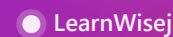


Fondamenti di localizzazione: cultura, lingua e risorse

Guida della lezione · Guida al lab / esame



Guida della lezione

Che cosa comprende davvero la localizzazione oltre alla traduzione: nomi di cultura come en, de-DE e fr-CA, file di risorse .resx letti tramite ResourceManager, il fallback di cultura di .NET verso la risorsa neutra, chiavi semantiche stabili al posto di frasi inglesi nel codice, e formattazione di date, numeri e importi sensibile alla cultura. 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

Che cosa comprende davvero la localizzazione oltre alla traduzione: nomi di cultura come en, de-DE e fr-CA, file di risorse .resx letti tramite ResourceManager, il fallback di cultura di .NET verso la risorsa neutra, chiavi semantiche stabili al posto di frasi inglesi nel codice, e formattazione di date, numeri e importi sensibile alla cultura.

1. Il problema in produzione

GlobalDesk è un piccolo service desk scritto per un solo ufficio. Ogni intestazione, etichetta di pulsante e messaggio di conferma è un letterale inglese digitato direttamente nel C#, e la dashboard stampa importi e date come capita di farlo a una macchina americana. Quando apre una filiale tedesca, quel testo non ha nessun posto dove andare — e chi comincia a correggere le etichette a mano incontra la seconda metà del problema: un'etichetta tedesca sopra un importo che recita ancora **1,850.75** non è una schermata tedesca.

2. Concetti chiave

Concetto	Perché è importante
localizzazione	Adattare un'applicazione alla lingua e alla cultura dell'utente: testo tradotto, date, numeri e importi formattati correttamente, spazio per parole più lunghe, e un layout che sappia rispecchiarsi per le lingue che si scrivono da destra a sinistra.
nome di cultura	Un identificatore come <code>en</code> , <code>de-DE</code> o <code>fr-CA</code> che indica una lingua e, facoltativamente, una regione; la parte della lingua decide quali risorse vengono preferite, il nome completo guida le convenzioni di formattazione.
risorsa neutra	L'insieme completo di risorse su cui si ricade, contenuto in un file come <code>Resources/Strings.resx</code> senza cultura nel nome; ogni chiave che l'applicazione chiede deve esistere qui.
<code>Wisej.Resources.ResourceManager</code>	L'oggetto che prende un nome base di risorse e un assembly e restituisce il valore di una chiave nella cultura che corrisponde meglio, così che il tuo codice non apra mai un file di persona.
fallback di cultura	La regola di .NET per trovare un valore: prova la cultura completa, poi la cultura solo lingua, poi la risorsa neutra, e usa la prima che contiene la chiave.
chiave di risorsa semantica	Un nome stabile che descrive il significato invece della formulazione, come <code>CustomerEditor.Save</code> o <code>Validation.Required</code> , così che il testo inglese possa essere riscritto senza toccare nessun'altra lingua.
<code>Application.CurrentCulture</code>	La cultura della sessione Wisej.NET corrente; due utenti collegati nello stesso momento possono avere due valori diversi, ed è per questo che la cultura non deve mai vivere in un campo statico.

3. Il filo della lezione

Tradurre è solo metà del lavoro

Dentro la parola "localizzazione" si nascondono due lavori diversi, e chi comincia di solito vede solo il primo. Il primo lavoro è la **localizzazione del testo dell'interfaccia**: le parole che l'utente legge, come l'intestazione di benvenuto di GlobalDesk, l'etichetta di navigazione Customers, il pulsante Save e il messaggio mostrato dopo che un cliente è stato salvato. Quelle parole le scrivi tu, vivono nei file di risorse, ed è la lingua della sessione a decidere quale versione viene restituita.

Il secondo lavoro è la **formattazione sensibile alla cultura**. Una data di scadenza, un conteggio di ticket e il totale di un ordine non sono affatto parole: sono valori. Lo stesso importo si scrive `$1,850.75` sotto `en-US` e `1.850,75 €` sotto `de-DE`, e nessuna di quelle due stringhe appartiene a un file di risorse o a una colonna di database. Tu memorizzi il `decimal` e il `DateTime`, e li formatti nel momento in cui li metti a schermo, passando la cultura della sessione come format provider.

È questa divisione la ragione per cui il lab si chiude con una nota scritta. Una volta che sai indicare qualcosa e dire "questo è un dato" oppure "questa è presentazione", il resto del corso ha dove appoggiarsi. Componi una stringa come "€ " + amount e hai scritto in silenzio nel codice un simbolo di valuta per ogni paese in cui rilascerai mai il prodotto.

