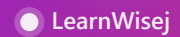


Die Wisej.NET-Oberfläche im Visual Studio Designer lokalisieren

Lektionsleitfaden · Lab- und Prüfungsleitfaden



Lektionsleitfaden

Lokalisierung im Designer auf einem Wisej.NET-Container der obersten Ebene: die Eigenschaft Localizable, der Language-Selektor des Designers, die Eigenschaften, die sich je Sprache legitim unterscheiden dürfen – etwa Text, Size, Location, Farben und Bilder –, die Ressourcendateien, die der Designer erzeugt, und die Regel, dass Designer-Ressourcen beim Erstellen der Controls angewendet werden. 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

Lokalisierung im Designer auf einem Wisej.NET-Container der obersten Ebene: die Eigenschaft Localizable, der Language-Selektor des Designers, die Eigenschaften, die sich je Sprache legitim unterscheiden dürfen, etwa Text, Size, Location, Farben und Bilder, die Ressourcendateien, die der Designer erzeugt, und die Regel, dass Designer-Ressourcen beim Erstellen der Controls angewendet werden.

1. Das Problem im Produktivbetrieb

Modul 1 hat GlobalDesk ein Dashboard gegeben, dessen Beschriftungen alle aus `Strings.resx` stammen, und das ist die richtige Heimat für Text, den der Code erzeugt. Beim Kundeneditor liegt der Fall anders: Er ist ein entworfener Bildschirm, nach Augenmaß im Visual Studio Designer angeordnet. Jede seiner Beschriftungen über einen Helper-Aufruf zu leiten hieße, die Designerdatei von Hand zu bearbeiten, und es löst das eigentliche Problem trotzdem nicht: dass ein deutsches *Kundennummer speichern* nicht in eine Schaltfläche passt, die für *Save* bemessen ist. Dieses Modul nutzt den Lokalisierungsworkflow, den Visual Studio ohnehin anbietet, sodass sich Beschriftungen und Geometrie eines entworfenen Bildschirms je Sprache unterscheiden können, ohne fest verdrahtete Zeichenfolgen und ohne eine zweite Layoutdatei.

2. Zentrale Konzepte

Konzept	Warum es wichtig ist
<code>Localizable</code>	Eine Entwurfszeit-Eigenschaft auf einem Container der obersten Ebene; <code>true</code> versetzt den Designer in den Lokalisierungsworkflow, sodass Werte je Kultur abgelegt werden können.
<code>Language</code> -Selektor des Designers	Erscheint, sobald <code>Localizable</code> auf <code>true</code> steht; die ausgewählte Kultur ist die Fassung, die Sie bearbeiten, deshalb wird jede Änderung nur für diese Kultur aufgezeichnet.
Die neutrale Sprache	Die Standardfassung, auf die jede Kultur zurückfällt, und die einzige, in der Sie Controls hinzufügen, entfernen oder umstrukturieren; Kulturanzeigen enthalten Übersreibungen, keinen zweiten Bildschirm.
Lokalisierbare Eigenschaften	Nicht nur <code>Text</code> : <code>Size</code> , <code>Location</code> , Farben und Bilder können je Kultur einen anderen Wert tragen, denn eine Übersetzung verändert neben den Wörtern auch die Geometrie.
Generierte Ressourcendateien	Der Designer schreibt lokalisierte Werte in Ressourcendateien statt in den C#-Designercode und hält sprachspezifische Werte aus dem Code heraus, den Sie täglich lesen und zusammenführen.
Beim Erstellen angewendet	Im Designer lokalisierte Werte werden beim Aufbau der Controls angewendet, deshalb schreibt eine spätere Änderung der Sitzungskultur einen bereits bestehenden Container nicht neu.
Übersetzungstolerantes Layout	Docking, Anchoring, <code>FlowLayoutPanel</code> , <code>TableLayoutPanel</code> , <code>FlexLayoutPanel</code> , <code>AutoSize</code> und Padding lassen Text von selbst wachsen, sodass Sie weniger Geometrie-Übersreibungen brauchen.

3. Der rote Faden der Lektion

Localizable einschalten

Die Lokalisierung im Designer wird auf dem Container aktiviert, nicht auf den einzelnen Controls. Öffnen Sie eine Wisej.NET- `Form` , - `Page` , ein `UserControl` oder einen `Desktop` im Entwurfsmodus, wählen Sie im Eigenschaftenfenster den Container selbst aus und setzen Sie `Localizable` auf `true` . Jedes Control, das der Container besitzt, wird damit vom Workflow erfasst; Sie setzen die Eigenschaft nie auf einem untergeordneten `Button` oder `Label` .

