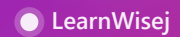


Das Bildmodell von Wisej.NET: Image, ImageSource und ImageList

Lektionsleitfaden · Lab- und Prüfungsleitfaden



Lektionsleitfaden

Die zwei Bild-Pipelines: eine Eigenschaft Image, die ein System.Drawing.Image enthält, das der Server als PNG an den Browser sendet, gegenüber einem String ImageSource oder IconSource, den der Client als URL, als Theme-Namen, als resource.wx-Pfad oder als Element eines Icon-Packs auflöst – dazu der gemeinsame Vertrag IImage und die ImageList, über die viele Steuerelemente einen gemeinsamen, nach Schlüsseln adressierten Satz von Bildern nutzen. Was das für Entwicklerinnen und Entwickler bedeutet, die die Grundlagen von C# beherrschen und schon etwas mit den Steuerelementen und dem Designer von Wisej.NET gearbeitet haben – und wie Sie dieses Modul lesen.



Fokus: Produktivbetrieb

Die zwei Bild-Pipelines: eine Eigenschaft Image, die ein System.Drawing.Image enthält, das der Server als PNG an den Browser sendet, gegenüber einem String ImageSource oder IconSource, den der Client als URL, als Theme-Namen, als resource.wx-Pfad oder als Element eines Icon-Packs auflöst – dazu der gemeinsame Vertrag IImage und die ImageList, über die viele Steuerelemente einen gemeinsamen, nach Schlüsseln adressierten Satz von Bildern nutzen.

1. Das Problem im Produktivbetrieb

Wisej.NET bietet bewusst mehr als einen Weg, eine Grafik auf ein Steuerelement zu bringen, und jeder davon funktioniert. Die Schwierigkeiten beginnen, wenn ein Team versehentlich alle Wege gleichzeitig verwendet: eine Toolbar, bei der ein Button eine Datei von der Serverfestplatte liest, der nächste auf einen Theme-Namen zeigt und ein dritter sein Icon still verloren hat, weil zwei Eigenschaften auf demselben Steuerelement gesetzt wurden. Sechs Monate später kann niemand mehr erklären, woher ein bestimmtes Bild eigentlich kommt oder warum es nach dem Deployment verschwunden ist.

2. Zentrale Konzepte

Konzept	Warum es wichtig ist
Eigenschaft Image	Ein <code>System.Drawing.Image</code> -Objekt, das im Serverspeicher liegt; Wisej.NET sendet es als PNG-Daten an den Browser, das Bild reist also als Bytes, die Ihr Code bereits dekodiert hat.
String ImageSource	Ein String, der eine Ressource benennt, die der Client auflösen soll: eine relative URL, eine absolute URL, ein benanntes Theme-Bild, ein <code>resource.wx</code> -Pfad oder ein Element in einer Icon-Pack-Assembly.
IconSource	Dieselbe String-Idee auf Steuerelementen, die neben ihrem Hauptbild einen eigenen Icon-Platz besitzen – prüfen Sie also die API des verwendeten Steuerelements, statt anzunehmen, jedes Steuerelement benenne die Eigenschaft gleich.
Vertrag IImage	Die gemeinsame Bildfamilie, die bildfähige Steuerelemente anbieten – <code>IImage</code> , <code>ImageAlign</code> , <code>ImageIndex</code> , <code>ImageKey</code> , <code>ImageList</code> und <code>ImageSource</code> –, sodass eine Hilfsmethode einen Button, ein Label oder beliebige andere Steuerelemente einheitlich ausstatten kann.
ImageList	Eine wiederverwendbare Sammlung von Bildobjekten, auf die mehrere Steuerelemente zeigen können, adressiert über Position oder Name, und in der Lage, für die nutzenden Steuerelemente eine gemeinsame Bildgröße festzulegen.
ImageKey gegenüber ImageIndex	Ein Schlüssel wie <code>"delete"</code> trägt Bedeutung und übersteht ein Umsortieren; der Index <code>3</code> wird stillschweigend zum falschen Bild, sobald jemand ein Bild darüber einfügt.
Sich gegenseitig ausschließende Zuweisung	All diese Eigenschaften bedienen einen einzigen visuellen Platz, eine Zuweisung kann die anderen also zurücksetzen – deshalb ist ein falsches Icon meist eine zweite Zuweisung und keine defekte Datei.

