

Lingue da destra a sinistra e layout a prova di localizzazione

Guida della lezione · Guida al lab / esame



Guida della lezione

Il supporto da destra a sinistra in Wisej.NET: `RightToLeft` con `Inherit`, `No` e `Yes`, `RightToLeftLayout` che rispecchia i figli di un contenitore, l'impostazione applicativa `rightToLeft`, layout che sopravvivono all'espansione del testo, lo stato `rightToLeft` del tema, e i limiti documentati su `DataGridView`, `ListView` e `PropertyGrid`. 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 supporto da destra a sinistra in Wisej.NET: `RightToLeft` con `Inherit`, `No` e `Yes`, `RightToLeftLayout` che rispecchia i figli di un contenitore, l'impostazione applicativa `rightToLeft`, layout che sopravvivono all'espansione del testo, lo stato `rightToLeft` del tema, e i limiti documentati su `DataGridView`, `ListView` e `PropertyGrid`.

1. Il problema in produzione

GlobalDesk sa già cambiare lingua mentre è in esecuzione: il selettore imposta

`Application.CurrentCulture`, il gestore di `Application.CultureChanged` riapplica ogni etichetta che arriva dalle risorse, e il pannello di anteprima riformatta i suoi campioni. Poi un tester apre la dashboard in arabo. Ogni parola è tradotta e ognuna è nel posto sbagliato: le label si stringono al bordo sinistro dei loro campi, la colonna di navigazione sta dove l'occhio di un lettore arabo arriva per ultimo, e le lunghe etichette tedesche del modulo precedente spingono già un pulsante fuori da un pannello a larghezza fissa.

La traduzione era la metà facile. Una schermata localizzata ha anche una direzione di lettura e una larghezza che non controlla, e sono entrambe decisioni di layout che prendi mentre costruisci la form, non riparazioni che fai dopo.

2. Concetti chiave

Concetto	Perché è importante
RightToLeft	Una proprietà del controllo con i valori <code>Inherit</code> , <code>No</code> e <code>Yes</code> che dice a un singolo controllo di adattare la direzione del proprio testo e dei propri elementi interni.
RightToLeft.Inherit	La scelta normale per un controllo figlio: segue il genitore, così l'intera schermata cambia direzione a partire da una sola decisione presa in cima.
RightToLeftLayout	Una proprietà diversa che rispecchia la posizione orizzontale dei figli diretti di un contenitore quando RTL è attivo, così la disposizione visiva si ribalta mentre il tuo layout logico resta come l'hai disegnato.
L'impostazione applicativa <code>rightToLeft</code>	Una voce in <code>Default.json</code> ; impostata su <code>"auto"</code> fa sì che Wisej.NET deduca il comportamento RTL dalla cultura della sessione invece che da controlli sul nome di cultura sparsi per le tue schermate.
<code>Application.RightToLeft</code>	La stessa decisione espressa nel codice, per le sessioni che devono essere commutate deliberatamente invece che dal rilevamento della cultura.
Espansione del testo	Un'etichetta tradotta ha raramente la lunghezza di quella inglese, ed è per questo che le posizioni fisse in pixel e le larghezze dei pannelli misurate a mano sono le prime a rompersi in una build localizzata.
I limiti RTL documentati	Wisej.NET documenta che <code>DataGridView</code> , <code>ListView</code> e <code>PropertyGrid</code> non supportano pienamente RTL, quindi un piano di test deve nominarli invece di dare per scontato che vengano rispecchiati.

3. Il filo della lezione

Due proprietà, due compiti

Il primo errore che fanno quasi tutti è trattare la scrittura da destra a sinistra come un'impostazione di allineamento del testo. L'arabo, l'ebraico e le altre lingue che si scrivono da destra a sinistra hanno bisogno che il testo scorra nell'altro verso e, di solito, che la disposizione della schermata venga rispecchiata. Wisej.NET separa questi due compiti in due proprietà, e sapere quale ti serve è gran parte di questo modulo.

`RightToLeft` appartiene a un singolo controllo e cambia quello che succede al suo interno: la direzione del suo testo e degli elementi che disegna per sé. Accetta `Inherit`, `No` o `Yes`. Lascialo su `Inherit` quasi ovunque, perché così un figlio segue il genitore e l'intera dashboard di GlobalDesk risponde a una sola decisione presa in cima. Impostare `Yes` sui singoli controlli ti dà una schermata in cui la direzione è configurata in cinquanta punti e nessuno sa dire quale sia quello sbagliato.

`RightToLeftLayout` è l'altro compito. Rispecchia la posizione orizzontale dei figli diretti di un contenitore quando RTL è attivo, così continui a disegnare l'editor cliente nell'ordine che ha senso per te mentre Wisej.NET ribalta la disposizione visiva. Fai attenzione alle parole "figli diretti": impostarlo sulla dashboard

rispecchia i pannelli che la dashboard contiene, e ogni pannello decide per conto suo se rispecchiare i propri controlli. È per questo che il lab lo imposta sulla dashboard e sull'editor cliente.

