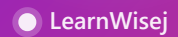


# Flux de traduction : ResX Resource Manager et autres outils

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



## Guide de la leçon

Traiter la traduction comme un travail d'équipe : la matrice de ResX Resource Manager sur tous les .resx de la solution, l'ajout d'une colonne de langue et le filtre sur les entrées non traduites, les commentaires de ressource qui donnent du contexte aux traducteurs, les chaînes invariantes, la traduction automatique comme simple brouillon, l'export et l'import avec une plateforme capable de lire le RESX, et la relecture des diffs de ressources comme on relit du code. 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

Traiter la traduction comme un travail d'équipe : la matrice de ResX Resource Manager sur tous les .resx de la solution, l'ajout d'une colonne de langue et le filtre sur les entrées non traduites, les commentaires de ressource qui donnent du contexte aux traducteurs, les chaînes invariantes, la traduction automatique comme simple brouillon, l'export et l'import avec une plateforme capable de lire le RESX, et la relecture des diffs de ressources comme on relit du code.

## 1. Le problème en production

GlobalDesk porte désormais un `Resources/Strings.resx` neutre, un `Strings.de.resx` allemand, des ressources de designer sur l'éditeur client et un `Resources-de.resx` pour les libellés système de Wisej. Ajouter l'italien avec l'éditeur de Visual Studio, c'est ouvrir ces fichiers un onglet à la fois et comparer les listes de clés à l'œil nu, et la personne qui parle italien n'est en général pas celle qui sait ouvrir la solution. Ce module transforme la traduction en un flux de travail doté d'une liste de manques visible, d'une transmission et d'une étape de relecture.

## 2. Concepts clés

| Concept                          | Pourquoi c'est important                                                                                                                                                                                                        |
|----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Vue matricielle                  | Les clés en lignes et les langues en colonnes, si bien qu'une valeur italienne manquante et un terme allemand qui a dérivé se voient d'un seul coup d'œil.                                                                      |
| ResX Resource Manager            | Un outil de localisation .NET open source, à la fois extension Visual Studio et application autonome, qui présente les ressources RESX d'une solution dans une grille centrale unique.                                          |
| Colonne de langue                | Ajouter une langue cible crée le fichier spécifique à la culture à côté du fichier neutre et vous donne une colonne vide à remplir ; les clés, elles, ne changent jamais.                                                       |
| Filtre des entrées non traduites | Ne liste que les entrées sans valeur dans la langue sélectionnée, si bien que « est-ce terminé ? » devient un nombre au lieu d'une impression.                                                                                  |
| Commentaire de ressource         | Une note stockée à côté de la valeur qui indique au traducteur si <b>Open</b> est un verbe, un adjectif, un état de ticket ou une commande.                                                                                     |
| Chaîne invariante                | Un marqueur sur les entrées qui ne doivent jamais être traduites, comme le nom du produit et les identifiants techniques.                                                                                                       |
| Aller-retour export / import     | Des colonnes envoyées à un traducteur ou à une plateforme capable de lire le RESX et renvoyées sous forme de fichiers <b>.resx</b> spécifiques à une culture, que la compilation traite comme n'importe quelle autre ressource. |

## 3. Le fil de la leçon

### Une seule grille au lieu de six fichiers

L'éditeur de ressources de Visual Studio convient parfaitement pour un seul fichier. Il cesse de convenir dès que la question devient comparative : chaque clé du fichier neutre a-t-elle une valeur allemande ? Un outil de traduction y répond en disposant les ressources en matrice, les clés en lignes et chaque langue dans sa propre colonne. ResX Resource Manager est le choix habituel : un outil .NET open source livré comme extension Visual Studio, qui analyse la solution active et rassemble ses ressources RESX dans une grille centrale unique, et aussi comme application autonome.

Il convient à Wisej.NET parce qu'il n'y a rien à adapter. Wisej.NET conserve son texte localisé dans des fichiers RESX .NET standard : un outil RESX générique lit donc GlobalDesk comme il lit n'importe quel autre projet .NET ; l'exemple StockViewer de Wisej mentionne d'ailleurs une localisation faite avec le designer conjointement avec ResX Manager. L'ensemble des fonctionnalités documentées, c'est le flux de travail lui-même : colonnes de langues côte à côte, entrées non traduites mises en évidence, traduction dans la grille, filtrage et tri, traduction automatique, import et export, détection des références de code, chaînes invariantes, instantanés, scripting et correction orthographique. La détection des références de code fait rapidement ses preuves : savoir où une clé est utilisée est souvent le seul moyen de décider de ce que signifie un libellé d'un seul mot.

Ce que l'outil n'est pas, c'est une partie de votre application. Il édite des fichiers ; à l'exécution, GlobalDesk effectue toujours une recherche de ressources .NET ordinaire via **Wisej.Resources.ResourceManager** ,

exactement comme au Module 3. Ne laissez jamais un outil de localisation devenir une dépendance d'exécution.

