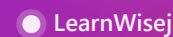


Risorse dell'applicazione, ResourceManager e i test di sistema di Wisej

Guida della lezione · Guida al lab / esame



Guida della lezione

Il livello delle risorse condivise: file .resx neutri e specifici per cultura che portano chiavi identiche, la ricerca di `Wisej.Resources.ResourceManager` per nome base e assembly, segnaposto formattati con `string.Format` e la cultura attiva, servizi che restituiscono chiavi invece di frasi inglesi, e `Resources-[LANG].resx` per le etichette di sistema di Wisej. Che cosa significa per uno sviluppatore che conosce le basi di C# e i controlli Wisej.NET e deve ora rilasciare l'applicazione in più di una lingua — e come affrontare questo modulo.



Focus di produzione

Il livello delle risorse condivise: file .resx neutri e specifici per cultura che portano chiavi identiche, la ricerca di `Wisej.Resources.ResourceManager` per nome base e assembly, segnaposto formattati con `string.Format` e la cultura attiva, servizi che restituiscono chiavi invece di frasi inglesi, e `Resources-[LANG].resx` per le etichette di sistema di Wisej.

1. Il problema in produzione

Il Modulo 2 ha lasciato l'editor cliente di GlobalDesk convincente in tedesco: ogni etichetta di cui è proprietario il designer cambia con la cultura e il layout regge. Poi l'utente preme Save e legge *Customer saved.* in inglese, perché quella frase non è una proprietà di nessun controllo — è un letterale di stringa dentro un gestore di click, accanto a *This field is required.* e alla riga di stato che conta i contatti caricati.

La localizzazione nel designer non può raggiungere quelle stringhe: sono generate dal codice, condivise da più schermate, oppure prodotte da un servizio che non sa nemmeno che esista un'interfaccia utente.

Questo modulo dà a GlobalDesk un secondo livello di risorse esattamente per quel testo, poi scende di un livello ancora: le etichette che Wisej.NET stesso disegna su una `MessageBox` o su un calendario.

2. Concetti chiave

Concetto	Perché è importante
file di risorse condiviso dell'applicazione	<code>Resources/Strings.resx</code> contiene il testo visibile all'utente che non appartiene a nessuna singola schermata disegnata: conferme di salvataggio, messaggi di validazione, righe di stato, intestazioni di report e risultati dei servizi.
file gemello specifico per cultura	<code>Strings.de.resx</code> e <code>Strings.it.resx</code> stanno accanto al file neutro e portano le stesse chiavi ; un traduttore cambia i valori, mai i nomi delle chiavi.
chiave semantica stabile	Un nome come <code>CustomerEditor.Saved</code> descrive il significato del testo, così riformulare la frase inglese non costringe mai a modificare il codice né rompe le traduzioni.
<code>Wisej.Resources.ResourceManager</code>	L'oggetto di ricerca che segue il modello delle risorse di .NET: dato un nome base e un assembly trova il valore per la cultura attiva e ricade verso il file neutro.
marcatore di chiave mancante	Un helper che restituisce <code>[Key]</code> quando una ricerca fallisce trasforma un difetto invisibile in qualcosa che puoi vedere e cercare, invece che in una label vuota.
stringa di formato con segnaposto	L'intera frase con i suoi <code>{0}</code> e <code>{1}</code> vive nella risorsa e <code>string.Format</code> la riempie, perché le lingue ordinano il nome, la data e il verbo in modo diverso.
<code>Resources-[LANG].resx</code>	Un file di risorse di sistema di Wisej.NET separato, nella radice del progetto o in una cartella <code>/Resources</code> , che sovrascrive o aggiunge etichette del framework come <code>\$Ok</code> , <code>\$Cancel</code> e <code>\$Offline</code> .

3. Il filo della lezione

Due file, un solo insieme di chiavi

Parti dal file neutro. In GlobalDesk è `Resources/Strings.resx`, e porta già le chiavi introdotte dal Modulo 1: `Navigation.Customers`, `CustomerEditor.Save`, `CustomerEditor.Cancel`, `CustomerEditor.Saved`, `Validation.Required`. Il file neutro è quello su cui .NET ricade quando non esiste niente di meglio, quindi non gli è mai permesso di essere incompleto. Una chiave che manca qui è un difetto dello sviluppatore, non una lacuna di traduzione.