Lascia decidere l'impostazione applicativa

`Default.json` porta un'impostazione `rightToLeft` accanto all'impostazione `culture` che hai configurato nel modulo precedente. Con il valore `"auto"`, Wisej.NET deduce il comportamento RTL dalla cultura sotto cui gira la sessione, quindi aggiungere l'arabo al selettore basta a far sì che la direzione segua la lingua. `Application.RightToLeft` espone la stessa decisione al codice quando una sessione deve essere guidata deliberatamente.

L'alternativa è il pattern da evitare: un test sul nome di cultura dentro ogni schermata, poi un secondo elenco di prefissi quando arriva l'ebraico, poi una schermata che qualcuno ha dimenticato. Una sola impostazione applicativa sostituisce tutto questo, ed è il motivo per cui i tuoi controlli possono restare su `Inherit`. La direzione diventa stato di sessione, esattamente come `Application.CurrentCulture`: un tester che passa all'arabo non deve mai rispecchiare la schermata di un collega.

Disegnare un layout che può essere rispecchiato

Il rispecchiamento viene bene solo se il layout era flessibile fin dall'inizio. Un controllo a una `Location` misurata a mano dentro un pannello di dimensioni fisse viene rispecchiato in una posizione che nessuno ha mai disegnato, e un'etichetta che in tedesco è il 40% più lunga esce semplicemente dal suo bordo. Gli strumenti sono quelli ordinari di layout di Wisej.NET: il docking perché una regione possieda un bordo, l'ancoraggio perché un controllo cresca con il suo contenitore, `AutoSize` dove una label deve poter essere lunga quanto la sua traduzione, un contenitore di layout dove una riga di pulsanti deve potersi ridisporre, e dimensioni minime sensate perché nulla collassi.

C'è un'eccezione deliberata, e il lab ti chiede di scrivere il commento che la spiega. Alcune stringhe non sono prosa: un codice cliente, un numero di serie, il contenuto di un editor di codice. Quelli si leggono da sinistra a destra in ogni cultura, e ribaltarli trasforma un identificatore corretto in uno confuso. È a questo che serve `RightToLeft.No` — usato con parsimonia, sempre con accanto una ragione.

```
private void ConfigureDirection()
{
    // Default.json porta rightToLeft "auto", quindi Application.RightToLeft
    // segue già la cultura della sessione. I figli restano su Inherit.
    this.RightToLeft = RightToLeft.Inherit;
    this.RightToLeftLayout = true;

    // Il codice cliente è un identificatore tecnico, non prosa:
    // deve continuare a leggersi da sinistra a destra in ogni cultura.
    txtCustomerCode.RightToLeft = RightToLeft.No;
}
```

Testare l'espansione e i controlli rispecchiati

Nessuno dei due problemi è visibile nel codice, quindi il modulo si chiude con un elenco di test invece che con una funzionalità. Fai passare ogni schermata localizzata attraverso almeno cinque casi:

- Testo corto nella lingua neutra, che è lo stato in cui è stato disegnato tutto.

- Una lingua con traduzioni sensibilmente più lunghe, come il tedesco.
- Una finestra del browser stretta, che è dove l'ancoraggio e le dimensioni minime si guadagnano lo stipendio.
- Una cultura RTL come l'arabo, controllando la disposizione e non solo l'allineamento.
- Contenuto misto, come una label in arabo accanto a un codice prodotto in caratteri latini.

Un file di risorse pseudo-localizzato rende onesti i casi di espansione. Copia le chiavi neutre, poi allunga e racchiudi tra parentesi ogni valore così che un'etichetta diventi qualcosa come `[[Save changes ~~~~]]`. Tutto ciò che si legge ancora come inglese puro non era una chiave di risorsa, e tutto ciò che si tronca ti dice quale controllo ha bisogno di ancoraggio. Anche i temi di Wisej.NET conoscono uno stato `rightToLeft`, così le icone e lo stile su un solo lato possono adattarsi senza un secondo tema; le regole di tema personalizzate e i widget personalizzati sono le parti che potrebbero ancora aver bisogno di un rispecchiamento scritto a mano.

I controlli con limiti documentati

Infine, la parte onesta. Wisej.NET documenta che `DataGridView`, `ListView` e `PropertyGrid` non supportano pienamente la scrittura da destra a sinistra. GlobalDesk mostra i suoi contatti in una `DataGridView`, quindi il lab ti chiede di annotare che cosa fa davvero quella griglia in arabo invece di quello che ti aspettavi. L'ordine delle colonne, l'allineamento delle intestazioni e il lato della barra di scorrimento sono le cose da guardare.

Non è una ragione per evitare quei controlli; è una ragione per nominarli nel piano di test e verificarli sull'esatta versione e configurazione che rilasci. Un layout localizzato non deve mai troncare un'etichetta tradotta, spingere un controllo fuori dall'area visibile, dipendere dalla lunghezza delle parole inglesi, incastonare una freccia direzionale in un'immagine senza variante RTL, o forzare un identificatore nella direzione di lettura sbagliata — ognuna di queste cose costa meno da correggere sulla prima schermata che sulla centesima.