### Ajouter l'italien et transformer « non traduit » en un nombre

Ajouter une langue crée le fichier spécifique à la culture à côté du fichier neutre, `Strings.it.resx` auprès de `Strings.resx`, et vous donne une colonne nouvelle et vide. Rien n'est re-clé : la ressource neutre reste la source de vérité unique quant aux clés qui existent. Le filtre des entrées non traduites transforme alors une question vague en une liste que l'on peut traiter et transmettre.

L'ordre de travail sûr mérite d'être noté sur une fiche. Gardez `Strings.resx` complet et relu dans le contrôle de code source, ouvrez ResX Resource Manager, ajoutez la langue cible, traduisez ou importez les valeurs en préservant les clés, filtrez les entrées vides et éliminez-les, compilez et exécutez dans la culture cible, relisez le diff avant de valider — car selon ses réglages l'outil peut enregistrer vos modifications immédiatement — et enfin testez l'interface utilisateur, et pas seulement la grille.

Une grille complète avec un bouton tronqué reste un bug, et les libellés italiens sont généralement plus longs que les anglais. Le piège inverse est plus discret : une clé sans valeur italienne n'explose pas, elle retombe sur la valeur neutre, si bien que l'écran a l'air terminé alors que le manque ne vit que dans la grille.

### La traduction automatique est un brouillon, et le diff est la relecture

La traduction automatique est vraiment utile pour une première couverture. C'est un brouillon. Relisez la terminologie, les espaces réservés, la ponctuation, les accès rapides, les noms de produits et le contexte avant de livrer quoi que ce soit, car une traduction peut être techniquement correcte et rester le mauvais choix de mots dans une interface. L'entrée `Customer.LastOrder` du Module 3 rend le risque concret : un brouillon qui perd ou réordonne ses deux espaces réservés transforme un appel de mise en forme en échec.

Le remède à l'ambiguïté, c'est le contexte, et le contexte vit dans le commentaire de ressource, qui voyage avec l'entrée à chaque aller-retour :

```
<data name="CustomerEditor.Open" xml:space="preserve">
  <value>Open</value>
  <comment>Button caption, verb: opens the selected customer.</comment>
</data>
<data name="App.ProductName" xml:space="preserve">
  <value>GlobalDesk</value>
  <comment>Product name, invariant: never translate.</comment>
</data>
```

Traitez ces fichiers comme du code : relisez les diffs, gardez les clés stables, évitez les renommages en masse pendant qu'un cycle est en cours, et ne laissez jamais un brouillon automatique écraser une traduction humaine approuvée sans relecture. Comme l'outil peut écrire au fil de votre saisie, le diff du contrôle de version est le filet de sécurité.

## Transmettre les chaînes à quelqu'un qui n'ouvre jamais Visual Studio

L'export et l'import existent exactement pour cela. Vous exportez la colonne neutre avec la colonne cible, le traducteur travaille dans ce fichier, et le réimport remet les lignes renvoyées en face de leurs clés : les clés et les espaces réservés survivent donc à une personne qui n'a jamais vu un `.resx`. Ce qui revient doit porter le même jeu de clés.

Quand les langues ou les relecteurs se multiplient, la même mécanique se déplace vers un système cloud de gestion de traduction. Parmi les plateformes qui prennent en charge le `.resx`, on trouve Crowdin, Lokalise, Phrase et POEditor, et l'aller-retour reste identique : téléversez ou synchronisez le RESX neutre, traduisez et relisez dans la plateforme, exportez les fichiers `.resx` spécifiques à une culture, validez-les, puis compilez, exécutez et vérifiez dans la vraie interface. Ce que vous achetez, c'est de la gouvernance : mémoire de traduction, rôles de relecture et terminologie centralisée.

## Choisir entre l'éditeur du designer, ResXManager et une plateforme cloud

Le guide pratique est court, et il parle de l'équipe plutôt que de l'outil :

- Une développeuse seule, deux langues : l'éditeur RESX de Visual Studio peut déjà suffire.
- Une équipe de développeurs, plusieurs langues : ResX Resource Manager, un flux efficace au sein de l'IDE.
- Des traducteurs externes : l'export et l'import, ou une plateforme prenant en charge le RESX.
- De nombreux produits, langues et relecteurs : un TMS cloud est plus facile à gouverner.

Quel que soit votre choix, le contrat du projet ne bouge pas : des fichiers `.resx` spécifiques à une culture à côté du fichier neutre, avec des clés identiques, lus par la recherche de ressources standard. Prenez l'option la plus légère qui rende les entrées non traduites visibles et le diff relisible, et ne montez d'un cran que lorsque l'aller-retour devient le goulot d'étranglement.



### Pièges courants

Livrer un brouillon automatique parce que la grille est passée au vert. Perdre un espace réservé dans une valeur traduite. Renommer des clés en plein cycle. Laisser l'outil enregistrer des modifications que vous n'avez jamais lues dans un diff. Traduire le nom du produit parce que personne ne l'a marqué invariant. Déclarer le travail terminé sans avoir exécuté l'application dans la culture cible.



