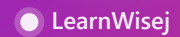


Grundlagen der Lokalisierung: Kultur, Sprache und Ressourcen

Lektionsleitfaden · Lab- und Prüfungsleitfaden



Lektionsleitfaden

Was Lokalisierung über die reine Übersetzung hinaus tatsächlich umfasst: Kulturnamen wie en, de-DE und fr-CA, .resx-Ressourcendateien, die über den ResourceManager gelesen werden, der .NET-Kulturrückfall auf die neutrale Ressource, stabile semantische Schlüssel statt englischer Sätze im Code und die kultursensible Formatierung von Datum, Zahlen und Währung. 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

Was Lokalisierung über die reine Übersetzung hinaus tatsächlich umfasst: Kulturnamen wie en, de-DE und fr-CA, .resx-Ressourcendateien, die über den ResourceManager gelesen werden, der .NET-Kulturrückfall auf die neutrale Ressource, stabile semantische Schlüssel statt englischer Sätze im Code und die kultursensible Formatierung von Datum, Zahlen und Währung.

1. Das Problem im Produktivbetrieb

GlobalDesk ist ein kleiner Service Desk, geschrieben für ein einziges Büro. Jede Überschrift, jede Schaltflächenbeschriftung und jede Bestätigungsmeldung ist ein englisches Literal, direkt in den C#-Code getippt, und das Dashboard gibt Beträge und Datumsangaben so aus, wie eine US-Maschine es zufällig tut. Wenn eine deutsche Niederlassung eröffnet, hat dieser Text nirgendwohin zu gehen — und wer damit beginnt, Beschriftungen von Hand zu ändern, trifft auf die zweite Hälfte des Problems: Eine deutsche Beschriftung über einem Betrag, der weiterhin **1,850.75** lautet, ergibt noch keinen deutschen Bildschirm.

2. Zentrale Konzepte

Konzept	Warum es wichtig ist
Lokalisierung	Eine Anwendung an Sprache <i>und</i> Kultur des Benutzers anpassen: übersetzter Text, korrekt formatierte Datumsangaben, Zahlen und Währungen, Platz für längere Wörter und ein Layout, das sich für Rechts-nach-links-Sprachen spiegeln lässt.
Kulturname	Ein Bezeichner wie <code>en</code> , <code>de-DE</code> oder <code>fr-CA</code> , der eine Sprache und optional eine Region benennt; der Sprachteil wählt, welche Ressourcen bevorzugt werden, der vollständige Name steuert die Formatierungskonventionen.
neutrale Ressource	Der vollständige Rückfallsatz an Ressourcen, abgelegt in einer Datei wie <code>Resources/Strings.resx</code> ohne Kultur im Namen; jeder Schlüssel, nach dem die Anwendung fragt, muss hier vorhanden sein.
<code>Wisej.Resources.ResourceManager</code>	Das Objekt, das einen Ressourcen-Basisnamen und eine Assembly entgegennimmt und den Wert für einen Schlüssel in der am besten passenden Kultur zurückgibt, sodass Ihr Code nie selbst eine Datei öffnet.
Kulturrückfall	Die .NET-Regel zum Auffinden eines Werts: erst die vollständige Kultur versuchen, dann die reine Sprachkultur, dann die neutrale Ressource — und den ersten Treffer nehmen, der den Schlüssel hat.
semantischer Ressourcenschlüssel	Ein stabiler Name, der die Bedeutung beschreibt statt der Formulierung, etwa <code>CustomerEditor.Save</code> oder <code>Validation.Required</code> , sodass der englische Text umformuliert werden kann, ohne eine andere Sprache anzufassen.
<code>Application.CurrentCulture</code>	Die Kultur der aktuellen Wisej.NET-Sitzung; zwei gleichzeitig angemeldete Benutzer können zwei verschiedene Werte halten, und genau deshalb darf die Kultur niemals in einem statischen Feld leben.

3. Der rote Faden der Lektion

Übersetzen ist nur die halbe Arbeit

Im Wort „Lokalisierung“ verstecken sich zwei verschiedene Aufgaben, und Einsteiger sehen meist nur die erste. Die erste Aufgabe ist die **Lokalisierung von Oberflächentext**: die Wörter, die der Benutzer liest, etwa die Willkommensüberschrift von GlobalDesk, die Navigationsbeschriftung Customers, die Schaltfläche Save und die Meldung nach dem Speichern eines Kunden. Diese Wörter stammen von Ihnen, sie leben in Ressourcendateien, und die Sprache der Sitzung entscheidet, welche Fassung zurückkommt.

