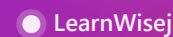


Ressources d'application, ResourceManager et textes système de Wisej

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



Guide de la leçon

La couche de ressources partagées : des fichiers .resx neutres et spécifiques à une culture qui portent des clés identiques, la recherche par `Wisej.Resources.ResourceManager` à partir du nom de base et de l'assembly, des espaces réservés mis en forme avec `string.Format` et la culture active, des services qui renvoient des clés plutôt que des phrases anglaises, et `Resources-[LANG].resx` pour les libellés système de Wisej. 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 couche de ressources partagées : des fichiers .resx neutres et spécifiques à une culture qui portent des clés identiques, la recherche par `Wisej.Resources.ResourceManager` à partir du nom de base et de l'assembly, des espaces réservés mis en forme avec `string.Format` et la culture active, des services qui renvoient des clés plutôt que des phrases anglaises, et `Resources-[LANG].resx` pour les libellés système de Wisej.

1. Le problème en production

Le Module 2 a laissé l'éditeur client de GlobalDesk très convaincant en allemand : chaque libellé que possède le designer change avec la culture et la mise en page tient. Puis l'utilisateur appuie sur Save et lit *Customer saved.* en anglais, parce que cette phrase n'est la propriété d'aucun contrôle — c'est un littéral dans un gestionnaire de clic, à côté de *This field is required.* et de la ligne d'état qui compte les contacts chargés.

La localisation dans le designer ne peut jamais atteindre ces chaînes : elles sont générées par le code, partagées par plusieurs écrans, ou produites par un service qui n'a aucune idée de l'existence d'une interface utilisateur. Ce module donne à GlobalDesk une deuxième couche de ressources pour exactement ce texte, puis descend d'un cran : les libellés que Wisej.NET dessine lui-même sur une `MessageBox` ou un calendrier.

2. Concepts clés

Concept	Pourquoi c'est important
fichier de ressources d'application partagé	<code>Resources/Strings.resx</code> contient le texte visible par l'utilisateur qu'aucun écran conçu ne possède en propre : confirmations d'enregistrement, messages de validation, lignes d'état, titres de rapports et résultats de services.
fichier frère spécifique à une culture	<code>Strings.de.resx</code> et <code>Strings.it.resx</code> se placent à côté du fichier neutre et portent les mêmes clés ; un traducteur change les valeurs, jamais les noms de clés.
clé sémantique stable	Un nom comme <code>CustomerEditor.Saved</code> décrit le sens du texte : reformuler la phrase anglaise n'oblige donc jamais à modifier le code ni ne casse les traductions.
<code>Wisej.Resources.ResourceManager</code>	L'objet de recherche qui suit le modèle de ressources .NET : à partir d'un nom de base et d'un assembly, il trouve la valeur pour la culture active et retombe progressivement vers le fichier neutre.
marqueur de clé manquante	Une classe utilitaire qui renvoie <code>[Key]</code> quand une recherche échoue transforme un défaut invisible en quelque chose que l'on voit et que l'on peut rechercher, au lieu d'un label vide.
chaîne de format avec espaces réservés	La phrase entière avec ses <code>{0}</code> et <code>{1}</code> vit dans la ressource et <code>string.Format</code> la remplit, parce que les langues ordonnent le nom, la date et le verbe différemment.
<code>Resources-[LANG].resx</code>	Un fichier de ressources système Wisej.NET distinct, à la racine du projet ou dans un dossier <code>/Resources</code> , qui redéfinit ou ajoute des libellés du framework comme <code>\$Ok</code> , <code>\$Cancel</code> et <code>\$Offline</code> .

3. Le fil de la leçon

Deux fichiers, un seul jeu de clés

Commencez par le fichier neutre. Dans GlobalDesk, c'est `Resources/Strings.resx`, et il porte déjà les clés introduites par le Module 1 : `Navigation.Customers`, `CustomerEditor.Save`, `CustomerEditor.Cancel`, `CustomerEditor.Saved`, `Validation.Required`. Le fichier neutre est celui sur lequel .NET retombe quand rien de mieux n'existe : il n'a donc jamais le droit d'être incomplet. Une clé qui y manque est un défaut du développeur, pas un manque de traduction.

Un fichier traduit porte le même nom de fichier avec un nom de culture avant l'extension :

`Strings.de.resx` pour l'allemand, `Strings.it.resx` pour l'italien. Seule la colonne des **valeurs** diffère. Renommez `CustomerEditor.Saved` dans le fichier allemand et il cesse de répondre à la recherche : l'écran retombe silencieusement sur l'anglais. La liste des clés est un contrat : ajouter une clé, c'est l'ajouter partout.

Un seul fichier allemand sert indifféremment `de`, `de-DE` et `de-AT` : .NET part de la culture la plus spécifique vers la ressource neutre et prend la première correspondance. N'ajoutez `Strings.de-AT.resx` que si la formulation autrichienne diffère réellement.