Leggere un nome di cultura: en, de-DE, fr-CA

Un nome di cultura mette insieme una lingua e delle convenzioni regionali. Un nome **solo lingua** (neutro) è soltanto la lingua: en, de, fr. Un nome **lingua-regione** aggiunge un paese: en-US, de-DE, fr-CA, it-IT. In .NET sono valori CultureInfo, e Wisej.NET espone quello che vale per la sessione corrente come Application.CurrentCulture.

Due nomi possono essere d'accordo sulla lingua e in disaccordo su tutto il resto. Una sessione sotto en-GB e una sotto en-US leggono entrambe le tue risorse inglesi, perché entrambe selezionano la lingua en; scrivono comunque le date in modo diverso e usano simboli di valuta diversi. Succede anche il contrario: fr-CA e fr-FR condividono il testo francese ma non le convenzioni su numeri e valuta. Quindi la parte della lingua risponde alla domanda "quali parole?" e il nome completo risponde a "come si scrivono i valori?".

Per GlobalDesk questo conta fin da subito. Il pannello di anteprima della cultura che costruisci nel lab mostra affiancati la data di oggi, una quantità e un importo in valuta, così puoi guardare tre valori cambiare mentre le etichette sopra di loro restano in una sola lingua.

File .resx e ResourceManager

Un file .resx è un file di risorse XML che Visual Studio modifica come una griglia chiave/valore. Wisej.NET non ha bisogno di un formato di traduzione proprio: usa il modello standard di .NET, quindi i file .resx, CultureInfo, gli assembly satellite e gli strumenti RESX di terze parti funzionano tutti senza modifiche. Aggiungi Resources/Strings.resx come file neutro e, più avanti, Strings.de.resx accanto a lui con le stesse chiavi e i valori tedeschi.