Un file tradotto è lo stesso nome di file con un nome di cultura prima dell'estensione: `Strings.de.resx` per il tedesco, `Strings.it.resx` per l'italiano. A cambiare è solo la colonna dei **valori**. Rinomina `CustomerEditor.Saved` nel file tedesco e smette di rispondere alla ricerca, così la schermata ricade silenziosamente sull'inglese. L'elenco delle chiavi è un contratto: aggiungere una chiave significa aggiungerla ovunque.

Un solo file tedesco serve allo stesso modo `de`, `de-DE` e `de-AT`: .NET risale dalla cultura più specifica verso la risorsa neutra e prende la prima corrispondenza. Aggiungi `Strings.de-AT.resx` solo quando la formulazione austriaca è davvero diversa.

Cercare il testo attraverso ResourceManager

Wisej.NET fornisce `Wisej.Resources.ResourceManager`, che segue il modello delle risorse di .NET e sa come individuare le risorse specifiche per cultura. Ne costruisci uno a partire da un nome base e dall'assembly che contiene le risorse; il nome base è il namespace del progetto più la cartella e il nome del file, che per GlobalDesk è `GlobalDesk.Resources.Strings`. Sbagliarlo è l'errore più comune di chi inizia, e si manifesta con tutte le chiavi che tornano mancanti in una volta sola.

Avvolgi il manager in un piccolo helper statico, così il resto dell'applicazione non lo tocca mai direttamente e un unico metodo decide che cosa succede quando una chiave manca:

```
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}]";
}
```

Il marcatore tra parentesi quadre è un buon comportamento predefinito in sviluppo: un'etichetta vuota sembra un bug di layout, mentre `[CustomerEditor.Saved]` ti dice esattamente quale chiave aggiungere e dove. In produzione puoi preferire di registrare a log la chiave mancante, cosa che il Modulo 7 trasforma in una politica. Quello che non devi mai fare è lasciare che una risorsa mancante diventi testo vuoto e silenzioso.

Risorse del designer, risorse condivise e dove appartiene ogni stringa

Ora nello stesso progetto convivono due meccanismi di risorse, e rispondono a domande diverse. Le risorse del designer appartengono a una singola superficie disegnata, vengono scritte nei file `.resx` di quel contenitore e vengono applicate quando i suoi controlli vengono creati. Le risorse condivise dell'applicazione appartengono all'applicazione e vengono lette su richiesta attraverso `Texts.Get`, quindi sono aggiornate al momento esatto in cui le chiami.

La prova pratica è la proprietà. Se un'etichetta esiste perché hai trascinato un `Button` sull'editor cliente, ne è proprietario il designer. Se il testo compare in una `MessageBox`, in una riga di stato, nell'intestazione di una colonna di griglia che costruisci nel codice, o su due schermate diverse, appartiene a `Strings.resx`. Duplicare una stessa frase in tre file del designer è il modo in cui nascono le applicazioni tradotte a metà: la terza copia è sempre quella che nessuno aggiorna.

Lo stesso ragionamento arriva fino al dominio. Un servizio di salvataggio non dovrebbe restituire la frase *Customer saved.*; dovrebbe restituire un risultato stabile come la chiave `CustomerEditor.Saved` o un

codice di risultato che l'interfaccia mappa su una chiave. La logica di business resta così testabile, nessun inglese trapela dal livello dei servizi e la schermata resta l'unico posto che sa come parlare a un utente.

Le etichette di Wisej: localizzazione di sistema e `Resources-[LANG].resx`

Non tutte le parole sullo schermo sono tue. Wisej.NET localizza le proprie risorse del framework — i pulsanti delle `MessageBox`, le etichette del calendario e il comportamento sensibile alla cultura degli editor numerici, di data e di valuta — e include traduzioni di sistema integrate per inglese, tedesco, francese, italiano, turco, spagnolo e portoghese. Esegui GlobalDesk in tedesco e i pulsanti Yes/No cambiano da soli.