Rechercher du texte via ResourceManager

Wisej.NET fournit `Wisej.Resources.ResourceManager`, qui suit le modèle de ressources .NET et sait localiser les ressources spécifiques à une culture. Vous en construisez un à partir d'un nom de base et de l'assembly qui contient les ressources ; le nom de base, c'est l'espace de noms du projet plus le dossier et le nom du fichier, soit `GlobalDesk.Resources.Strings` pour GlobalDesk. Se tromper là-dessus est l'échec le plus courant chez les débutants, et cela se manifeste par toutes les clés qui reviennent manquantes d'un coup.

Encapsulez le manager dans une petite classe utilitaire statique, pour que le reste de l'application n'y touche jamais directement et qu'une seule méthode décide de ce que fait une clé manquante :

```
using Wisej.Resources;

public static class Texts
{
    private static readonly ResourceManager RS =
        new ResourceManager("GlobalDesk.Resources.Strings", typeof(Texts).Assembly);

    public static string Get(string key)
        ⇒ RS.GetString(key) ?? $"[{key}]";
}
```

Le marqueur entre crochets est un bon comportement par défaut en développement : un libellé vide ressemble à un bug de mise en page, alors que `[CustomerEditor.Saved]` vous dit exactement quelle clé ajouter et où. En production, vous préférerez peut-être journaliser la clé manquante, ce que le Module 7 transforme en politique. Ce qu'il ne faut jamais faire, c'est laisser une ressource manquante devenir du texte vide et silencieux.

Ressources du designer, ressources partagées, et où va chaque chaîne

Deux mécanismes de ressources cohabitent désormais dans le même projet, et ils répondent à des questions différentes. Les ressources du designer appartiennent à une seule surface conçue, sont écrites dans les fichiers `.resx` propres à ce conteneur, et sont appliquées à la création de ses contrôles. Les ressources d'application partagées appartiennent à l'application et sont lues à la demande via `Texts.Get` : elles sont donc à jour au moment même où vous les appelez.

Le test pratique, c'est la propriété. Si un libellé existe parce que vous avez déposé un `Button` sur l'éditeur client, il appartient au designer. Si le texte apparaît dans une `MessageBox`, dans une ligne d'état, dans un en-tête de colonne de grille que vous construisez dans le code, ou sur deux écrans différents, il relève de `Strings.resx`. Dupliquer une phrase dans trois fichiers de designer, c'est ainsi que naissent les applications à moitié traduites : la troisième copie est toujours celle que personne ne met à jour.

Le même raisonnement va jusque dans le métier. Un service d'enregistrement ne doit pas renvoyer la phrase *Customer saved.* ; il doit renvoyer un résultat stable comme la clé `CustomerEditor.Saved` ou un

code de résultat que l'interface convertit en clé. La logique métier reste alors testable, aucun anglais ne fuit hors de la couche de services, et l'écran demeure le seul endroit qui sache s'adresser à un utilisateur.

Les libellés de Wisej : localisation système et Resources-[LANG].resx

Tous les mots à l'écran ne sont pas les vôtres. Wisej.NET localise ses propres ressources de framework — boutons de `MessageBox`, libellés de calendrier, et comportement dépendant de la culture des éditeurs numériques, de date et de montant — et fournit des traductions système intégrées pour l'anglais, l'allemand, le français, l'italien, le turc, l'espagnol et le portugais. Exécutez GlobalDesk en allemand et les boutons Yes/No changent d'eux-mêmes.

Pour une langue que Wisej.NET ne couvre pas, ou pour une formulation que votre produit impose, Wisej.NET prend en charge un fichier `Resources-[LANG].resx` à la racine du projet ou dans un dossier `/Resources`. Ses clés sont celles du framework et commencent par un signe dollar : `$Ok`, `$Cancel`, `$Yes`, `$No`, `$Retry`, les libellés de navigation du calendrier et `$Offline`. Gardez les deux bien distincts : `Strings.de.resx` est ce que dit GlobalDesk, `Resources-de.resx` est ce que dit Wisej.NET.

Messages, validation et images sous la culture active

Les phrases qui mentionnent des données ont besoin d'espaces réservés, et c'est là que les débutants perdent l'ordre des mots. Stockez la phrase complète dans la ressource — `Customer.LastOrder` contient `{0} last ordered on {1}`. — et laissez le code fournir les valeurs :

```
string.Format(Application.CurrentCulture, Texts.Get("Customer.LastOrder"), customerName, lastOrderDate)
```

Passer la culture met aussi la date en forme comme la session l'attend. Ne concaténez jamais `"Customer " + name + " saved"` : un traducteur ne peut pas déplacer un fragment que le code a collé.

Les espaces réservés n'ont aucun sens pour quelqu'un qui ne voit que la colonne des valeurs : servez-vous du champ commentaire de l'éditeur `.resx` pour dire ce que sont `{0}` et `{1}`. Les messages de validation méritent le même soin : `Validation.Required` et `Validation.Email` sont réutilisés par plusieurs écrans, et c'est pourquoi ils relèvent du fichier partagé plutôt que de chaque formulaire.

