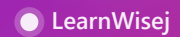


Browser-Kultur, Sitzungskultur und Sprachwechsel zur Laufzeit

Lektionsleitfaden · Lab- und Prüfungsleitfaden



Lektionsleitfaden

Woher die Kultur einer Sitzung kommt und was geschehen muss, wenn sie sich ändert: die Einstellung culture in Default.json, die automatische Browsererkennung, der URL-Parameter lang, Application.CurrentCulture als Zustand pro Sitzung, das Ereignis Application.CultureChanged und das erneute Anwenden ressourcengestützter Texte, damit kein halb übersetzter Bildschirm den Wechsel übersteht. Was das für Entwicklerinnen und Entwickler bedeutet, die die Grundlagen von C# und die Wisej.NET-Controls kennen und die Anwendung nun in mehr als einer Sprache ausliefern müssen – und wie Sie dieses Modul lesen.



Fokus: Produktivbetrieb

Woher die Kultur einer Sitzung kommt und was geschehen muss, wenn sie sich ändert: die Einstellung culture in Default.json, die automatische Browsererkennung, der URL-Parameter lang, Application.CurrentCulture als Zustand pro Sitzung, das Ereignis Application.CultureChanged und das erneute Anwenden ressourcengestützter Texte, damit kein halb übersetzter Bildschirm den Wechsel übersteht.

1. Das Problem im Produktivbetrieb

Nach Modul 3 hat GlobalDesk eine neutrale Ressourcendatei, eine deutsche daneben, einen **Texts** - Helper, einen im Designer lokalisierten Kundeneditor und sogar eine eigene Systembeschriftung von Wisej. Worauf es noch keine Antwort hat, ist die Frage, die ein echter Benutzer zuerst stellt: Wie bekomme ich das in meiner Sprache, und was passiert mit dem Bildschirm, wenn ich danach frage? Alles bisher wurde während des Bauens der Anwendung entschieden; in diesem Modul geht es um die beiden Momente, in denen die Entscheidung tatsächlich fällt — wenn eine Sitzung startet und wenn ein Benutzer mittendrin seine Meinung ändert.

2. Zentrale Konzepte

Konzept	Warum es wichtig ist
automatische Browsererkennung	Wisej.NET kann die vom Browser gemeldete Sprache lesen und die Sitzung damit starten, sodass jeder Benutzer eine sinnvolle Kultur erhält, ohne dass eine serverweite Einstellung für alle entscheidet.
<code>culture</code> in <code>Default.json</code>	Die Anwendungseinstellung, die entscheidet, woher die Kultur einer neuen Sitzung kommt; sie steht standardmäßig auf <code>auto</code> , und ein fester Name wie <code>de-DE</code> bindet stattdessen jede Sitzung an ein Gebietsschema.
<code>rightToLeft</code> in <code>Default.json</code>	Die Nachbareinstellung, deren Wert <code>auto</code> derjenigen Kultur folgt, bei der die Sitzung schließlich landet; Modul 5 baut darauf auf, aber Sie begegnen ihr hier in derselben Datei.
<code>Application.CurrentCulture</code>	Liest oder setzt die Kultur der <i>aktuellen Wisej.NET-Sitzung</i> — genau der Geltungsbereich, den eine Mehrbenutzer-Webanwendung braucht: Die Wahl eines Benutzers darf niemals den Bildschirm eines anderen neu zeichnen.
URL-Parameter <code>lang</code>	Eine unterstützte Möglichkeit, beim Einstieg eine Sprache anzufordern, zum Beispiel <code>?lang=de</code> , wodurch sich Testlinks und sprachspezifische Einstiegs-URLs mühelos erzeugen lassen.
<code>Application.CultureChanged</code>	Wird ausgelöst, wenn die aktuelle Kultur von der abweicht, mit der die Sitzung gestartet ist; die eine Stelle, an der Sie ressourcengestützten Text erneut anwenden und alles bereits Sichtbare neu formatieren.
im Designer lokalisierte Controls	Ihre lokalisierten Werte werden beim Erstellen der Controls angewendet, deshalb schreibt ein späterer Kulturwechsel sie nicht neu — Sie bauen den Container neu auf oder setzen die Werte im Code erneut.

3. Der rote Faden der Lektion

Woher die Kultur einer Sitzung kommt

Browser kündigen bei jeder Anfrage an, welche Sprachen ihr Benutzer bevorzugt, und Wisej.NET kann diese Ankündigung nutzen, um die Kultur für die Sitzung zu wählen, die es gleich startet. Das ist ein weit besserer Standard als eine Servereinstellung, denn der Server hat genau eine regionale Konfiguration, seine Benutzer aber nicht: Zwei Personen, die GlobalDesk in derselben Sekunde aus Browsern mit englischer und deutscher Einstellung öffnen, bekommen jeweils ihre eigene Sitzung mit ihrer eigenen Kultur, im selben Prozess.