Il tuo codice non apre mai quei file. Chiede una chiave a un ResourceManager costruito dal nome base delle risorse e dal suo assembly. Racchiudere tutto questo in un minuscolo helper statico ti dà un unico punto in cui decidere che aspetto ha una chiave mancante:

```
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 chiavi sono l'altra metà della disciplina. Usa CustomerEditor.Save, Navigation.Customers o Validation.Required: nomi che descrivono il significato, non la formulazione. Usare la frase inglese come chiave significa che ogni riscrittura rompe ogni file di traduzione. E localizza solo il testo di prodotto

— etichette, menu, tooltip, messaggi di validazione e di conferma. Gli identificatori dei log, i nomi dei tipi di eccezione e i nomi delle colonne del database restano come sono.

Fallback: tre tentativi, poi il file neutro

Quando una sessione che gira sotto `fr-CA` chiede una chiave, .NET non si arrende se `Strings.fr-CA.resx` non ha nulla per lei. Prova prima la risorsa più specifica, poi la meno specifica `fr`, poi la risorsa neutra, e restituisce il primo valore che trova. È per questo che il file neutro deve essere completo: è l'ultimo posto in cui la ricerca guarda, e una chiave che manca lì è una chiave che manca ovunque.

Il fallback è una rete di sicurezza, non una strategia di traduzione. Impedisce a un file tedesco lasciato a metà di svuotare una schermata, ma nasconde anche le lacune: l'utente vede l'inglese dove si aspettava il tedesco e potrebbe non segnalarlo mai. Quindi tieni acceso il fallback e vai a cercare le lacune di proposito — è a questo che serve il marcatore tra parentesi quadre nell'helper `Texts`. Un'etichetta che compare come `[Dashboard.Welcome]` è un bug che puoi vedere.

Preparare la dashboard di GlobalDesk

Preparare una pagina per la localizzazione è meccanico, una volta prese le decisioni. Costruisci la dashboard normalmente, poi passala in rassegna e chiediti di ogni stringa visibile: chi la legge? L'intestazione di benvenuto, il pulsante Customers e ogni label dell'anteprima sono letti dagli utenti, quindi ciascuno riceve una chiave in `Resources/Strings.resx` e una chiamata a `Texts.Get` al posto del suo letterale. Dodici chiavi sono un inizio realistico per una sola schermata.

Poi fai l'altra metà. Il pannello di anteprima non deve stampare testo che hai assemblato tu; formatta valori veri, passando `Application.CurrentCulture` a `ToString` così che il formato della data, i separatori e il simbolo di valuta vengano scelti per la sessione. Quella cultura è per sessione, quindi due utenti possono leggere due lingue nello stesso processo del server — e un campo di cultura statico permetterebbe a uno di loro di cambiare la schermata dell'altro.



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:

- ✓ Spiegare la differenza tra la localizzazione del testo dell'interfaccia e la formattazione sensibile alla cultura.
- ✓ Riconoscere i nomi di cultura neutri, solo lingua e lingua-regione come `en`, `de-DE` e `fr-CA`.
- ✓ Spiegare come i file `.resx` e `ResourceManager` partecipano alla localizzazione in .NET.

- ✓ Descrivere il fallback di cultura di .NET in termini adatti a chi comincia.
- ✓ Costruire una piccola pagina Wisej.NET con tutte le stringhe visibili all'utente pronte per la localizzazione.

Guida al lab / esame

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



Lab pratico

Crea la pagina dashboard di GlobalDesk con un'intestazione di benvenuto, un pulsante di navigazione Customers e un piccolo pannello di anteprima della cultura. Aggiungi Resources/Strings.resx come risorsa neutra con almeno dodici chiavi semantiche stabili come Navigation.Customers e CustomerEditor.Save, scrivi un helper statico Texts sopra Wisej.Resources.ResourceManager che restituisce la chiave tra parentesi quadre quando manca, e sostituisci ogni etichetta scritta direttamente nel codice con una chiamata a Texts.Get. Poi riempi il pannello di anteprima con valori veri: formatta la data di oggi, una quantità e un importo con Application.CurrentCulture invece di comporre le stringhe a mano, e concludi con una breve nota di lab che separa il valore memorizzato dalla sua presentazione sensibile alla cultura.

Punto di partenza

Apri il progetto Wisej.NET GlobalDesk in Visual Studio. È un piccolo service desk con una pagina principale, un abbozzo di editor cliente e una griglia dei contatti, e in questo momento ogni parola visibile all'utente al suo interno è un letterale inglese digitato nel C#, con importi e date stampati come capita al server di formattarli. In questo lab aggiungi la pagina dashboard e dai all'applicazione il suo primo livello di localizzazione: chiavi di risorsa per il testo e formattazione consapevole della cultura per i valori.

Approccio suggerito

- ✓ Aggiungi una pagina `Dashboard` e disponi un'intestazione di benvenuto, un pulsante di navigazione `btnCustomers` e un piccolo pannello di anteprima della cultura con tre label di sola lettura per una data, una quantità e un importo. Costruiscila prima in inglese, esattamente come faresti normalmente.
- ✓ Aggiungi `Resources/Strings.resx` come risorsa neutra e riempilo con almeno dodici chiavi semantiche come `Dashboard.Welcome`, `Navigation.Customers`, `CustomerEditor.Save` e `Preview.Amount`. Scrivi qui i valori inglesi, e da nessun'altra parte.
- ✓ Aggiungi una classe statica `Texts` che contiene un solo `Wisej.Resources.ResourceManager` costruito dal nome base delle risorse e dall'assembly dell'helper stesso, e un metodo `Get(string key)` che restituisce il valore oppure la chiave tra parentesi quadre `[key]` quando la ricerca non trova nulla.
- ✓ Passa in rassegna la pagina e sostituisci ogni etichetta scritta direttamente nel codice con una chiamata a `Texts.Get`. Poi chiedi di proposito una chiave che non hai mai aggiunto e verifica che a schermo compaia la chiave tra parentesi quadre invece di una label vuota.
- ✓ Riempi il pannello di anteprima con valori veri: chiama `ToString` su `DateTime.Today`, su una quantità e su un importo `decimal`, passando ogni volta `Application.CurrentCulture`. Non concatenare mai un simbolo o un separatore di tua iniziativa, poi scrivi la nota di lab.

Deliverable richiesti

- ✓ Pagina dashboard di GlobalDesk le cui etichette provengono tutte da chiavi di risorsa, senza testo visibile all'utente scritto direttamente nel codice
- ✓ File neutro Resources/Strings.resx con almeno dodici chiavi semantiche stabili
- ✓ Helper Texts sopra Wisej.Resources.ResourceManager che segnala una chiave mancante invece di restituire una stringa vuota
- ✓ Pannello di anteprima della cultura che formatta una data, un numero e un importo con Application.CurrentCulture
- ✓ Nota di lab che spiega quali valori sono dati e quali sono presentazione sensibile alla cultura

Domande di autovalutazione

- ✓ Cambi la formulazione inglese del pulsante Save da "Save" a "Save customer". Quali file devono cambiare se la tua chiave è `CustomerEditor.Save`, e quali avrebbero dovuto cambiare se la chiave fosse stata la frase inglese stessa?
- ✓ Il tuo helper `Texts` restituisce `[Preview.Amount]` per una label. Indica due errori diversi che producono esattamente quel marcatore, e di' come li distingueresti senza un debugger.
- ✓ Un collega propone di memorizzare il totale dell'ordine nel database come la stringa "1.850,75 €" così che gli utenti tedeschi lo vedano sempre correttamente. Spiega che cosa si rompe, e quale parte di quella stringa è dato e quale è presentazione.



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.