Un fichier `.resx` peut contenir autre chose que des chaînes — les images et d'autres données sont des entrées légitimes, et la localisation dans le designer peut échanger une image selon la langue. Faites-le en connaissance de cause, toutefois. Une image qui ne diffère que parce qu'elle a du texte incrusté est un piège de maintenance ; laissez ce texte sous forme de vrai texte dans l'interface, où `Texts.Get` le traduit sans qu'aucun graphiste n'ait à redessiner quoi que ce soit.



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 :

- ✓ Créer des fichiers de ressources .resx neutres et spécifiques à une culture.
- ✓ Récupérer des valeurs via ResourceManager en utilisant des clés stables.
- ✓ Organiser les chaînes réutilisables séparément des ressources du designer.
- ✓ Expliquer la localisation système intégrée de Wisej.NET et les fichiers Resources-[LANG].resx personnalisés.
- ✓ Utiliser des ressources adaptées à la culture pour les messages, la validation et les images.

Guide du lab / de l'examen

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



Lab pratique

Enrichissez les ressources partagées de GlobalDesk. Ajoutez `Strings.de.resx` à côté de `Resources/Strings.resx` avec exactement les mêmes clés et des valeurs traduites, ajoutez une entrée `Customer.LastOrder` qui contient deux espaces réservés, et composez la phrase avec `string.Format` et `Application.CurrentCulture` au lieu de concaténer des fragments. Faites passer par la classe utilitaire `Texts` chaque message affiché par l'éditeur client, y compris la confirmation d'enregistrement et les deux messages de validation, et faites en sorte que le service d'enregistrement renvoie une clé de résultat stable comme `CustomerEditor.Saved` plutôt qu'une phrase anglaise que l'interface affiche telle quelle. Ajoutez ensuite un fichier `Resources-de.resx` qui redéfinit un libellé système de Wisej, démontrez-le dans une `MessageBox`, et rédigez une courte note sur la différence entre le texte de votre produit et le texte système de Wisej.

Point de départ

Vous ouvrez GlobalDesk tel que le Module 2 l'a laissé : le tableau de bord lit ses libellés depuis `Resources/Strings.resx` via la classe utilitaire `Texts`, et le contrôle utilisateur `CustomerEditor` est localisé dans le designer avec une variante allemande. La confirmation d'enregistrement, les deux messages de validation et la ligne de dernière commande sont encore des littéraux anglais dans le code, et il n'existe pas encore de fichier de ressources partagé traduit.

Approche suggérée

- ✓ Ajoutez `Resources/Strings.de.resx` à côté du fichier neutre et recopiez-y la liste des clés sans la modifier, en ne traduisant que les valeurs ; exécutez l'application en `de-DE` et vérifiez que les libellés du tableau de bord changent avant de toucher au moindre code.
- ✓ Ajoutez une entrée `Customer.LastOrder` dont la valeur est une phrase complète avec `{0}` et `{1}`, remplissez le champ commentaire pour qu'un traducteur sache que le premier est le nom du client et le second la date de commande, et composez la ligne avec `string.Format(Application.CurrentCulture, Texts.Get("Customer.LastOrder"), customerName, lastOrderDate)`.
- ✓ Remplacez les littéraux de `CustomerEditor` par des appels à `Texts.Get` pour `CustomerEditor.Saved`, `Validation.Required` et `Validation.Email`, et supprimez toute concaténation de chaînes que vous croiserez au passage.
- ✓ Modifiez le service d'enregistrement pour qu'il renvoie un résultat stable — la clé `CustomerEditor.Saved` ou un code de résultat — et laissez l'éditeur convertir ce résultat en texte via `Texts.Get`, pour qu'aucune phrase anglaise ne subsiste dans la couche de services.
- ✓ Ajoutez `Resources-de.resx` à la racine du projet ou dans le dossier `/Resources`, redéfinissez un libellé système de Wisej.NET comme `$Ok` ou `$Cancel`, puis affichez une `MessageBox` en allemand et comparez ses boutons avec l'exécution en anglais.

Livrables attendus

- ✓ Strings.de.resx portant les mêmes clés que le fichier neutre, avec uniquement des valeurs traduites
- ✓ Ressource Customer.LastOrder composée avec string.Format et la culture active, jamais par concaténation
- ✓ Service d'enregistrement qui renvoie une clé de ressource ou un code de résultat au lieu d'une phrase anglaise
- ✓ Resources-de.resx qui redéfinit au moins un libellé système de Wisej, montré dans une MessageBox
- ✓ Note de lab distinguant le texte du produit, le texte système de Wisej et le texte de diagnostic

Questions de révision

- ✓ Supprimez une clé de `Strings.de.resx` et exécutez en allemand. Décrivez ce que voit l'utilisateur, pourquoi rien ne plante, et comment vous vous en apercevriez dans une vraie application de deux cents clés.
- ✓ Votre service d'enregistrement renvoie actuellement `CustomerEditor.Saved`. Qu'est-ce qui casserait dans les tests unitaires, et dans l'écran allemand, s'il renvoyait la phrase elle-même ?
- ✓ Un collègue ajoute le mot "Cancel" à `Resources-de.resx` sous une clé de son invention. Expliquez pourquoi le bouton de la `MessageBox` ne change pas, et où cette chaîne avait réellement sa place.



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.