3. Der rote Faden der Lektion

Objekt oder String: die erste Frage, die Sie beantworten

Bevor Sie etwas tippen, entscheiden Sie, ob der Server ein *Bildobjekt* besitzt oder ob der Browser eine *Bildkennung* auflösen soll. Das sind die beiden Pipelines, und fast jede spätere Verwirrung in diesem Kurs entsteht daraus, sie zu vermischen. Wenn Sie `IImage` zuweisen, übergeben Sie Wisej.NET ein `System.Drawing.Image` : echte dekodierte Pixeldaten im Speicher Ihrer Session, die das Framework anschließend als PNG an den Browser sendet. Das ist der richtige Weg, wenn Ihr C#-Code tatsächlich ein Bildobjekt erzeugt oder empfangen hat – ein gerendertes Diagramm, ein verkleinertes Vorschaubild, ein Foto, das als Stream von einem Dienst kam.

Wenn Sie `ImageSource` oder `IconSource` zuweisen, übergeben Sie Wisej.NET stattdessen einen String. Auf dem Server wird nichts dekodiert. Der String benennt etwas, das der Client selbst holen und zwischenspeichern kann: `"Images/open.png"` relativ zur Anwendung, eine absolute Adresse, einen Theme-Bildnamen wie `"icon-save"` oder einen `resource.wx` -Pfad in eine Assembly. Weil die Ressource ihre Identität behält, bleibt ein SVG bis in den Browser hinein ein SVG – deshalb bleiben Oberflächen-Icons

in jeder Größe scharf und lassen sich später umfärben. In IconDesk sehen Sie beides: `btnOpen` erhält ein vom Server geladenes Bildobjekt, `btnSave` einen Theme-Namen als String, und das Diagnose-Label muss sie auseinanderhalten.

Eine Eigenschaftsfamilie namens `IImage`

Wisej.NET gibt nicht jedem Steuerelement ein eigenes Vokabular für Bilder. Bildfähige Steuerelemente teilen sich einen Vertrag namens `IImage`, der `Image`, `ImageAlign`, `ImageIndex`, `ImageKey`, `ImageList` und `ImageSource` zu einer Familie zusammenfasst. Diese Familie zu erkennen ist für Einsteiger wertvoller, als auf sie zu casten: Wenn Sie einem neuen Steuerelement begegnen, wissen Sie bereits, nach welchen Eigenschaften Sie suchen und welche vier davon um denselben Platz konkurrieren.

Der Cast bleibt für Infrastrukturcode nützlich. Eine Methode, die ein `Control` entgegennimmt, `is IImage` prüft und dann `ImageSource` setzt, kann das Speichern-Icon des Projekts auf einen Button, ein Label oder einen Menüeintrag anwenden, ohne eine switch-Anweisung. Das Diagnose-Label von IconDesk im Lab macht das Umgekehrte: Es liest jeden Button über `IImage`, prüft, welche der vier Eigenschaften belegt ist, und gibt den Mechanismus hinter dem Icon aus, statt ihn aus dem Bild zu erraten.

Den Mechanismus zum vorliegenden Asset wählen

Vier Fälle decken fast alles ab. Eine neben der Anwendung ausgelieferte Datei ist eine Bildquelle mit relativer URL. Ein anderswo gehostetes Bild ist eine Bildquelle mit absoluter URL. Ein Icon, das das aktive Theme bereits definiert, ist eine Bildquelle mit dem Namen des Themes dafür – der Client nimmt es dann aus dem Theme-Cache, den er ohnehin geladen hat. Ein Bild, das Ihr Code erzeugt, verarbeitet oder als .NET-Objekt empfangen hat, ist ein `Image`.

Der fünfte Fall ist der gemeinsame, und dafür gibt es die `ImageList`. Eine `ImageList` hält eine Sammlung von Bildobjekten; Steuerelemente referenzieren einen Eintrag über den numerischen `ImageIndex` oder über den String `ImageKey`. Zwei Eigenschaften dieser Indirektion sind in einer echten Anwendung wichtig: Viele Steuerelemente können eine Sammlung gemeinsam nutzen, und das unter einem Schlüssel gespeicherte Bild lässt sich ersetzen, ohne ein einziges Steuerelement anzufassen, das diesen Schlüssel verwendet. Die Liste kann außerdem eine Bildgröße für die nutzenden Steuerelemente festlegen, was praktisch ist, wenn sonst das Theme über die Icon-Größe entscheiden würde.

```
var icons = new ImageList();
icons.Images.Add("print", Image.FromFile("Images/print.png"));
icons.Images.Add("delete", Image.FromFile("Images/delete.png"));

btnPrint.ImageList = icons;
btnPrint.ImageKey = "print";
btnDelete.ImageList = icons;
btnDelete.ImageKey = "delete";

// btnSave holt sein Icon stattdessen aus einem String – ohne ImageList.
btnSave.ImageSource = "icon-save";
```