Die Erkennung ist nur ein Ausgangspunkt und keine Entscheidung, an der Sie hängen bleiben. Die Sprache eines Benutzers kann ebenso aus einer Profilzeile, einer Anmeldeeinstellung oder der Auswahl stammen, die Sie gleich bauen. Die Regel, auf die es ankommt, ist unabhängig von der Quelle dieselbe: Lösen Sie sie einmal beim Start der Sitzung auf und laden Sie sie dort in `Application.CurrentCulture`. Code, der später Ressourcen liest oder einen Wert formatiert, sollte nie erneut fragen müssen, woher die Sprache kam.

Die Einstellung culture in Default.json konfigurieren

`Default.json` ist der Ort, an dem die Anwendung ihre Absicht ausspricht. Die Einstellung `culture` steht standardmäßig auf `auto`, also auf dem eben beschriebenen Erkennungsverhalten, und die benachbarte Einstellung `rightToLeft` verhält sich genauso — `auto` folgt der Kultur. Im Lab belassen Sie `culture` auf `auto`, damit GlobalDesk beim Browser beginnt, und belegen es dann, indem Sie die bevorzugte Sprache Ihres Browsers ändern und neu laden.

Ein fester Wert ist kein Fehler, sondern eine andere Absicht. Die Einstellung auf einen Kulturnamen festzulegen ist die ehrliche Wahl für ein Produkt, das bewusst in einem einzigen Gebietsschema ausgeliefert wird, und es ist der schnellste Weg, beim Debuggen eine Meldung zu einer Sprache zu reproduzieren: Jede Sitzung landet in dieser Kultur, egal was der Browser des Testers sagt. Halten Sie fest, welche der beiden Varianten Sie gewählt haben und warum, denn der nächste Entwickler wird diese Einstellung als Aussage über das Produkt lesen.

CurrentCulture gilt pro Sitzung, niemals als statisches Feld

Die Sprache zur Laufzeit zu wechseln ist eine einzige Zuweisung: Setzen Sie `Application.CurrentCulture` auf die vom Benutzer gewählte Kultur, erzeugt mit `CultureInfo.GetCultureInfo` aus einem Namen wie `"en-US"`, `"de-DE"` oder `"fr-CA"`. Weil die Eigenschaft auf die Wisej.NET-Sitzung begrenzt ist, verändert diese Zuweisung die Welt eines einzigen Browser-Tabs und sonst nichts. Die Sprach-ComboBox im Lab tut genau das in ihrem `SelectedIndexChanged`-Handler und legt die Kulturnamen als Werte der Einträge ab, sodass der Handler nie eine Anzeigebeschriftung zerlegen muss.

Die verlockende Abkürzung ist eine `static CultureInfo` an einer bequemen Stelle, „damit alle sie sehen können“. Statische Felder leben im Serverprozess, und ein Wisej.NET-Server beherbergt jede Sitzung in diesem einen Prozess, deshalb hielte das Feld genau den Wert, den der zuletzt aktive Benutzer gewählt hat. Ein Kollege in einem anderen Land würde zusehen, wie sein Bildschirm die Sprache wechselt, weil Sie Deutsch getestet haben. Wenn mehrere Klassen die aktive Kultur brauchen, lesen Sie sie jedes Mal aus `Application.CurrentCulture`; das ist bereits der sitzungsbezogene Speicher, den Sie gerade neu erfinden wollten.

Der lang-Parameter und das Ereignis CultureChanged

Zwei Wisej.NET-Funktionen rahmen den Wechsel ein. Beim Einstieg fragt der URL-Parameter `lang` nach einer Sprache, bevor irgendein Code von Ihnen läuft, sodass `?lang=de` GlobalDesk auf Deutsch öffnet, ohne die Browsereinstellungen anzutasten — ideal für einen Testlink, eine Fehlermeldung oder eine sprachspezifische Einstiegs-URL, die Sie einem Kunden geben. Unterwegs wird `Application.CultureChanged` ausgelöst, wenn die aktuelle Kultur von der Anfangskultur der Sitzung abweicht — Ihr Aufhänger für alles, was bereits auf dem Bildschirm steht.

Abonnieren Sie einmal beim Erstellen der Seite und erledigen Sie die Arbeit in einem einzigen Handler, damit nichts davon abhängt, welches Control den Wechsel ausgelöst hat:

```

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

private void Application_CultureChanged(object sender, EventArgs e)
{
    ApplyTextResources(); // every resource-backed caption
    UpdateCulturePreview(); // date, number and currency samples
    RecreateCustomerEditor(); // designer-localized: rebuild it
}

```

Nachziehen, was der Wechsel zurückgelassen hat

Ein Kulturwechsel reist nicht rückwärts durch Beschriftungen, die Sie bereits zugewiesen haben.