Per una lingua che Wisej.NET non copre, o per una formulazione su cui il tuo prodotto insiste, Wisej.NET supporta un file `Resources-[LANG].resx` nella radice del progetto o in una cartella `/Resources`. Le sue chiavi sono quelle del framework e iniziano con un simbolo di dollaro: `$Ok`, `$Cancel`, `$Yes`, `$No`, `$Retry`, le etichette di navigazione del calendario e `$Offline`. Tieni separate le due cose: `Strings.de.resx` è quello che dice GlobalDesk, `Resources-de.resx` è quello che dice Wisej.NET.

Messaggi, validazione e immagini sotto la cultura attiva

Le frasi che citano dei dati hanno bisogno di segnaposto, ed è proprio sui segnaposto che chi inizia perde l'ordine delle parole. Conserva la frase completa nella risorsa — `Customer.LastOrder` contiene `{0} last ordered on {1}`. — e lascia che sia il codice a fornire i valori: `string.Format(Application.CurrentCulture, Texts.Get("Customer.LastOrder"), customerName, lastOrderDate)`. Passare la cultura formatta anche la data come la sessione si aspetta. Non concatenare mai `"Customer " + name + " saved"`: un traduttore non può spostare un frammento che il codice ha incollato insieme.

I segnaposto non dicono nulla a chi vede soltanto la colonna dei valori, quindi usa il campo commento dell'editor `.resx` per spiegare che cosa sono `{0}` e `{1}`. I messaggi di validazione meritano la stessa cura: `Validation.Required` e `Validation.Email` sono riutilizzati da più schermate, ed è per questo che appartengono al file condiviso e non a ciascuna form.

Un file `.resx` può contenere più delle sole stringhe — immagini e altri dati sono voci del tutto legittime, e la localizzazione nel designer può sostituire un'immagine per lingua. Fallo però con intenzione.

Un'immagine che cambia solo perché ha delle parole incorporate è una trappola di manutenzione; lascia quel testo come testo vero nell'interfaccia, dove `Texts.Get` lo traduce e nessun designer deve ridisegnare niente.



La regola dei livelli di localizzazione

Decidi dove vive ogni stringa prima di scriverla. Le risorse del designer appartengono a una singola schermata disegnata e vengono applicate quando i suoi controlli vengono creati; le risorse condivise dell'applicazione nei file `.resx` portano il testo che il codice genera o che più schermate riutilizzano; `Resources-[LANG].resx` sovrascrive le etichette di sistema di Wisej.NET. Non memorizzare mai una data, un numero o un importo già formattati: conserva il valore e formattalo per la cultura attiva al momento di mostrarlo. E ricorda che la cultura è per sessione, quindi la lingua di un utente non deve mai cambiare la schermata di un altro.



Obiettivi di apprendimento

Alla fine di questo modulo sarai in grado di:

- ✓ Creare file di risorse .resx neutri e specifici per cultura.
- ✓ Recuperare i valori attraverso ResourceManager usando chiavi stabili.
- ✓ Organizzare le stringhe riutilizzabili separandole dalle risorse del designer.
- ✓ Spiegare la localizzazione di sistema integrata di Wisej.NET e i file Resources-[LANG].resx personalizzati.
- ✓ Usare risorse consapevoli della cultura per messaggi, validazione e immagini.

Guida al lab / esame

Che cosa costruirai nel lab pratico, l'approccio suggerito e i deliverable richiesti.

Lab pratico

Fai crescere le risorse condivise di GlobalDesk. Aggiungi `Strings.de.resx` accanto a `Resources/Strings.resx` con esattamente le stesse chiavi e i valori tradotti, aggiungi una voce `Customer.LastOrder` che contiene due segnaposto, e componi la frase con `string.Format` e `Application.CurrentCulture` invece di concatenare frammenti. Fai passare dall'helper `Texts` ogni messaggio che l'editor cliente mostra, compresi la conferma di salvataggio e i due messaggi di validazione, e fai restituire al servizio di salvataggio una chiave di risultato stabile come `CustomerEditor.Saved` invece di una frase inglese che l'interfaccia stampa così com'è. Poi aggiungi un file `Resources-de.resx` che sovrascrive un'etichetta di sistema di Wisej, dimostrarlo in una `MessageBox` e scrivi una breve nota sulla differenza tra il testo del tuo prodotto e il testo di sistema di Wisej.