Tun Sie das, nachdem das neutrale Layout fertig ist, nicht vorher. Im Lab stellen Sie zuerst das englische GlobalDesk-UserControl `CustomerEditor` fertig: Felder, GroupBox, die Schaltflächen Save und Cancel, Docking. Den Schalter mittendrin umzulegen heißt, strukturelle Entscheidungen bei ausgewählter Kultur zu treffen, sodass eine Entscheidung, die zu jeder Sprache gehört, für eine einzelne aufgezeichnet wird.

Über den Language-Selektor bearbeiten

Mit eingeschaltetem `Localizable` bietet der Designer auf dem Container eine `Language`-Auswahl an. Belassen Sie sie auf dem Standardeintrag, um die neutrale Fassung zu bearbeiten. Wählen Sie eine Kultur wie Deutsch, und die Entwurfsfläche wird zu dieser lokalisierten Fassung des Bildschirms: Sie zeigt weiterhin Ihr Layout, aber jede Eigenschaft, die Sie nun ändern, wird als Überschreibung allein für diese Kultur aufgezeichnet.

Die Disziplin ist einfach: Ändern Sie nur, was sich unterscheiden muss. Im Lab wechseln Sie zu Deutsch, übersetzen die Beschriftungen, verbreitern die beiden längsten Schaltflächen und verschieben ein Label — und rühren sonst nichts an. Wechseln Sie dann vor jeder strukturellen Änderung zurück zur Standardsprache, denn ein Control bei ausgewähltem Deutsch hinzuzufügen ist der Weg, auf dem ein Team bei einem Bildschirm landet, der in einer Sprache ein Control hat und in einer anderen nicht.

Mehr als Text: Size, Location, Farben und Bilder

Wisej.NET nennt `Size`, `Location`, `Text`, Farben und Bilder als Eigenschaften, die sich je Sprache unterscheiden dürfen, und diese Liste ist keine Zierde. Deutsche Komposita werden lang, deshalb schneidet eine für *Save* bemessene Schaltfläche *Speichern* ab. Französische Dialoge brauchen oft mehr Höhe. Eine Illustration, die in einem Markt klar verständlich ist, kann in einem anderen nichts bedeuten, deshalb ist auch ein `Image` ein legitimer lokalisierter Wert.

Mnemonic-Zeichen gehören in dieselbe Kategorie: Die übersetzte Beschriftung entscheidet, welcher Zugriffstastenbuchstabe in dieser Sprache verfügbar und sinnvoll ist, nehmen Sie also nie an, dass der englische mitkommt. Und obwohl Farben lokalisierbar sind, ist eine Änderung je Kultur seltener als eine Änderung der Geometrie; behandeln Sie eine Farbe je Sprache also als Entscheidung, die Sie begründen können, und nicht als Reflex.

Was der Designer schreibt und wohin es geht

Wenn Sie eine Kulturansicht bearbeiten, hängt der Designer keinen Zweig an `InitializeComponent` an. Er legt die lokalisierten Eigenschaftswerte in generierten Ressourcendateien neben dem Container ab, einen Satz je bearbeiteter Kultur, und hält sprachspezifische Werte aus dem eigentlichen C#-Designercode heraus. Diese Trennung macht den Workflow überlebensfähig: Ihr Designercode bleibt lesbar, die Werte je Sprache sind Daten, und ein Reviewer kann eine Ressourcendatei lesen, ohne C# zu lesen.

Sie zieht außerdem eine Grenze für den Rest der Anwendung. Designer-Ressourcen gehören zu einem entworfenen Bildschirm, deshalb passen feste Beschriftungen, Schaltflächentexte, Tooltips, statische Bilder und Geometrie je Sprache dorthin. Alles, was im Code erzeugt wird, jede Meldung, die ein Service hervorbringt, jeder Validierungstext und alles, was mehr als ein Bildschirm wiederverwendet, gehört in eine Anwendungsressourcendatei wie `Strings.resx`, gelesen über `Wisej.Resources.ResourceManager` — genau das, was Modul 1 gebaut hat. GlobalDesk behält beides.

Designer-Ressourcen werden beim Erstellen der Controls angewendet

Das ist die Regel, die überrascht. Im Designer lokalisierte Werte werden beim Aufbau des Control-Baums angewendet. `Application.CurrentCulture` danach zu setzen ändert die Kultur der Sitzung, und Code, der ab diesem Moment Ressourcen liest, übernimmt die neue Sprache — aber es reist nicht zurück und

schreibt Eigenschaften auf bereits bestehenden Controls neu. Ein Benutzer, der die Sprache wechselt und den halben Editor weiterhin auf Englisch sieht, ist auf diese Regel gestoßen und nicht auf einen Fehler.