`ApplyTextResources` geht die Controls durch, deren Text aus der Ressourcenschicht stammt, und fragt `Texts.Get` erneut nach jedem Schlüssel, sodass die Überschrift, die Schaltfläche Customers und jede Meldung ihre neuen Werte lesen. `UpdateCulturePreview` ist ihr Zwilling für die Formatierung: Die gespeicherten Daten haben sich überhaupt nicht geändert, deshalb nimmt sie dasselbe Datum, dieselbe Menge und denselben Dezimalbetrag und gibt sie mit der nun aktiven Kultur aus — und genau deshalb behalten Sie den Wert und nie die formatierte Zeichenfolge.

Der im Designer lokalisierte Kundeneditor ist der Teil, der die Leute erwischt. Seine lokalisierten Eigenschaftswerte werden beim Erstellen seiner Controls angewendet, deshalb behält er die Beschriftungen, mit denen er geboren wurde, ganz gleich, wie oft die Kultur danach wechselt. Die Lösung ist strukturell, nicht raffiniert: den Editor verwerfen und eine frische Instanz einfügen, ihn schließen und wieder öffnen oder zu einer neu erstellten Seite navigieren. Was immer Sie wählen: Tun Sie es im selben `CultureChanged` -Handler und sehen Sie sich dann den ganzen Bildschirm an — ein Wechsel, der das halbe Fenster in der vorherigen Sprache lässt, ist schlimmer als gar kein Wechsel, weil der Benutzer nun beiden Hälften misstraut.



Gewohnheit für die Lokalisierungsschichten

Entscheiden Sie, wo jede Zeichenfolge lebt, bevor Sie sie tippen. Designer-Ressourcen gehören zu einem entworfenen Bildschirm und werden beim Erstellen seiner Controls angewendet; gemeinsame Anwendungsressourcen in `.resx` tragen den Text, den Code erzeugt oder den mehrere Bildschirme wiederverwenden; `Resources-[LANG].resx` überschreibt die Systembeschriftungen von Wisej.NET. Speichern Sie niemals ein formatiertes Datum, eine formatierte Zahl oder einen formatierten Währungsbetrag: Behalten Sie den Wert und formatieren Sie ihn zum Anzeigzeitpunkt für die aktive Kultur. Und denken Sie daran, dass die Kultur pro Sitzung gilt, sodass die Sprache eines Benutzers niemals den Bildschirm eines anderen verändern darf.



Lernziele

Am Ende dieses Moduls können Sie:

- ✓ die automatische Erkennung der Browsersprache in Wisej.NET erklären.
- ✓ culture="auto" oder eine feste Kultur in Default.json konfigurieren.
- ✓ die Sitzungskultur mit Application.CurrentCulture wechseln.
- ✓ den URL-Parameter lang verwenden und Application.CultureChanged behandeln.
- ✓ ressourcengestützte Oberflächen nach einem Sprachwechsel zur Laufzeit erneut anwenden.

Lab- und Prüfungsleitfaden

Was Sie im praktischen Lab bauen, das vorgeschlagene Vorgehen und die geforderten Ergebnisse.



Praktisches Lab

Geben Sie GlobalDesk einen Sprachwechsel, der die Sache zu Ende bringt. Setzen Sie `culture` in `Default.json` auf `auto`, damit die Sitzung mit der Browsersprache beginnt, fügen Sie eine Sprach-ComboBox mit Englisch, Deutsch und einer Sprach-Regions-Kultur hinzu und setzen Sie `Application.CurrentCulture` in deren `SelectedIndexChanged`-Handler aus `CultureInfo.GetCultureInfo`, ohne die Auswahl jemals in einem statischen Feld abzulegen. Behandeln Sie `Application.CultureChanged`, um eine Methode `ApplyTextResources` aufzurufen, die jede ressourcengestützte Beschriftung auffrischt, und eine Methode `UpdateCulturePreview`, die die Datums-, Zahlen- und Währungsbeispiele neu formatiert, und bauen Sie im selben Handler den im Designer lokalisierten Kundeneditor neu auf. Prüfen Sie dieselbe Seite über den URL-Parameter `lang` und schreiben Sie dann eine Lab-Notiz darüber, was ein statisches Kulturfeld bei einer zweiten, gleichzeitig angemeldeten Person anrichten würde.

Ausgangspunkt

Öffnen Sie GlobalDesk so, wie Sie es nach Modul 3 hinterlassen haben: Das Dashboard liest jede Beschriftung über `Texts.Get`, `Resources/Strings.resx` und `Strings.de.resx` tragen dieselben Schlüssel, der Vorschaubereich für die Kultur formatiert ein Datum, eine Zahl und einen Betrag, und der Kundeneditor hat eine deutsche Designer-Variante. Auf dem Bildschirm gibt es noch nichts, womit ein Benutzer die Sprache wechseln könnte, und `Application.CultureChanged` hat keinen Abonnenten.

