client, et expliquez dans un commentaire pourquoi cet identifiant ne doit pas être inversé. Reconstituez avec du docking, de l'ancrage et un conteneur de mise en page la partie de l'écran placée à des positions fixes, pour qu'une traduction longue ne puisse pas être tronquée, ajoutez un fichier de ressources pseudo-localisé dont les valeurs sont allongées et encadrées de crochets, et notez ce que fait réellement la `DataGridView` des contacts en RTL au lieu de supposer qu'elle se met en miroir. Terminez par une liste de tests de mise en page en cinq cas que vous pourrez rejouer sur n'importe quel écran.

Point de départ

GlobalDesk après le Module 4 : le tableau de bord, l'éditeur client et la grille des contacts lisent tous leurs libellés depuis les ressources, le sélecteur de langue propose l'anglais, l'allemand et une culture langue-région, et un gestionnaire `Application.CultureChanged` appelle déjà `ApplyTextResources` et `UpdateCulturePreview`. Tous les écrans sont encore construits de gauche à droite, et le panneau de navigation du tableau de bord utilise toujours les valeurs fixes de `Location` et de `Size` héritées du Module 1.

Approche suggérée

- ✓ Ajoutez une entrée arabe au sélecteur de langue, mettez `rightToLeft` sur `"auto"` à côté du paramètre `culture` existant dans `Default.json`, puis exécutez l'application et passez à l'arabe avant de changer quoi que ce soit d'autre, pour voir exactement ce que fait la détection automatique à elle seule.
- ✓ Passez `RightToLeftLayout` à `true` sur le tableau de bord et sur l'éditeur client, et vérifiez que chaque contrôle enfant est toujours sur `RightToLeft.Inherit` — résistez à l'envie de mettre `Yes` sur des contrôles individuels pour « corriger » ce que le conteneur devrait décider.
- ✓ Mettez `RightToLeft.No` sur le seul champ de code client, et écrivez le commentaire qui explique pourquoi un identifiant technique doit garder son sens de lecture alors que le texte autour de lui s'inverse.
- ✓ Remplacez les positions absolues du panneau de navigation par du docking et de l'ancrage, placez la rangée de boutons de l'éditeur dans un conteneur de mise en page pour qu'elle puisse se réagencer, et donnez `AutoSize` aux libellés et des tailles minimales raisonnables aux panneaux.
- ✓ Ajoutez un fichier de ressources pseudo-localisé dont les valeurs sont allongées et encadrées de crochets, faites-y passer les cinq cas de test, notez ce que fait réellement la `DataGridView` des contacts en arabe, et transformez toute cette passe en une liste de tests réutilisable.

Livrables attendus

- ✓ Entrée arabe dans le sélecteur de langue avec `rightToLeft` sur `auto` dans `Default.json`
- ✓ Tableau de bord et éditeur client utilisant `RightToLeftLayout`, leurs enfants restant sur `Inherit`
- ✓ Un contrôle délibérément laissé de gauche à droite, avec un commentaire expliquant l'exception
- ✓ Mise en page reconstruite avec docking, ancrage et un conteneur de mise en page pour qu'une traduction longue ne soit jamais tronquée
- ✓ Liste de tests RTL couvrant un texte court, un texte long, une fenêtre étroite, une culture RTL et la limitation de la `DataGridView`

Questions de révision

- ✓ Un collègue met `RightToLeft.Yes` sur vingt contrôles et l'écran arabe paraît correct, mais l'écran allemand n'a pas changé. Qu'a-t-il réellement configuré, et quelle propriété unique aurait fait le travail pour tout le conteneur ?
- ✓ Votre passe pseudo-localisée montre trois libellés encore en anglais courant. Rien n'a été tronqué et rien n'a levé d'exception. Que cela vous apprend-il sur ces trois libellés, et le travail de quel module précédent comporte-t-il une lacune ?
- ✓ La grille des contacts ne se met pas en miroir comme le reste du tableau de bord. Laquelle de vos options — l'accepter et la documenter, remplacer le contrôle, ou écrire la mise en miroir à la main — pouvez-vous défendre devant un relecteur, et que devriez-vous vérifier avant de promettre l'une d'elles ?



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.