Punto di partenza

Apri GlobalDesk come l'ha lasciato il Modulo 2: la dashboard legge le sue etichette da `Resources/Strings.resx` attraverso l'helper `Texts`, e lo user control `CustomerEditor` è localizzato nel designer con una variante tedesca. La conferma di salvataggio, i due messaggi di validazione e la riga dell'ultimo ordine sono ancora letterali inglesi nel codice, e non esiste ancora nessun file di risorse condiviso tradotto.

Approccio suggerito

- ✓ Aggiungi `Resources/Strings.de.resx` accanto al file neutro e riporta l'elenco delle chiavi invariato, traducendo solo i valori; esegui l'applicazione in `de-DE` e verifica che le etichette della dashboard cambino prima di toccare qualunque riga di codice.
- ✓ Aggiungi una voce `Customer.LastOrder` il cui valore è una frase completa con `{0}` e `{1}`, riempi il campo commento così che un traduttore sappia che il primo è il nome del cliente e il secondo la data dell'ordine, e componi la riga con

```
string.Format(Application.CurrentCulture, Texts.Get("Customer.LastOrder"), customerName, lastOrderDate) .
```
- ✓ Sostituisci i letterali in `CustomerEditor` con chiamate a `Texts.Get` per `CustomerEditor.Saved`, `Validation.Required` e `Validation.Email`, ed elimina ogni concatenazione di stringhe che incontri lungo la strada.
- ✓ Modifica il servizio di salvataggio così che restituisca un risultato stabile — la chiave `CustomerEditor.Saved` o un codice di risultato — e lascia che sia l'editor a mappare quel risultato su un testo attraverso `Texts.Get`, così che nel livello dei servizi non resti nessuna frase inglese.
- ✓ Aggiungi `Resources-de.resx` nella radice del progetto o nella cartella `/Resources`, sovrascrivi un'etichetta di sistema di Wisej.NET come `$Ok` o `$Cancel`, poi mostra una `MessageBox` in tedesco e confronta i suoi pulsanti con l'esecuzione in inglese.

Deliverable richiesti

- ✓ Strings.de.resx con le stesse chiavi del file neutro e solo i valori tradotti
- ✓ Risorsa Customer.LastOrder composta con string.Format e la cultura attiva, mai concatenata
- ✓ Servizio di salvataggio che restituisce una chiave di risorsa o un codice di risultato invece di una frase inglese
- ✓ Resources-de.resx che sovrascrive almeno un'etichetta di sistema di Wisej, mostrata in una MessageBox
- ✓ Nota di lab che distingue testo di prodotto, testo di sistema di Wisej e testo diagnostico

Domande di autovalutazione

- ✓ Elimina una chiave da `Strings.de.resx` ed esegui in tedesco. Descrivi che cosa vede l'utente, perché non va in crash niente, e come te ne accorgeresti in un'applicazione reale con duecento chiavi.
- ✓ Il tuo servizio di salvataggio oggi restituisce `CustomerEditor.Saved`. Che cosa si romperebbe negli unit test, e nella schermata in tedesco, se restituisse invece la frase vera e propria?
- ✓ Un collega aggiunge la parola "Cancel" a `Resources-de.resx` sotto una chiave di sua invenzione. Spiega perché il pulsante della `MessageBox` non cambia, e dove apparteneva davvero quella stringa.



La regola dei livelli di localizzazione

Decidi dove vive ogni stringa prima di scriverla. Le risorse del designer appartengono a una singola schermata disegnata e vengono applicate quando i suoi controlli vengono creati; le risorse condivise dell'applicazione nei file .resx portano il testo che il codice genera o che più schermate riutilizzano; Resources-[LANG].resx sovrascrive le etichette di sistema di Wisej.NET. Non memorizzare mai una data, un numero o un importo già formattati: conserva il valore e formattalo per la cultura attiva al momento di mostrarlo. E ricorda che la cultura è per sessione, quindi la lingua di un utente non deve mai cambiare la schermata di un altro.