Vorgeschlagenes Vorgehen

- ✓ Öffnen Sie `Default.json`, vergewissern Sie sich, dass `culture` auf `auto` steht, und belegen Sie es: Stellen Sie die bevorzugte Sprache Ihres Browsers auf Deutsch, laden Sie GlobalDesk neu und notieren Sie, welche Beschriftungen übersetzt ankommen. Halten Sie in einer Zeile fest, warum Sie die Einstellung stattdessen auf einen festen Kulturnamen festlegen würden.
- ✓ Fügen Sie dem Dashboard eine ComboBox `languageCombo` mit drei Einträgen hinzu, deren Werte Kulturnamen sind — `en-US`, `de-DE` und eine Sprach-Regions-Kultur wie `fr-CA` — und deren Beschriftungen selbst Ressourcenschlüssel sind, keine fest verdrahteten Wörter.
- ✓ Weisen Sie in `languageCombo_SelectedIndexChanged` `Application.CurrentCulture = CultureInfo.GetCultureInfo(name)` zu und sonst nichts. Widerstehen Sie jeder Versuchung, die Auswahl in einem `static`-Feld zwischenspeichern; die Sitzung speichert sie bereits.
- ✓ Abonnieren Sie `Application.CultureChanged` im Konstruktor der Seite und schreiben Sie die beiden Methoden, die er aufruft: `ApplyTextResources` liest jede ressourcengestützte Beschriftung über `Texts.Get` neu ein, und `UpdateCulturePreview` formatiert dasselbe gespeicherte Datum, dieselbe Menge und denselben Dezimalbetrag mit `Application.CurrentCulture` neu.
- ✓ Verwerfen Sie im selben Handler den im Designer lokalisierten Kundeneditor und fügen Sie seinem Host-Container eine frische Instanz hinzu; wechseln Sie dann bei geöffnetem Editor die Sprache und prüfen Sie jede Beschriftung auf dem Bildschirm. Schließen Sie ab, indem Sie ?

`lang=de` in einer zweiten Browsersitzung öffnen und Ihre Lab-Notiz zum statischen Kulturzustand ergänzen.

Geforderte Ergebnisse

- ✓ Default.json mit culture auf auto, dazu eine Notiz, wann eine feste Kultur die richtige Wahl ist
- ✓ Handler der Sprach-ComboBox, der Application.CurrentCulture aus CultureInfo.GetCultureInfo setzt
- ✓ Handler für Application.CultureChanged, der ApplyTextResources und UpdateCulturePreview aufruft
- ✓ Im Designer lokalisierter Editor, der nach dem Wechsel neu aufgebaut wird, damit keine Beschriftung in der alten Sprache stehen bleibt
- ✓ Prüfung über den URL-Parameter lang und eine Notiz, warum die Kultur nicht in einem statischen Feld leben darf

Wiederholungsfragen

- ✓ Ihre Methode `ApplyTextResources` frischt das Dashboard einwandfrei auf, der Kundeneditor behält nach einem Wechsel zu Deutsch aber seine englischen Beschriftungen. Erklären Sie, was an diesen Beschriftungen anders ist, und nennen Sie zwei Strategien, die das beheben.
- ✓ Ein Kollege schlägt vor, die gewählte `CultureInfo` in einem statischen Feld zwischenzuspeichern, „um sie nicht zweimal nachschlagen zu müssen“. Beschreiben Sie, was ein zweiter Benutzer, im selben Moment aus einem anderen Land angemeldet, tatsächlich sehen würde.
- ✓ Die Kulturvorschau zeigt einen Währungsbetrag. Was hat sich nach einem Wechsel von `en-US` zu `de-DE` im Speicher geändert und was nur auf dem Bildschirm — und welches von beidem würden Sie je in die Datenbank schreiben?



Gewohnheit für die Lokalisierungsschichten

Entscheiden Sie, wo jede Zeichenfolge lebt, bevor Sie sie tippen. Designer-Ressourcen gehören zu einem entworfenen Bildschirm und werden beim Erstellen seiner Controls angewendet; gemeinsame Anwendungsressourcen in .resx tragen den Text, den Code erzeugt oder den mehrere Bildschirme wiederverwenden; Resources-[LANG].resx überschreibt die Systembeschriftungen von Wisej.NET. Speichern Sie niemals ein formatiertes Datum, eine formatierte Zahl oder einen formatierten Währungsbetrag: Behalten Sie den Wert und formatieren Sie ihn zum Anzeigzeitpunkt für die aktive Kultur. Und denken Sie daran, dass die Kultur pro Sitzung gilt, sodass die Sprache eines Benutzers niemals den Bildschirm eines anderen verändern darf.