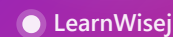


Cultura del browser, cultura di sessione e cambio lingua a runtime

Guida della lezione · Guida al lab / esame



Guida della lezione

Da dove arriva la cultura di una sessione e che cosa deve succedere quando cambia: l'impostazione culture in Default.json, il rilevamento automatico del browser, il parametro URL lang, Application.CurrentCulture come stato della singola sessione, l'evento Application.CultureChanged, e la riapplicazione del testo che viene dalle risorse perché nessuna schermata tradotta a metà sopravviva al cambio. 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

Da dove arriva la cultura di una sessione e che cosa deve succedere quando cambia: l'impostazione culture in Default.json, il rilevamento automatico del browser, il parametro URL lang, Application.CurrentCulture come stato della singola sessione, l'evento Application.CultureChanged, e la riapplicazione del testo che viene dalle risorse perché nessuna schermata tradotta a metà sopravviva al cambio.

1. Il problema in produzione

Dopo il Modulo 3, GlobalDesk ha un file di risorse neutro, uno tedesco accanto, un helper `Texts`, un editor cliente localizzato nel designer e perfino un'etichetta di sistema di Wisej tutta sua. Quello a cui ancora non sa rispondere è la domanda che un utente vero fa per prima: come lo ottengo nella mia lingua, e che cosa succede alla schermata quando lo chiedo? Fin qui tutto è stato deciso mentre l'applicazione veniva costruita; questo modulo parla dei due momenti in cui la decisione viene presa davvero — quando una sessione inizia, e quando un utente cambia idea a metà strada.

2. Concetti chiave

Concetto	Perché è importante
rilevamento automatico del browser	Wisej.NET può leggere la lingua che il browser dichiara e avviare la sessione con quella, così ogni utente ottiene una cultura sensata senza che un'unica impostazione valida per tutto il server decida per tutti.
<code>culture</code> in <code>Default.json</code>	L'impostazione dell'applicazione che decide da dove arriva la cultura di una nuova sessione; il suo valore predefinito è <code>auto</code> , mentre un nome fisso come <code>de-DE</code> fissa invece ogni sessione su un solo locale.
<code>rightToLeft</code> in <code>Default.json</code>	L'impostazione vicina, il cui valore <code>auto</code> segue qualunque cultura finisca per avere la sessione; il Modulo 5 ci costruisce sopra, ma la incontri qui nello stesso file.
<code>Application.CurrentCulture</code>	Legge o imposta la cultura della <i>sessione Wisej.NET corrente</i> , che è esattamente lo scope di cui ha bisogno un'applicazione web multi-utente: la scelta di un utente non deve mai ridisegnare la schermata di un altro.
parametro URL <code>lang</code>	Un modo supportato per chiedere una lingua già in ingresso, per esempio <code>?lang=de</code> , che rende banale produrre link di test e URL di ingresso specifici per lingua.
<code>Application.CultureChanged</code>	Viene sollevato quando la cultura corrente cambia rispetto a quella con cui la sessione è partita; è l'unico posto in cui riapplicare il testo che viene dalle risorse e riformattare tutto ciò che è già sullo schermo.
controlli localizzati nel designer	I loro valori localizzati vengono applicati quando i controlli vengono creati, quindi un cambio di cultura successivo non li riscrive — ricrei il contenitore oppure riapplich i valori nel codice.

3. Il filo della lezione

Da dove arriva la cultura di una sessione

I browser dichiarano a ogni richiesta le lingue che il loro utente preferisce, e Wisej.NET può usare quella dichiarazione per scegliere la cultura della sessione che sta per avviare. È un comportamento predefinito molto migliore di un'impostazione del server, perché il server ha esattamente una configurazione regionale mentre i suoi utenti no: due persone che aprono GlobalDesk nello stesso secondo da browser impostati su inglese e tedesco ottengono ciascuna la propria sessione, ciascuna con la propria cultura, nello stesso processo.

Il rilevamento è solo un punto di partenza, non una decisione da cui non si torna indietro. La lingua di un utente può altrettanto bene arrivare da una riga di profilo, da una preferenza di accesso o dal selettore che stai per costruire. La regola che conta è la stessa qualunque sia la fonte che vince: risolvila una volta sola, all'inizio della sessione, e carica lì in `Application.CurrentCulture`. Il codice che più avanti legge le risorse o formatta un valore non deve mai doversi chiedere di nuovo da dove arrivasse la lingua.

Configurare l'impostazione culture in Default.json

`Default.json` è il posto in cui l'applicazione dichiara la propria intenzione. L'impostazione `culture` ha come valore predefinito `auto`, cioè il comportamento di rilevamento del browser appena descritto, e l'impostazione vicina `rightToLeft` si comporta allo stesso modo — `auto` segue la cultura. Nel lab lasci `culture` su `auto` così che GlobalDesk parta dal browser, poi lo dimostri cambiando la lingua preferita del tuo browser e ricaricando.