Sie haben zwei ehrliche Möglichkeiten: den lokalisierten Container neu aufbauen, damit seine Ressourcenwerte unter der neuen Kultur angewendet werden, oder ihn behalten und Code schreiben, der die ressourcengestützten Eigenschaften selbst erneut setzt. Für einen Bildschirm dieser Größe ist der Neuaufbau einfacher, und deshalb fügt das Lab eine Schaltfläche Recreate editor hinzu, die das Control verwirft und ein neues aufbaut.

```
private void btnRecreateEditor_Click(object sender, EventArgs e)
{
    // Designer resources are applied while the controls are created,
    // so the new culture only reaches this screen on a rebuild.
    this.hostPanel.Controls.Remove(this.customerEditor);
    this.customerEditor.Dispose();

    this.customerEditor = new CustomerEditor { Dock = DockStyle.Fill };
    this.hostPanel.Controls.Add(this.customerEditor);
}
```

Layouts planen, die längere Übersetzungen überstehen

Jede Geometrie-Überschreibung, die Sie aufzeichnen, ist ein Wert, den jemand in jeder künftigen Sprache pflegen muss. Die günstigste Überschreibung ist die, die Sie nie gebraucht haben; bauen Sie das neutrale Layout also aus Dingen, die sich anpassen: Docken Sie ein Panel an, statt es festzunageln, verankern Sie eine Schaltfläche am rechten Rand, statt ihre **Location** festzulegen, lassen Sie ein **FlowLayoutPanel**, **TableLayoutPanel** oder **FlexLayoutPanel** eine Reihe platzieren, schalten Sie **AutoSize** dort ein, wo eine Beschriftung ihre Breite selbst bestimmen soll, und lassen Sie Padding, damit ein längeres Wort irgendwohin kann.

Und dann belegen Sie es. Eine Designeransicht ist nicht die endgültige Antwort: Führen Sie GlobalDesk im Browser unter der Zielkultur aus. Achten Sie auf abgeschnittene Beschriftungen, überlappende Schaltflächen, Labels, die in ein Nachbarelement umbrechen, eine Tabulatorreihenfolge, die der sichtbaren nicht mehr folgt, auf Bilder und auf jeden Dialog, den der Bildschirm öffnen kann. Das Lab endet mit einer Notiz, die auflistet, welche Eigenschaften Sie für Deutsch überschrieben haben und welche Sie bewusst dem neutralen Layout überlassen haben — und diese zweite Liste ist die interessante: Sie ist der Beleg, dass Ihr Layout die Arbeit macht und nicht Ihre Überschreibungen.



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 Lokalisierung auf einer Wisej.NET-Form, -Page, einem UserControl oder einem Desktop aktivieren.
- ✓ den Language-Selektor des Designers verwenden, um lokalisierte Eigenschaften zu bearbeiten.
- ✓ erklären, welche Eigenschaften sich üblicherweise je Sprache ändern: Text, Size, Location, Color und Images.
- ✓ verstehen, was der Designer in .resx-Dateien schreibt.
- ✓ Layouts planen, die längere Übersetzungen überstehen.

Lab- und Prüfungsleitfaden

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



Praktisches Lab

Lokalisieren Sie den GlobalDesk-Kundeneditor über den Visual Studio Designer. Stellen Sie zuerst das neutrale englische Layout des UserControl's CustomerEditor fertig, setzen Sie dann Localizable auf true und fügen Sie über den Language-Selektor des Designers eine deutsche Variante hinzu, wobei Sie nur ändern, was sich unterscheiden muss: die Beschriftungen, die Breite der beiden längsten Schaltflächen und die Location eines Labels. Führen Sie die Anwendung unter de-DE aus und vergleichen Sie die tatsächliche Browserausgabe mit der Designeransicht, und fügen Sie dann eine Schaltfläche Recreate editor hinzu, die das Control verwirft und neu aufbaut, damit Sie selbst sehen, warum Designer-Ressourcen einem Kulturwechsel zur Laufzeit nicht folgen. Schließen Sie mit einer Lab-Notiz ab, die die überschriebenen Eigenschaften auflistet und diejenigen, die Sie bewusst dem neutralen Layout überlassen haben.