### 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 ; `Ressources-[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 ResX Resource Manager pour relire plusieurs langues RESX côte à côte.
- ✓ Ajouter une langue et repérer les entrées non traduites.
- ✓ Expliquer l'usage sûr de la traduction automatique et du contrôle de code source.
- ✓ Exporter/importer le contenu de traduction pour des traducteurs externes ou un TMS.
- ✓ Comparer les flux de localisation limités à Visual Studio, ceux de ResXManager et ceux du cloud.

## Guide du lab / de l'examen

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

### Lab pratique

Menez un vrai cycle de traduction sur GlobalDesk. Ouvrez la solution dans ResX Resource Manager, ajoutez l'italien comme nouvelle colonne de langue, et servez-vous du filtre des entrées non traduites pour lister toutes les clés encore sans valeur. Ajoutez un commentaire aux trois clés dont le sens est ambigu hors contexte, en commençant par le libellé d'un seul mot Open, puis traduisez ou faites pré-traduire automatiquement les valeurs italiennes et marquez le nom du produit et les deux identifiants techniques comme invariants pour que personne ne les traduise. Exportez la colonne neutre et la colonne italienne, réimportez-les, et relisez le diff .resx obtenu avant de le valider comme une seule modification relisible. Terminez par une note d'une page comparant l'éditeur de Visual Studio, ResX Resource Manager et une plateforme cloud pour une équipe dont les traducteurs ne sont pas développeurs.

## Point de départ

GlobalDesk après le Module 5 : `Resources/Strings.resx` et `Strings.de.resx` portent les mêmes clés, l'éditeur client possède des ressources de designer allemandes, `Resources-de.resx` redéfinit un libellé système de Wisej, le sélecteur de langue propose déjà l'anglais, l'allemand et l'arabe, et la mise en page survit aux chaînes longues. Il n'y a d'italien nulle part, aucun commentaire sur la moindre entrée, et aucun moyen de répondre à « combien en reste-t-il ? » sinon en faisant défiler deux fichiers côte à côte.

## Approche suggérée

- ✓ Avant d'ajouter quoi que ce soit, ouvrez `Strings.resx` et `Strings.de.resx` dans l'éditeur de Visual Studio et essayez de trouver à l'œil nu les clés qui manquent au fichier allemand. Chronométrez-vous ; ce nombre est l'argument qui justifie le reste du lab.
- ✓ Ouvrez la solution dans ResX Resource Manager et ajoutez l'italien. Vérifiez que `Strings.it.resx` apparaît à côté du fichier neutre, que la liste des clés est intacte, et que la nouvelle colonne est vide. Activez le filtre des entrées non traduites et notez le compte.
- ✓ Commentez avant de traduire. Donnez à `CustomerEditor.Open` un commentaire indiquant s'il s'agit d'un verbe ou d'un état de ticket, puis faites de même pour les deux autres clés dont le sens n'est pas évident à partir de la seule valeur, et marquez le nom du produit et les deux identifiants techniques comme invariants.
- ✓ Remplissez la colonne italienne, en vous aidant si vous le souhaitez de la traduction automatique pour une première couverture, puis relisez chaque valeur : terminologie, ponctuation, et surtout les espaces réservés de `Customer.LastOrder`. Relancez le filtre des entrées non traduites jusqu'à ce qu'il ne renvoie plus rien.
- ✓ Exportez la colonne neutre et la colonne italienne, réimportez le même fichier, et lisez le `git diff` de `Strings.it.resx` ligne par ligne avant de valider le cycle comme une seule modification. Exécutez ensuite GlobalDesk en `it-IT` et vérifiez les écrans, pas la grille.

## Livrables attendus

- ✓ Strings.it.resx produit avec ResX Resource Manager, sans aucune clé non traduite
- ✓ Commentaires de ressource sur au moins trois clés ambiguës pour donner le contexte au traducteur
- ✓ Marqueurs d'invariance sur le nom du produit et sur les identifiants qui ne doivent pas être traduits
- ✓ Export, réimport et diff .resx relu, validés comme une seule modification
- ✓ Note de flux de travail comparant l'éditeur de Visual Studio, ResX Resource Manager et un TMS cloud

## Questions de révision

- ✓ Votre colonne italienne est pleine et l'application semble correcte en `it-IT`, mais un collègue affirme que deux clés n'ont jamais été traduites. Comment les deux affirmations peuvent-elles être vraies en même temps, et quelle fonctionnalité de l'outil tranche le débat ?
- ✓ Un brouillon automatique italien a renvoyé `Ultimo ordine: {1} il {0}` pour une clé dont la valeur neutre est `Last order: {0} on {1}`. Qu'est-ce qui casse, à quel moment, et qu'est-ce qui l'aurait détecté avant la validation ?
- ✓ Votre équipe gère cinq langues, dont deux traduites par une agence qui n'a jamais ouvert Visual Studio. Lequel des trois flux de travail adopteriez-vous, et que faudrait-il changer avant de passer au cran supérieur ?



### 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.