Die zweite Aufgabe ist die **kultursensible Formatierung**. Ein Fälligkeitsdatum, eine Ticketanzahl und eine Auftragssumme sind überhaupt keine Wörter; es sind Werte. Derselbe Betrag wird unter `en-US` als `$1,850.75` geschrieben und unter `de-DE` als `1.850,75 €` , und keine dieser Zeichenfolgen gehört in eine Ressourcendatei oder in eine Datenbankspalte. Sie speichern den `decimal` - und den `DateTime` -

Wert und formatieren sie in dem Moment, in dem Sie sie auf den Bildschirm bringen, und übergeben dabei die Kultur der Sitzung als Formatanbieter.

Wegen dieser Trennung endet das Lab mit einer schriftlichen Notiz. Sobald Sie auf etwas zeigen und sagen können „das sind Daten“ oder „das ist Darstellung“, hat der Rest des Kurses einen Halt. Bauen Sie eine Zeichenfolge wie "€ " + amount , und Sie haben stillschweigend ein Währungssymbol für jedes Land fest verdrahtet, in das Sie je ausliefern werden.

Einen Kulturnamen lesen: en, de-DE, fr-CA

Ein Kulturname verbindet eine Sprache mit regionalen Konventionen. Ein **reiner Sprachname** (neutral) ist nur die Sprache: en , de , fr . Ein **Sprach-Regions-Name** ergänzt ein Land: en-US , de-DE , fr-CA , it-IT . In .NET sind das CultureInfo -Werte, und Wisej.NET stellt denjenigen, der für die aktuelle Sitzung gilt, als Application.CurrentCulture bereit.

Zwei Namen können sich über die Sprache einig sein und über alles andere nicht. Eine Sitzung unter en-GB und eine Sitzung unter en-US lesen beide Ihre englischen Ressourcen, weil beide die Sprache en auswählen; trotzdem schreiben sie Datumsangaben unterschiedlich und verwenden verschiedene Währungssymbole. Auch der umgekehrte Fall kommt vor: fr-CA und fr-FR teilen sich den französischen Text, aber nicht die Zahlen- und Währungskonventionen. Der Sprachteil des Namens beantwortet also „welche Wörter?“ und der vollständige Name „wie werden Werte geschrieben?“.

Für GlobalDesk zählt das sofort. Der Vorschaubereich für die Kultur, den Sie im Lab bauen, zeigt das heutige Datum, eine Menge und einen Währungsbetrag nebeneinander, sodass Sie drei Werte wechseln sehen, während die Beschriftungen darüber in einer Sprache bleiben.

.resx-Dateien und ResourceManager

Eine .resx -Datei ist eine XML-Ressourcendatei, die Visual Studio als Schlüssel-Wert-Raster bearbeitet. Wisej.NET braucht kein eigenes Übersetzungsformat: Es nutzt das Standardmodell von .NET, deshalb funktionieren .resx -Dateien, CultureInfo , Satellitenassemblys und RESX-Werkzeuge von Drittanbietern unverändert. Sie fügen Resources/Strings.resx als neutrale Datei hinzu und später Strings.de.resx daneben, mit denselben Schlüsseln und deutschen Werten.

Ihr Code öffnet diese Dateien nie. Er fragt einen ResourceManager , erzeugt aus dem Ressourcen-Basisnamen und seiner Assembly, nach einem Schlüssel. Das in einen winzigen statischen Helper zu kapseln gibt Ihnen eine einzige Stelle, an der Sie entscheiden, wie ein fehlender Schlüssel aussieht:

```
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}]";
}
```

Die Schlüssel sind die andere Hälfte der Disziplin. Verwenden Sie `CustomerEditor.Save`, `Navigation.Customers` oder `Validation.Required`: Namen, die Bedeutung beschreiben, nicht Formulierung. Den englischen Satz als Schlüssel zu nehmen bedeutet, dass jede Umformulierung jede Übersetzungsdatei zerstört. Und lokalisieren Sie ausschließlich Produkttext — Beschriftungen, Menüs, Tooltips, Validierungs- und Bestätigungsmeldungen. Protokollbezeichner, Namen von Ausnahmetypen und Datenbankspaltennamen bleiben, wie sie sind.