Beachten Sie, was die ersten Zeilen *nicht* tun. `Image.FromFile` liest eine Datei auf dem Server, nie auf dem Rechner der Benutzerin oder des Benutzers – ein absoluter Pfad, der auf Ihrem Laptop funktioniert hat, ist

also ein Deployment-Fehler mit Ansage. Ein Bildquellen-String wird dagegen so adressiert, wie der Browser alles andere adressiert.

Warum das gesetzte Icon nicht immer das sichtbare Icon ist

Ein Steuerelement hat einen visuellen Platz für sein Bild, und `Image`, `ImageSource`, `ImageIndex` und `ImageKey` konkurrieren alle darum, ihn zu füllen. Wisej.NET löst das, indem es sie gegenseitig ausschließt: Eine zu setzen kann die anderen zurücksetzen, sodass die Konfiguration nie mehrdeutig ist. Wenn `btnDelete` zufrieden den Listeneintrag unter `ImageKey = "delete"` zeigt und eine spätere Zeile `ImageSource` setzt, wird der String zur aktiven Darstellung und der Schlüssel spielt keine Rolle mehr.

Diese eine Regel erklärt die meisten Fehlermeldungen der Art „das falsche Icon wird angezeigt“. Der Designer hat eine Eigenschaft gesetzt, der Konstruktor des Formulars eine zweite und ein Theme oder eine Hilfsmethode eine dritte. Wenn das Bild falsch ist, öffnen Sie nicht als Erstes die Datei – untersuchen Sie jede bildbezogene Eigenschaft des Steuerelements und finden Sie die zuletzt ausgeführte Zuweisung.

- Eine Bildeigenschaft pro Steuerelement, bewusst gewählt.
- Schlüssel statt Indizes, wo die Identität Bedeutung hat.
- Eine Textalternative für jedes Icon, das für sich genommen Bedeutung trägt.



Zuerst den Mechanismus benennen

Bevor Sie ein Bild zuweisen, sagen Sie, welchen Mechanismus Sie verwenden: ein `System.Drawing.Image`-Objekt, das der Server als PNG rendert, einen Bildquellen-String, den der Browser auflöst, ein benanntes Theme-Bild, eine über `resource.wx` eingebettete Ressource oder einen Icon-Font, der als HTML gezeichnet wird. Weisen Sie pro Steuerelement genau eine Bildeigenschaft zu, damit die anderen sie nicht still zurücksetzen, halten Sie den Quell-String aussagekräftig statt als eingefügten magischen Pfad, bevorzugen Sie SVG für Oberflächen-Icons, damit sie in jeder Größe und Skalierung scharf bleiben, und geben Sie jedem bedeutungstragenden Icon eine Textalternative.



Lernziele

Am Ende dieses Moduls können Sie:

- ✓ Den Unterschied zwischen einem `System.Drawing.Image`-Wert und einem Bild-String wie `ImageSource` oder `IconSource` erklären.
- ✓ Das gemeinsame Modell `Image` nutzen, um die Eigenschaften `Image`, `ImageSource`, `ImageList`, `ImageIndex` und `ImageKey` wiederzuerkennen.
- ✓ Den richtigen Bildmechanismus für eine lokale Bitmap, eine URL, ein Theme-Bild oder ein wiederverwendbares Icon wählen.
- ✓ Verstehen, warum das Setzen einer von mehreren sich gegenseitig ausschließenden Bildeigenschaften die anderen zurücksetzen kann.

Lab- und Prüfungsleitfaden

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



Praktisches Lab

Erstellen Sie das Projekt IconDesk und geben Sie ihm ein Befehls-Panel mit den Buttons Open, Save, Print und Delete, von denen jeder sein Icon über einen anderen Mechanismus erhält. Laden Sie btnOpen aus einem serverseitigen System.Drawing.Image, weisen Sie btnSave.ImageSource ein benanntes Theme-Bild zu und lassen Sie btnPrint und btnDelete über ImageKey auf eine gemeinsame ImageList zeigen. Ergänzen Sie ein Diagnose-Label, das jeden Button über IImage untersucht und meldet, welcher Mechanismus tatsächlich im Spiel ist; tauschen Sie dann das unter einem Schlüssel der ImageList gespeicherte Bild aus und prüfen Sie, dass jedes Steuerelement, das diesen Schlüssel verwendet, der Änderung folgt.

