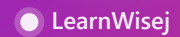


Anwendungsressourcen, ResourceManager und Wisej-Systemtexte

Lektionsleitfaden · Lab- und Prüfungsleitfaden



Lektionsleitfaden

Die gemeinsame Ressourcenschicht: neutrale und kulturspezifische .resx-Dateien mit identischen Schlüsseln, die Suche über Wisej.Resources.ResourceManager nach Basisname und Assembly, Platzhalter, die mit string.Format und der aktiven Kultur formatiert werden, Services, die Schlüssel statt englischer Sätze zurückgeben, und Resources-[LANG].resx für die Systembeschriftungen von Wisej. 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

Die gemeinsame Ressourcenschicht: neutrale und kulturspezifische .resx-Dateien mit identischen Schlüsseln, die Suche über Wisej.Resources.ResourceManager nach Basisname und Assembly, Platzhalter, die mit string.Format und der aktiven Kultur formatiert werden, Services, die Schlüssel statt englischer Sätze zurückgeben, und Resources-[LANG].resx für die Systembeschriftungen von Wisej.

1. Das Problem im Produktivbetrieb

Modul 2 hat den GlobalDesk-Kundeneditor auf Deutsch überzeugend hinterlassen: Jede Beschriftung, die der Designer besitzt, wechselt mit der Kultur, und das Layout hält stand. Dann drückt der Benutzer Save und liest *Customer saved.* auf Englisch, denn dieser Satz ist keine Eigenschaft irgendeines Controls — er ist ein Zeichenfolgenliteral in einem Click-Handler, gleich neben *This field is required.* und der Statuszeile, die die geladenen Kontakte zählt.

Die Lokalisierung im Designer kann diese Zeichenfolgen nie erreichen: Sie werden im Code erzeugt, von mehreren Bildschirmen geteilt oder von einem Service hervorgebracht, der nichts von einer Benutzeroberfläche weiß. Dieses Modul gibt GlobalDesk eine zweite Ressourcenschicht für genau diesen Text und geht dann eine Ebene tiefer: zu den Beschriftungen, die Wisej.NET selbst auf einer `MessageBox` oder einem Kalender zeichnet.

2. Zentrale Konzepte

Konzept	Warum es wichtig ist
gemeinsame Anwendungsressourcendatei	<code>Resources/Strings.resx</code> enthält den für den Benutzer sichtbaren Text, den kein einzelner entworfener Bildschirm besitzt: Speicherbestätigungen, Validierungsmeldungen, Statuszeilen, Berichtsüberschriften und Serviceergebnisse.
kulturspezifische Geschwisterdatei	<code>Strings.de.resx</code> und <code>Strings.it.resx</code> liegen neben der neutralen Datei und tragen dieselben Schlüssel ; ein Übersetzer ändert Werte, niemals Schlüsselnamen.
stabiler semantischer Schlüssel	Ein Name wie <code>CustomerEditor.Saved</code> beschreibt die Bedeutung des Textes, sodass eine Umformulierung des englischen Satzes nie eine Codeänderung erzwingt oder die Übersetzungen zerstört.
<code>Wisej.Resources.ResourceManager</code>	Das Suchobjekt, das dem .NET-Ressourcenmodell folgt: Mit Basisname und Assembly findet es den Wert für die aktive Kultur und fällt in Richtung neutraler Datei zurück.
Markierung für fehlende Schlüssel	Ein Helper, der bei einer fehlgeschlagenen Suche <code>[Key]</code> zurückgibt, macht aus einem unsichtbaren Defekt etwas, das Sie sehen und suchen können, statt eines leeren Labels.
Formatzeichenfolge mit Platzhaltern	Der gesamte Satz mit seinen <code>{0}</code> und <code>{1}</code> lebt in der Ressource und <code>string.Format</code> füllt ihn, denn Sprachen ordnen Name, Datum und Verb unterschiedlich an.
<code>Resources-[LANG].resx</code>	Eine separate Wisej.NET-Systemressourcendatei im Projektstamm oder in einem Ordner <code>/Resources</code> , die Framework-Beschriftungen wie <code>\$Ok</code> , <code>\$Cancel</code> und <code>\$Offline</code> überschreibt oder ergänzt.