Ausgangspunkt

Sie öffnen GlobalDesk aus Modul 1, wo die Dashboard-Seite bereits jede Beschriftung über den `Texts`-Helper aus `Resources/Strings.resx` liest und ihre Vorschauwerte mit `Application.CurrentCulture` formatiert. In diesem Lab arbeiten Sie am UserControl `CustomerEditor`, das die Schaltfläche Customers des Dashboards öffnet: die Felder für Kundencode, Name, Ort und Notizen samt Labels, eine GroupBox und die Schaltflächen Save und Cancel.

Vorgeschlagenes Vorgehen

- ✓ Stellen Sie das neutrale englische Layout von `CustomerEditor` fertig, bevor Sie irgendetwas lokalisieren: Docken Sie das Notizfeld an, verankern Sie Save und Cancel unten rechts, legen Sie die Paare aus Label und Editor in ein Container-Layout wie ein `TableLayoutPanel` oder `FlexLayoutPanel` und setzen Sie `AutoSize` auf den Labels, die sich selbst anpassen sollen.
- ✓ Wählen Sie im Designer das UserControl selbst aus, setzen Sie `Localizable` im Eigenschaftenfenster auf `true` und prüfen Sie, dass die Auswahl `Language` nun erscheint; belassen Sie sie auf dem Standardeintrag und erstellen Sie einmal neu, damit die neutrale Fassung die Grundlage ist.
- ✓ Stellen Sie die Auswahl `Language` auf Deutsch um und übersetzen Sie ausschließlich die Beschriftungen — die Labels, den Kopftext der GroupBox, Save und Cancel. Verbreitern Sie dann über `Size` die beiden Schaltflächen, deren deutscher Text nicht mehr passt, und verschieben Sie ein Label über seine `Location`, damit die Spalte weiterhin ausgerichtet ist. Wechseln Sie zurück zur Standardsprache, bevor Sie irgendetwas Strukturelles anfassen.
- ✓ Führen Sie GlobalDesk unter `de-DE` aus und prüfen Sie den Browser, nicht den Designer: Achten Sie auf abgeschnittene Beschriftungen, überlappende Controls, ein Label, das in sein Nachbar-element umbricht, die Tabulatorreihenfolge und die Dialoge, die der Editor öffnen kann. Halten Sie jeden Unterschied fest, den Sie zwischen Designeransicht und echter Seite finden.

- ✓ Fügen Sie auf der Hostseite eine Schaltfläche **Recreate editor** hinzu, die den aktuellen **CustomerEditor** aus seinem Container entfernt, ihn verwirft und einen frisch erzeugten einfügt. Wechseln Sie die Kultur, drücken Sie die Schaltfläche und sehen Sie zu, wie die Beschriftungen endlich folgen; schreiben Sie dann die Notiz mit jeder Eigenschaft, die Sie für Deutsch überschrieben haben, und jeder, die Sie dem neutralen Layout überlassen haben.

Geforderte Ergebnisse

- ✓ UserControl CustomerEditor mit Localizable auf true über einem vollständigen neutralen englischen Layout
- ✓ Deutsche Designer-Sprachvariante, die die Beschriftungen sowie mindestens eine Size und eine Location überschreibt
- ✓ Anchoring, Docking oder AutoSize so eingesetzt, dass die deutschen Beschriftungen nicht abgeschnitten werden können
- ✓ Schaltfläche Recreate editor, die den Container nach einem Kulturwechsel verwirft und neu aufbaut
- ✓ Lab-Notiz mit den überschriebenen Eigenschaften und denen, die dem neutralen Layout überlassen wurden

Wiederholungsfragen

- ✓ Ein Teammitglied fügt **CustomerEditor** ein neues Telefonfeld hinzu, während in der **Language** -Liste Deutsch ausgewählt ist. Was sieht der englische Benutzer wahrscheinlich, und welche Regel hätte das verhindert?
- ✓ Die Speicherbestätigung, die der Editor nach einer erfolgreichen Aktualisierung zeigt, stammt weiterhin aus **Strings.resx** und nicht aus den Designer-Ressourcen. Nennen Sie zwei Gründe, warum das der richtige Ort dafür ist, und benennen Sie eine Beschriftung auf diesem Bildschirm, die tatsächlich in den Designer gehört.
- ✓ Ihre Schaltfläche Recreate editor funktioniert, aber ein Kollege argumentiert, der Bildschirm solle die lokalisierten Eigenschaften einfach erneut anwenden, statt neu aufzubauen. Was müsste dieser Code wissen, was der Neuaufbau nicht wissen muss, und wann wäre sein Ansatz der bessere Kompromiss?



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.