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:

- ✓ Usare correttamente `RightToLeft` e `RightToLeftLayout` di Wisej.NET.
- ✓ Spiegare l'impostazione applicativa `rightToLeft="auto"`.
- ✓ Costruire un layout che si rispecchia in modo appropriato per le culture RTL.

- ✓ Testare l'espansione del testo e i controlli rispecchiati.
- ✓ Identificare i controlli Wisej il cui supporto RTL ha limitazioni documentate.

Guida al lab / esame

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



Lab pratico

Fai sopravvivere GlobalDesk a una cultura che si scrive da destra a sinistra. Aggiungi l'arabo al selettore della lingua e imposta `rightToLeft` su `auto` in `Default.json`, lascia i controlli figli su `RightToLeft.Inherit`, e imposta `RightToLeftLayout` sulla dashboard e sull'editor cliente così che i loro figli vengano rispecchiati. Tieni esattamente un controllo da sinistra a destra, il campo del codice cliente, e spiega in un commento perché quell'identificatore non deve ribaltarsi. Ricostruisci con docking, ancoraggio e un contenitore di layout la parte a posizione fissa del layout, così una traduzione lunga non può troncarsi, aggiungi un file di risorse pseudo-localizzato i cui valori sono allungati e racchiusi tra parentesi, e annota che cosa fa davvero la `DataGridView` dei contatti sotto RTL invece di dare per scontato che venga rispecchiata. Concludi con un elenco di cinque casi di test del layout che puoi rieseguire su qualsiasi schermata.

Punto di partenza

GlobalDesk dopo il Modulo 4: la dashboard, l'editor cliente e la griglia dei contatti leggono tutti le proprie etichette dalle risorse, il selettore della lingua offre l'inglese, il tedesco e una cultura lingua-regione, e un gestore di `Application.CultureChanged` chiama già `ApplyTextResources` e `UpdateCulturePreview`. Ogni schermata è ancora costruita da sinistra a destra, e il pannello di navigazione della dashboard usa ancora i valori fissi di `Location` e `Size` del Modulo 1.

Approccio suggerito

- ✓ Aggiungi una voce araba al selettore della lingua, imposta `rightToLeft` su `"auto"` accanto all'impostazione `culture` già presente in `Default.json`, poi esegui l'applicazione e passa all'arabo prima di cambiare qualsiasi altra cosa, così puoi vedere esattamente che cosa fa da solo il rilevamento automatico.
- ✓ Imposta `RightToLeftLayout` a `true` sulla dashboard e sull'editor cliente, e verifica che ogni controllo figlio sia ancora su `RightToLeft.Inherit` — resisti alla tentazione di impostare `Yes` sui singoli controlli per "sistemare" quello che dovrebbe decidere il contenitore.
- ✓ Imposta `RightToLeft.No` solo sul campo del codice cliente, e scrivi il commento che spiega perché un identificatore tecnico deve conservare la propria direzione di lettura mentre il testo attorno si ribalta.
- ✓ Sostituisci le posizioni assolute del pannello di navigazione con docking e ancoraggio, metti la riga dei pulsanti dell'editor in un contenitore di layout così che possa ridisporsi, e dai alle etichette `AutoSize` e ai pannelli dimensioni minime sensate.
- ✓ Aggiungi un file di risorse pseudo-localizzato i cui valori sono allungati e racchiusi tra parentesi, fai passare i cinque casi di test attraverso quel file, annota che cosa fa davvero la `DataGridView` dei contatti in arabo, e trasforma l'intero giro in un elenco di test riutilizzabile.

Deliverable richiesti

- ✓ Voce araba nel selettore della lingua con `rightToLeft` impostato su `auto` in `Default.json`

- ✓ Dashboard ed editor cliente che usano `RightToLeftLayout` con i figli lasciati su `Inherit`
- ✓ Un controllo deliberatamente da sinistra a destra con un commento che spiega l'eccezione
- ✓ Layout ricostruito con docking, ancoraggio e un contenitore di layout così che le traduzioni lunghe non possano troncarsi
- ✓ Elenco di test RTL che copre testo corto, testo lungo, una finestra stretta, una cultura RTL e il limite della `DataGridView`

Domande di autovalutazione

- ✓ Un collega imposta `RightToLeft.Yes` su venti controlli e la schermata araba sembra corretta, ma quella tedesca è rimasta invariata. Che cosa ha configurato in realtà, e quale singola proprietà avrebbe fatto il lavoro per l'intero contenitore?
- ✓ Il tuo giro pseudo-localizzato mostra tre etichette ancora in inglese puro. Niente si è troncato e niente ha sollevato un'eccezione. Che cosa ti dice questo su quelle tre etichette, e il lavoro di quale modulo precedente ha una lacuna?
- ✓ La griglia dei contatti non viene rispecchiata come il resto della dashboard. Quale delle tue opzioni — accettarlo e documentarlo, sostituire il controllo, oppure scrivere il rispecchiamento a mano — puoi difendere davanti a un revisore, e che cosa dovresti verificare prima di prometterne una qualsiasi?



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.