Un valore fisso non è un errore, è un'intenzione diversa. Fissare l'impostazione su un solo nome di cultura è la scelta onesta per un prodotto che viene rilasciato di proposito in un solo locale, ed è il modo più rapido per riprodurre una segnalazione su una lingua mentre stai facendo debug: ogni sessione finisce in quella cultura qualunque cosa dica il browser del tester. Metti per iscritto quale delle due hai scelto e perché, perché lo sviluppatore successivo leggerà quell'impostazione come una dichiarazione sul prodotto.

CurrentCulture è per sessione, mai un campo statico

Cambiare la lingua a runtime è una sola assegnazione: imposta `Application.CurrentCulture` sulla cultura scelta dall'utente, costruita con `CultureInfo.GetCultureInfo` a partire da un nome come `"en-US"`, `"de-DE"` o `"fr-CA"`. Poiché la proprietà ha come scope la sessione Wisej.NET, quell'assegnazione cambia il mondo di una sola scheda del browser e nient'altro. La ComboBox della lingua nel lab fa esattamente questo nel suo gestore `SelectedIndexChanged`, memorizzando i nomi di cultura come valori degli item così che il gestore non debba mai interpretare un'etichetta visualizzata.

La scorciatoia allettante è un `static CultureInfo` in un punto comodo, "così lo vedono tutti". I campi statici vivono nel processo del server, e un server Wisej.NET ospita ogni sessione in quell'unico processo, quindi il campo conterrebbe il valore scelto dall'utente più recente. Un collega in un altro paese vedrebbe la propria schermata cambiare lingua perché tu stavi provando il tedesco. Se più classi hanno bisogno della cultura attiva, leggila ogni volta da `Application.CurrentCulture`; è già la memoria per sessione che stavi per reinventare.

Il parametro lang e l'evento CultureChanged

Due funzionalità di Wisej.NET racchiudono il cambio. In ingresso, il parametro URL `lang` chiede una lingua prima che venga eseguita qualunque riga del tuo codice, così `?lang=de` apre GlobalDesk in tedesco senza toccare le impostazioni del browser — ideale per un link di test, una segnalazione di bug o un URL di ingresso specifico per lingua da consegnare a un cliente. Lungo il percorso, `Application.CultureChanged` viene sollevato quando la cultura corrente cambia rispetto alla cultura iniziale della sessione, ed è il tuo aggancio per tutto ciò che è già sullo schermo.

Iscriviti una volta sola, quando la pagina viene costruita, e fai il lavoro in un unico gestore così che niente dipenda da quale controllo ha provocato il cambio:

```

public MainPage()
{
    InitializeComponent();
    Application.CultureChanged += Application_CultureChanged;
    ApplyTextResources();
}

private void Application_CultureChanged(object sender, EventArgs e)
{
    ApplyTextResources(); // ogni etichetta che viene dalle risorse
    UpdateCulturePreview(); // esempi di data, numero e importo
    RecreateCustomerEditor(); // localizzato nel designer: ricostruiscilo
}

```

Ridisegnare quello che il cambio si è lasciato dietro

Un cambio di cultura non torna indietro attraverso le etichette che hai già assegnato.

`ApplyTextResources` percorre i controlli il cui testo veniva dal livello delle risorse e chiede di nuovo a `Texts.Get` ogni chiave, così l'intestazione, il pulsante Customers e ogni messaggio leggono i loro nuovi valori. `UpdateCulturePreview` è il suo gemello per la formattazione: i dati memorizzati non sono cambiati affatto, quindi prende la stessa data, la stessa quantità e lo stesso importo decimale e li rende con la cultura ora attiva, ed è per questo che conservi il valore e mai la stringa formattata.

L'editor cliente localizzato nel designer è la parte su cui in molti inciampano. I valori localizzati delle sue proprietà vengono applicati quando i suoi controlli vengono creati, quindi conserva le etichette con cui è nato per quante volte la cultura cambi in seguito. La soluzione è strutturale, non ingegnosa: fai il dispose dell'editor e aggiungi una nuova istanza, chiudilo e riaprilo, oppure naviga verso una pagina appena creata. Qualunque strada tu scelga, falla nello stesso gestore di `CultureChanged`, e poi guarda tutta la schermata — un cambio che lascia metà finestra nella lingua precedente è peggio di nessun cambio, perché ora l'utente non si fida di nessuna delle due metà.



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 il rilevamento automatico della lingua del browser di Wisej.NET.
- ✓ Configurare culture="auto" o una cultura fissa in Default.json.