3. Der rote Faden der Lektion

Zwei Dateien, ein Satz Schlüssel

Beginnen Sie mit der neutralen Datei. In GlobalDesk ist das `Resources/Strings.resx`, und sie trägt bereits die Schlüssel, die Modul 1 eingeführt hat: `Navigation.Customers`, `CustomerEditor.Save`, `CustomerEditor.Cancel`, `CustomerEditor.Saved`, `Validation.Required`. Die neutrale Datei ist diejenige, auf die .NET zurückfällt, wenn nichts Besseres existiert, deshalb darf sie nie unvollständig sein. Ein hier fehlender Schlüssel ist ein Entwicklerfehler, keine Übersetzungslücke.

Eine übersetzte Datei trägt denselben Dateinamen mit einem Kulturnamen vor der Erweiterung: `Strings.de.resx` für Deutsch, `Strings.it.resx` für Italienisch. Nur die **Wertespalte** unterscheidet sich. Benennen Sie `CustomerEditor.Saved` in der deutschen Datei um, und er beantwortet die Suche nicht mehr, sodass der Bildschirm still auf Englisch zurückfällt. Die Schlüsselliste ist ein Vertrag: Einen Schlüssel hinzuzufügen heißt, ihn überall hinzuzufügen.

Eine deutsche Datei bedient `de`, `de-DE` und `de-AT` gleichermaßen: .NET geht von der spezifischsten Kultur in Richtung neutrale Ressource und nimmt den ersten Treffer. Fügen Sie `Strings.de-AT.resx` nur hinzu, wenn die österreichische Formulierung sich wirklich unterscheidet.

Text über den ResourceManager nachschlagen

Wisej.NET stellt `Wisej.Resources.ResourceManager` bereit, der dem .NET-Ressourcenmodell folgt und weiß, wie kulturspezifische Ressourcen zu finden sind. Sie erzeugen ihn aus einem Basisnamen und der Assembly, die die Ressourcen enthält; der Basisname ist der Projekt-Namespace plus Ordner- und Dateiname, für GlobalDesk also `GlobalDesk.Resources.Strings`. Ihn falsch zu setzen ist der häufigste Einsteigerfehler, und er zeigt sich darin, dass auf einen Schlag jeder Schlüssel als fehlend zurückkommt.