Ausgangspunkt

Dies ist das erste Modul, Sie beginnen also bei null: Erstellen Sie eine neue Wisej.NET-4-Anwendung namens **IconDesk** und fügen Sie eine Page für das Befehls-Panel hinzu. Legen Sie einen kleinen Bildordner neben die Anwendung – eine Grafik für Open, Save, Print und Delete genügt – und prüfen Sie, dass das leere Projekt baut und sich im Browser öffnet, bevor Sie ein einziges Icon zuweisen.

Vorgeschlagenes Vorgehen

- ✓ Platzieren Sie zuerst die vier Buttons nur mit Text, damit sich das Panel lesen lässt, bevor es ein Bild gibt.
- ✓ Geben Sie **btnOpen** ein **System.Drawing.Image**, das der Server lädt, und **btnSave** einen Theme-Bildnamen auf **ImageSource**; sprechen Sie laut aus, welche Pipeline jeweils verwendet wird.
- ✓ Erstellen Sie eine **ImageList**, fügen Sie die Bilder für Print und Delete unter sprechenden Schlüssel hinzu und lassen Sie dann beide Buttons über **ImageKey** statt **ImageIndex** auf die Liste zeigen.
- ✓ Schreiben Sie das Diagnose-Label: Gehen Sie die Steuerelemente des Panels durch, prüfen Sie jedes auf **IImage** und melden Sie, welche der Angaben **Image**, **ImageSource**, **ImageList** plus **ImageKey** tatsächlich belegt ist.
- ✓ Ersetzen Sie das unter dem Delete-Schlüssel gespeicherte Bild, laden Sie die Seite neu und setzen Sie dann absichtlich eine zweite Bildeigenschaft auf einem Button, um festhalten zu können, was aus der ersten Zuweisung wurde.

Geforderte Ergebnisse

- ✓ IconDesk-Befehls-Panel mit den Buttons Open, Save, Print und Delete
- ✓ btnOpen mit einem auf dem Server geladenen System.Drawing.Image und btnSave mit einem benannten Theme-Bild auf ImageSource
- ✓ btnPrint und btnDelete, die sich über ImageKey statt ImageIndex eine ImageList teilen

- ✓ Diagnose-Label, das jeden Button über `Image` liest und den Mechanismus hinter seinem Icon benennt
- ✓ Lab-Notiz, die festhält, was beim Austausch eines `ImageList`-Schlüssels geschah und was, als zwei Bildeigenschaften auf demselben Button gesetzt wurden

Wiederholungsfragen

- ✓ Ihr Diagnose-Label meldet, `btnDelete` verwende eine Bildquelle, obwohl Sie nur `ImageKey` gesetzt haben. Was muss gelaufen sein, und wo würden Sie danach suchen?
- ✓ Print und Delete haben sich beide geändert, als Sie einen Eintrag in der `ImageList` ersetzt haben. Wäre das auch so, wenn die Buttons `ImageIndex` verwendeten, und was passierte, wenn ein fünftes Bild oben in die Liste eingefügt würde?
- ✓ Der Open-Button liest seine Datei auf dem Server. Welcher Teil dieser Anordnung bricht, wenn die Anwendung auf einem anderen Rechner ausgeliefert wird, und welcher Teil bräche nicht, wenn dasselbe Bild ein Bildquellen-String wäre?



Zuerst den Mechanismus benennen

Bevor Sie ein Bild zuweisen, sagen Sie, welchen Mechanismus Sie verwenden: ein `System.Drawing.Image`-Objekt, das der Server als PNG rendert, einen Bildquellen-String, den der Browser auflöst, ein benanntes Theme-Bild, eine über `resource.wx` eingebettete Ressource oder einen Icon-Font, der als HTML gezeichnet wird. Weisen Sie pro Steuerelement genau eine Bildeigenschaft zu, damit die anderen sie nicht still zurücksetzen, halten Sie den Quell-String aussagekräftig statt als eingefügten magischen Pfad, bevorzugen Sie SVG für Oberflächen-Icons, damit sie in jeder Größe und Skalierung scharf bleiben, und geben Sie jedem bedeutungstragenden Icon eine Textalternative.