Rückfall: drei Versuche, dann die neutrale Datei

Wenn eine Sitzung unter `fr-CA` nach einem Schlüssel fragt, gibt .NET nicht auf, falls `Strings.fr-CA.resx` nichts dazu hat. Es versucht zuerst die spezifischste Ressource, dann die allgemeinere `fr`, dann die neutrale Ressource und gibt den ersten gefundenen Wert zurück. Deshalb muss die neutrale Datei vollständig sein: Sie ist die letzte Stelle, an der die Suche nachsieht, und ein dort fehlender Schlüssel fehlt überall.

Der Rückfall ist ein Sicherheitsnetz, keine Übersetzungsstrategie. Er verhindert, dass eine halbfertige deutsche Datei einen Bildschirm leert, aber er verbirgt auch Lücken: Der Benutzer sieht Englisch, wo Deutsch erwartet wurde, und meldet es vielleicht nie. Lassen Sie den Rückfall also aktiv und finden Sie die Lücken gezielt — genau dafür ist die Klammermarkierung im `Texts`-Helper da. Eine Beschriftung, die als `[Dashboard.Welcome]` erscheint, ist ein sichtbarer Fehler.

Das GlobalDesk-Dashboard vorbereiten

Eine Seite für die Lokalisierung vorzubereiten ist mechanisch, sobald die Entscheidungen gefallen sind. Bauen Sie das Dashboard ganz normal, gehen Sie es dann durch und fragen Sie zu jeder sichtbaren Zeichenfolge: Wer liest das? Die Willkommensüberschrift, die Schaltfläche Customers und jedes Vorschau-Label werden von Benutzern gelesen, also bekommt jedes einen Schlüssel in `Resources/Strings.resx` und einen `Texts.Get`-Aufruf anstelle seines Literals. Zwölf Schlüssel sind ein realistischer Anfang für einen Bildschirm.

Dann erledigen Sie die andere Hälfte. Der Vorschaubereich darf keinen Text ausgeben, den Sie selbst zusammengesetzt haben; er formatiert echte Werte und übergibt `Application.CurrentCulture` an `ToString`, damit Datumsmuster, Trennzeichen und Währungssymbol für die Sitzung gewählt werden. Diese Kultur gilt pro Sitzung, deshalb können zwei Benutzer im selben Serverprozess zwei Sprachen lesen — und ein statisches Kulturfeld würde einen von ihnen den Bildschirm des anderen verändern lassen.



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:

- ✓ den Unterschied zwischen der Lokalisierung von Oberflächentext und kultursensibler Formatierung erklären.
- ✓ neutrale, reine Sprach- und Sprach-Regions-Kulturnamen wie en, de-DE und fr-CA erkennen.
- ✓ erklären, wie .resx-Dateien und ResourceManager an der .NET-Lokalisierung mitwirken.
- ✓ den .NET-Kulturrückfall in Einsteigerworten beschreiben.
- ✓ eine kleine Wisej.NET-Seite bauen, in der alle für den Benutzer sichtbaren Zeichenfolgen für die Lokalisierung vorbereitet sind.

Lab- und Prüfungsleitfaden

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



Praktisches Lab

Erstellen Sie die GlobalDesk-Dashboard-Seite mit einer Willkommensüberschrift, einer Navigationsschaltfläche Customers und einem kleinen Vorschaubereich für die Kultur. Fügen Sie Resources/Strings.resx als neutrale Ressource mit mindestens zwölf stabilen semantischen Schlüsseln wie Navigation.Customers und CustomerEditor.Save hinzu, schreiben Sie einen statischen Texts-Helper über Wisej.Resources.ResourceManager, der bei einem fehlenden Schlüssel eine in Klammern gesetzte Markierung zurückgibt, und ersetzen Sie jede fest verdrahtete Beschriftung auf der Seite durch einen Texts.Get-Aufruf. Füllen Sie anschließend den Vorschaubereich mit echten Werten: Formatieren Sie das heutige Datum, eine Menge und einen Währungsbetrag mit Application.CurrentCulture, statt die Zeichenfolgen von Hand zusammenzusetzen, und schließen Sie mit einer kurzen Lab-Notiz ab, die den gespeicherten Wert von seiner kultursensiblen Darstellung trennt.