Kapseln Sie den Manager in einen kleinen statischen Helfer, damit der Rest der Anwendung ihn nie direkt anfasst und eine einzige Methode entscheidet, was ein fehlender Schlüssel bewirkt:

```
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 Klammermarkierung ist ein guter Standard für die Entwicklung: Eine leere Beschriftung sieht aus wie ein Layoutfehler, während `[CustomerEditor.Saved]` Ihnen genau sagt, welcher Schlüssel wo fehlt. Im Produktivbetrieb protokollieren Sie den fehlenden Schlüssel womöglich lieber, woraus Modul 7 eine Richtlinie macht. Was Sie niemals tun dürfen, ist, eine fehlende Ressource zu stillem Leertext werden zu lassen.

Designer-Ressourcen, gemeinsame Ressourcen und wohin welche Zeichenfolge gehört

Zwei Ressourcenmechanismen leben nun im selben Projekt, und sie beantworten verschiedene Fragen. Designer-Ressourcen gehören zu einer entworfenen Fläche, werden in die eigenen `.resx`-Dateien dieses Containers geschrieben und beim Erstellen seiner Controls angewendet. Gemeinsame Anwendungsressourcen gehören zur Anwendung und werden bei Bedarf über `Texts.Get` gelesen, sind also so aktuell wie der Moment des Aufrufs.

Der praktische Test ist die Eigentümerschaft. Existiert eine Beschriftung, weil Sie einen `Button` auf dem Kundeneditor abgelegt haben, gehört sie dem Designer. Erscheint der Text in einer `MessageBox`, in einer Statuszeile, in einem Spaltenkopf eines Grids, den Sie im Code aufbauen, oder auf zwei verschiedenen Bildschirmen, gehört er in `Strings.resx`. Einen Satz in drei Designerdateien zu duplizieren ist der Weg, auf dem halb übersetzte Anwendungen entstehen: Die dritte Kopie ist immer die, die niemand aktualisiert.

Dieselbe Überlegung reicht bis in die Domäne. Ein Save-Service sollte nicht den Satz *Customer saved* zurückgeben; er sollte ein stabiles Ergebnis liefern, etwa den Schlüssel `CustomerEditor.Saved` oder einen Ergebniscode, den die Oberfläche auf einen Schlüssel abbildet. Die Geschäftslogik bleibt dann testbar, kein

Englisch sickert aus der Serviceschicht, und der Bildschirm bleibt die einzige Stelle, die weiß, wie man mit einem Benutzer spricht.

Die Beschriftungen von Wisej: Systemlokalisierung und Resources-[LANG].resx

Nicht jedes Wort auf dem Bildschirm ist Ihres. Wisej.NET lokalisiert seine eigenen Framework-Ressourcen — die Schaltflächen der `MessageBox`, Kalenderbeschriftungen und das kultursensible Verhalten von Zahlen-, Datums- und Währungseditoren — und liefert eingebaute Systemübersetzungen für Englisch, Deutsch, Französisch, Italienisch, Türkisch, Spanisch und Portugiesisch mit. Führen Sie GlobalDesk auf Deutsch aus, und die Ja/Nein-Schaltflächen wechseln von selbst.

Für eine Sprache, die Wisej.NET nicht abdeckt, oder für eine Formulierung, auf der Ihr Produkt besteht, unterstützt Wisej.NET eine Datei `Resources-[LANG].resx` im Projektstamm oder in einem Ordner `/Resources`. Ihre Schlüssel sind die des Frameworks und beginnen mit einem Dollarzeichen: `$Ok`, `$Cancel`, `$Yes`, `$No`, `$Retry`, die Navigationsbeschriftungen des Kalenders und `$Offline`. Halten Sie beides auseinander: `Strings.de.resx` ist, was GlobalDesk sagt, `Resources-de.resx` ist, was Wisej.NET sagt.

Meldungen, Validierung und Bilder unter der aktiven Kultur

Sätze, die Daten erwähnen, brauchen Platzhalter, und an Platzhaltern verlieren Einsteiger die Wortstellung. Legen Sie den vollständigen Satz in der Ressource ab — `Customer.LastOrder` enthält `{0} last ordered on {1}`. — und lassen Sie den Code die Werte liefern: `string.Format(Application.CurrentCulture, Texts.Get("Customer.LastOrder"), customerName, lastOrderDate)`. Die Kultur mitzugeben formatiert außerdem das Datum so, wie die Sitzung es erwartet. Verketteten Sie niemals `"Customer " + name + " saved"`: Ein Übersetzer kann kein Fragment verschieben, das der Code zusammengeklebt hat.

Platzhalter sind für jemanden, der nur die Wertespalte sieht, bedeutungslos; nutzen Sie deshalb das Kommentarfeld im `.resx`-Editor, um zu sagen, was `{0}` und `{1}` sind. Validierungsmeldungen verdienen dieselbe Sorgfalt: `Validation.Required` und `Validation.Email` werden von mehreren Bildschirmen wiederverwendet, und genau deshalb gehören sie in die gemeinsame Datei statt in jedes Formular.

Eine `.resx`-Datei kann mehr als Zeichenfolgen enthalten — Bilder und andere Daten sind legitime Einträge, und die Lokalisierung im Designer kann ein Bild je Sprache austauschen. Tun Sie es aber mit Absicht. Ein Bild, das sich nur unterscheidet, weil Text darin eingebrennt ist, ist eine Wartungsfalle; belassen Sie diesen Text als echten Text in der Oberfläche, wo `Texts.Get` ihn übersetzt und niemand etwas neu zeichnen muss.



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:

- ✓ neutrale und kulturspezifische .resx-Ressourcendateien erstellen.
- ✓ Werte über den ResourceManager mit stabilen Schlüsseln abrufen.
- ✓ wiederverwendbare Zeichenfolgen getrennt von Designer-Ressourcen organisieren.
- ✓ die eingebaute Systemlokalisierung von Wisej.NET und eigene Resources-[LANG].resx-Dateien erklären.
- ✓ kulturbewusste Ressourcen für Meldungen, Validierung und Bilder einsetzen.

Lab- und Prüfungsleitfaden

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



Praktisches Lab

Erweitern Sie die gemeinsamen Ressourcen von GlobalDesk. Legen Sie `Strings.de.resx` neben `Resources/Strings.resx` an, mit exakt denselben Schlüsseln und übersetzten Werten, fügen Sie einen Eintrag `Customer.LastOrder` mit zwei Platzhaltern hinzu und bauen Sie den Satz mit `string.Format` und `Application.CurrentCulture` auf, statt Fragmente aneinanderzuhängen. Leiten Sie jede Meldung, die der Kundeneditor anzeigt, über den Texts-Helper, einschließlich der Speicherbestätigung und der beiden Validierungsmeldungen, und lassen Sie den Save-Service einen stabilen Ergebnisschlüssel wie `CustomerEditor.Saved` zurückgeben statt eines englischen Satzes, den die Oberfläche direkt ausgibt. Fügen Sie anschließend eine Datei `Resources-de.resx` hinzu, die eine Systembeschriftung von Wisej überschreibt, weisen Sie das in einer `MessageBox` nach und schreiben Sie eine kurze Notiz zum Unterschied zwischen Ihrem Produkttext und dem Systemtext von Wisej.

Ausgangspunkt

Sie öffnen GlobalDesk so, wie Modul 2 es hinterlassen hat: Das Dashboard liest seine Beschriftungen über den `Texts`-Helper aus `Resources/Strings.resx`, und das UserControl `CustomerEditor` ist im Designer lokalisiert, mit einer deutschen Variante. Die Speicherbestätigung, die beiden Validierungsmeldungen und die Zeile zur letzten Bestellung sind weiterhin englische Literale im Code, und eine übersetzte gemeinsame Ressourcendatei gibt es noch nicht.

Vorgeschlagenes Vorgehen

- ✓ Legen Sie `Resources/Strings.de.resx` neben der neutralen Datei an, übernehmen Sie die Schlüsselliste unverändert und übersetzen Sie ausschließlich die Werte; führen Sie die Anwendung unter `de-DE` aus und bestätigen Sie, dass die Beschriftungen des Dashboards umspringen, bevor Sie Code anfassen.
- ✓ Fügen Sie einen Eintrag `Customer.LastOrder` hinzu, dessen Wert ein vollständiger Satz mit `{0}` und `{1}` ist, füllen Sie das Kommentarfeld so, dass ein Übersetzer weiß, dass der erste der Kundenname und der zweite das Bestelldatum ist, und bauen Sie die Zeile mit `string.Format(Application.CurrentCulture, Texts.Get("Customer.LastOrder"), customerName, lastOrderDate)` auf.
- ✓ Ersetzen Sie die Literale in `CustomerEditor` durch `Texts.Get`-Aufrufe für `CustomerEditor.Saved`, `Validation.Required` und `Validation.Email` und löschen Sie jede Zeichenfolgenverkettung, die Ihnen unterwegs begegnet.
- ✓ Ändern Sie den Save-Service so, dass er ein stabiles Ergebnis zurückgibt — den Schlüssel `CustomerEditor.Saved` oder einen Ergebnisscode — und lassen Sie den Editor dieses Ergebnis über `Texts.Get` auf Text abbilden, sodass in der Serviceschicht kein englischer Satz zurückbleibt.
- ✓ Fügen Sie `Resources-de.resx` im Projektstamm oder im Ordner `/Resources` hinzu, überschreiben Sie eine Systembeschriftung von Wisej.NET wie `$Ok` oder `$Cancel` und zeigen

Sie dann eine `MessageBox` auf Deutsch und vergleichen Sie ihre Schaltflächen mit dem englischen Lauf.

Geforderte Ergebnisse

- ✓ `Strings.de.resx` mit denselben Schlüsseln wie die neutrale Datei und ausschließlich übersetzten Werten
- ✓ Ressource `Customer.LastOrder`, mit `string.Format` und der aktiven Kultur aufgebaut, niemals zusammengehängt
- ✓ `Save-Service`, der einen Ressourcenschlüssel oder Ergebniscode statt eines englischen Satzes zurückgibt
- ✓ `Resources-de.resx`, die mindestens eine Systembeschriftung von Wisej überschreibt, gezeigt in einer `MessageBox`
- ✓ `Lab-Notiz`, die Produkttext, Wisej-Systemtext und Diagnosetext voneinander trennt

Wiederholungsfragen

- ✓ Löschen Sie einen Schlüssel aus `Strings.de.resx` und führen Sie die Anwendung auf Deutsch aus. Beschreiben Sie, was der Benutzer sieht, warum nichts abstürzt und wie Sie das in einer echten Anwendung mit zweihundert Schlüsseln bemerken würden.
- ✓ Ihr `Save-Service` gibt derzeit `CustomerEditor.Saved` zurück. Was ginge in den Unit-Tests und auf dem deutschen Bildschirm kaputt, wenn er stattdessen den Satz selbst zurückgäbe?
- ✓ Ein Kollege fügt das Wort „Cancel“ unter einem selbst erfundenen Schlüssel in `Resources-de.resx` ein. Erklären Sie, warum die Schaltfläche der `MessageBox` sich nicht ändert und wohin diese Zeichenfolge tatsächlich gehört hätte.



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.