- ✓ Cambiare la cultura della sessione con `Application.CurrentCulture`.
- ✓ Usare il parametro URL `lang` e gestire `Application.CultureChanged`.
- ✓ Riapplicare l'interfaccia che viene dalle risorse dopo un cambio lingua a runtime.

Guida al lab / esame

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



Lab pratico

Dai a GlobalDesk un cambio lingua che porta a termine il lavoro. Imposta culture su auto in `Default.json` così che la sessione parta dalla lingua del browser, aggiungi una `ComboBox` della lingua che offre inglese, tedesco e una cultura lingua-regione, e imposta `Application.CurrentCulture` da `CultureInfo.GetCultreInfo` nel suo gestore `SelectedIndexChanged` senza mai memorizzare la scelta in un campo statico. Gestisci `Application.CultureChanged` per chiamare un metodo `ApplyTextResources` che aggiorna ogni etichetta che viene dalle risorse e un metodo `UpdateCulturePreview` che riformatta gli esempi di data, numero e importo, e ricrea nello stesso gestore l'editor cliente localizzato nel designer. Verifica la stessa pagina attraverso il parametro URL `lang`, poi scrivi una nota di lab che descrive che cosa farebbe un campo di cultura statico a un secondo utente collegato nello stesso momento.

Punto di partenza

Apri GlobalDesk come l'hai lasciato dopo il Modulo 3: la dashboard legge ogni etichetta da `Texts.Get`, `Resources/Strings.resx` e `Strings.de.resx` contengono le stesse chiavi, il pannello di anteprima della cultura formatta una data, un numero e un importo, e l'editor cliente ha una variante tedesca nel designer. Sullo schermo non c'è ancora niente che permetta a un utente di cambiare lingua, e `Application.CultureChanged` non ha nessun iscritto.

Approccio suggerito

- ✓ Apri `Default.json`, verifica che `culture` sia `auto`, e dimostrarlo: cambia in tedesco la lingua preferita del tuo browser, ricarica GlobalDesk e annota quali etichette arrivano tradotte. Scrivi in una riga perché invece fisseresti l'impostazione su un nome di cultura fisso.
- ✓ Aggiungi alla dashboard una `ComboBox` `languageCombo` con tre item i cui valori sono nomi di cultura — `en-US`, `de-DE` e una cultura lingua-regione come `fr-CA` — e le cui etichette sono a loro volta chiavi di risorsa, non parole scritte direttamente nel codice.
- ✓ In `languageCombo_SelectedIndexChanged`, assegna `Application.CurrentCulture = CultureInfo.GetCultreInfo(name)` e nient'altro. Resisti a ogni tentazione di mettere in cache la scelta in un campo `static`; la sessione la memorizza già.
- ✓ Iscriviti a `Application.CultureChanged` nel costruttore della pagina e scrivi i due metodi che chiama: `ApplyTextResources` rilegge attraverso `Texts.Get` ogni etichetta che viene dalle risorse, e `UpdateCulturePreview` riformatta con `Application.CurrentCulture` la stessa data, la stessa quantità e lo stesso importo decimale memorizzati.
- ✓ Nello stesso gestore, fai il dispose dell'editor cliente localizzato nel designer e aggiungi una nuova istanza al suo contenitore ospite, poi cambia lingua con l'editor aperto e controlla ogni etichetta sullo schermo. Concludi aprendo `?lang=de` in una seconda sessione del browser e aggiungendo la tua nota di lab sullo stato di cultura statico.

Deliverable richiesti

- ✓ Default.json con culture impostato su auto, più una nota su quando una cultura fissa è la scelta giusta
- ✓ Gestore della ComboBox della lingua che imposta Application.CurrentCulture da CultureInfo.GetCultureInfo
- ✓ Gestore di Application.CultureChanged che chiama ApplyTextResources e UpdateCulturePreview
- ✓ Editor localizzato nel designer ricreato dopo il cambio, così nessuna etichetta resta nella lingua precedente
- ✓ Verifica attraverso il parametro URL lang e una nota sul perché la cultura non deve vivere in un campo statico

Domande di autovalutazione

- ✓ Il tuo metodo `ApplyTextResources` aggiorna perfettamente la dashboard, eppure l'editor cliente conserva le sue etichette inglesi dopo il passaggio al tedesco. Spiega che cosa hanno di diverso quelle etichette, e indica due strategie che risolvono il problema.
- ✓ Un collega propone di mettere in cache in un campo statico il `CultureInfo` scelto "per non doverlo cercare due volte". Descrivi che cosa vedrebbe davvero un secondo utente, collegato da un altro paese nello stesso momento.
- ✓ L'anteprima della cultura mostra un importo in valuta. Dopo un passaggio da `en-US` a `de-DE`, che cosa è cambiato in memoria e che cosa è cambiato solo sullo schermo — e quale delle due cose scriveresti mai nel database?



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.