Ausgangspunkt

Öffnen Sie das Wisej.NET-Projekt GlobalDesk in Visual Studio. Es ist ein kleiner Service Desk mit einer Hauptseite, einem Rumpf des Kundeneditors und einem Kontakt-Grid, und derzeit ist darin jedes für den Benutzer sichtbare Wort ein englisches Literal im C#-Code, während Beträge und Datumsangaben so ausgegeben werden, wie der Server sie gerade formatiert. In diesem Lab fügen Sie die Dashboard-Seite hinzu und geben der Anwendung ihre erste Lokalisierungsschicht: Ressourcenschlüssel für den Text und kulturbewusste Formatierung für die Werte.

Vorgeschlagenes Vorgehen

- ✓ Fügen Sie eine `Dashboard`-Page hinzu und legen Sie eine Willkommensüberschrift, eine Navigationsschaltfläche `btnCustomers` und einen kleinen Vorschaubereich für die Kultur mit drei schreibgeschützten Labels für ein Datum, eine Menge und einen Betrag an. Bauen Sie das zuerst auf Englisch, genau so, wie Sie es sonst tun würden.
- ✓ Fügen Sie `Resources/Strings.resx` als neutrale Ressource hinzu und füllen Sie sie mit mindestens zwölf semantischen Schlüsseln wie `Dashboard.Welcome`, `Navigation.Customers`, `CustomerEditor.Save` und `Preview.Amount`. Die englischen Werte stehen hier und sonst nirgends.
- ✓ Fügen Sie eine statische Klasse `Texts` hinzu, die einen `Wisej.Resources.ResourceManager` aus dem Ressourcen-Basisnamen und der eigenen Assembly des Helpers hält, sowie eine Methode `Get(string key)`, die den Wert oder eine in Klammern gesetzte Markierung `[key]` zurückgibt, wenn die Suche nichts findet.
- ✓ Gehen Sie die Seite durch und ersetzen Sie jede fest verdrahtete Beschriftung durch einen `Texts.Get`-Aufruf. Fragen Sie dann absichtlich nach einem Schlüssel, den Sie nie angelegt haben, und bestätigen Sie, dass die Klammermarkierung auf dem Bildschirm erscheint statt eines leeren Labels.

- ✓ Füllen Sie den Vorschaubereich mit echten Werten: Rufen Sie `ToString` auf `DateTime.Today`, auf eine Menge und auf einen `decimal`-Betrag auf und übergeben Sie jedes Mal `Application.CurrentCulture`. Setzen Sie niemals selbst ein Symbol oder ein Trennzeichen zusammen und schreiben Sie anschließend die Lab-Notiz.

Geforderte Ergebnisse

- ✓ GlobalDesk-Dashboard-Seite, deren Beschriftungen alle aus Ressourcenschlüsseln stammen, ohne fest verdrahteten sichtbaren Text
- ✓ Neutrale Datei `Resources/Strings.resx` mit mindestens zwölf stabilen semantischen Schlüsseln
- ✓ Texts-Helper über `Wisej.Resources.ResourceManager`, der einen fehlenden Schlüssel markiert, statt eine leere Zeichenfolge zurückzugeben
- ✓ Vorschaubereich für die Kultur, der ein Datum, eine Zahl und einen Währungsbetrag mit `Application.CurrentCulture` formatiert
- ✓ Lab-Notiz, die erklärt, welche Werte Daten sind und welche kultursensible Darstellung

Wiederholungsfragen

- ✓ Sie ändern die englische Formulierung der Schaltfläche `Save` von „Save“ zu „Save customer“. Welche Dateien müssen sich ändern, wenn Ihr Schlüssel `CustomerEditor.Save` lautet, und welche hätten sich ändern müssen, wenn der Schlüssel der englische Satz selbst gewesen wäre?
- ✓ Ihr `Texts`-Helper gibt für ein Label `[Preview.Amount]` zurück. Nennen Sie zwei verschiedene Fehler, die genau diese Markierung erzeugen, und sagen Sie, wie Sie sie ohne Debugger auseinanderhalten würden.
- ✓ Ein Kollege schlägt vor, die Auftragssumme als Zeichenfolge „1.850,75 €“ in der Datenbank zu speichern, damit deutsche Benutzer sie immer korrekt sehen. Erklären Sie, was dabei kaputtgeht, und welcher Teil dieser Zeichenfolge Daten sind und welcher Darstellung.



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.