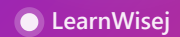


Bitmaps, SVGs und PictureBox: Formate, Größenanpassung und Laden

Lektionsleitfaden · Lab- und Prüfungsleitfaden



Lektionsleitfaden

Zwei Formatlisten, weil es zwei Pipelines gibt: Bitmap, Metafile, Icon, JPEG, GIF und PNG über System.Drawing und PictureBox.Image, dagegen jpg, jpeg, png, gif, svg und bmp, die resource.wx und Theme-Bilder als String akzeptieren – dazu die Size Modes der PictureBox und die Einstiegspunkte Load (synchron) und LoadAsync (asynchron). 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

Zwei Formatlisten, weil es zwei Pipelines gibt: Bitmap, Metafile, Icon, JPEG, GIF und PNG über System.Drawing und PictureBox.Image, dagegen jpg, jpeg, png, gif, svg und bmp, die resource.wx und Theme-Bilder als String akzeptieren – dazu die Size Modes der PictureBox und die Einstiegspunkte Load (synchron) und LoadAsync (asynchron).

1. Das Problem im Produktivbetrieb

„Welche Bildformate unterstützt Wisej.NET?“ ist die Frage, die jedes Team in der ersten Woche stellt, und die ehrliche Antwort hängt davon ab, welchen API-Weg das Bild nimmt. Beantworten Sie sie mit einer vagen Liste, bekommen Sie ein verzerrtes Logo auf einem High-DPI-Laptop, eine Seite, die einfriert, während ein Produktfoto von einem langsamen Host geladen wird, und eine Fehlermeldung, weil sich eine `.ico`-Datei nicht über einen Ressourcen-String auflösen lässt. Dieses Modul ersetzt die vage Liste durch zwei präzise, jede fest verbunden mit der Eigenschaft des Steuerelements, die sie verwendet.

2. Zentrale Konzepte

Konzept	Warum es wichtig ist
System.Drawing-Pipeline	Die Eigenschaft <code>Image</code> hält ein <code>System.Drawing.Image</code> im Serverspeicher; Wisej.NET sendet es als PNG-Daten an den Browser. Ihre dokumentierten Quellen sind Bitmap-, Metafile-, Icon-, JPEG-, GIF- und PNG-Dateien.
Bildquellen-Pipeline	Ein String, den der Client selbst auflöst: eine relative oder absolute URL, ein benanntes Theme-Bild oder ein <code>resource.wx</code> -Pfad. Die dokumentierten Bild-Erweiterungen für <code>resource.wx</code> sind .jpg, .jpeg, .png, .gif, .svg und .bmp.
PictureBox.Image	Der Einstiegspunkt für Pixel, die Ihr C#-Code bereits besitzt oder gerade erzeugt hat, etwa ein mit <code>Graphics</code> gezeichnetes Diagramm.
PictureBox.ImageSource	Der Einstiegspunkt für ein Bild, das der Browser selbst holen kann. Statt Bildbytes reist ein kurzer String – deshalb gehört SVG hierher.
PictureBox.SizeMode	Die sechs Werte <code>Normal</code> , <code>AutoSize</code> , <code>CenterImage</code> , <code>StretchImage</code> , <code>Zoom</code> und <code>Cover</code> , die entscheiden, ob ein Bild beschnitten, zentriert, verzerrt, eingepasst oder zugeschnitten wird.
Load und LoadAsync	<code>Load(url)</code> aktualisiert das Bild synchron aus einer URL; <code>LoadAsync(url)</code> holt dieselbe Datei asynchron, sodass ein langsamer Host den Bildschirm nicht blockiert.
Vektor gegenüber Raster	Ein SVG ist Markup, das in der vom Layout gewünschten Größe gezeichnet wird, und geeignete einfarbige Grafiken lassen sich umfärben; eine Rasterdatei ist ein festes Pixelraster, das immer nur neu berechnet wird.

3. Der rote Faden der Lektion

Zwei Formatlisten, weil es zwei Pipelines gibt

Das erste Ziel dieses Moduls ist, aufzuhören, eine einzige Liste zu zitieren. Für die `PictureBox` beschreibt die Steuerelement-Dokumentation von Wisej.NET Grafiken, die aus Bitmap-, Metafile-, Icon-, JPEG-, GIF- und PNG-Dateien geladen werden. Das ist die vertraute Welt von `System.Drawing.Image`: Sie übergeben dem Steuerelement ein Objekt, und der Server macht daraus PNG-Daten auf der Leitung. Für String-Ressourcen sind die dokumentierten Bild-Erweiterungen von `resource.wx` .jpg, .jpeg, .png, .gif, .svg und .bmp, und Theme-Bilddefinitionen unterstützen png, gif, svg, jpg und bmp.

Interessant sind die Einträge, die nur in einer Liste auftauchen. Icon und Metafile gehören zur System.Drawing-Seite und stehen nicht auf der Liste der erlaubten Bilder für `resource.wx`. Bei SVG ist es umgekehrt: Es ist ein ausgezeichneter Wert für eine Bildquelle und ein vollwertiges Theme-Bild, aber es ist kein `System.Drawing.Image`-Format in demselben Sinn wie eine Bitmap oder ein PNG. Zu sagen „Wisej.NET unterstützt SVG“, ohne die Pipeline zu nennen, ist genau das, was später im Sprint zu einer fehlgeschlagenen `Image`-Zuweisung führt.

Im Lab halten Sie das ausdrücklich fest: Die Format-Notiz neben Ihrer Seite ImageLab listet jedes Asset auf und benennt die Pipeline, die es transportiert hat.

Die PictureBox hat zwei Einstiegspunkte, und Sie wählen einen pro Steuerelement

Beim zweiten Ziel geht es darum, die richtige Eigenschaft zu wählen. `PictureBox.Image` nimmt ein `System.Drawing.Image` entgegen. `PictureBox.ImageSource` nimmt einen String entgegen. Die Regel, die ein Projekt lesbar hält, ist einfach: Wenn Ihr C#-Code die Pixel erzeugt oder verändert, verwenden Sie `Image`; wenn das Bild eine Datei ist, die der Browser genauso gut selbst holen könnte, verwenden Sie `ImageSource` und halten den String aussagekräftig.

```
pictureBox1.ImageSource = "Images/product.png"; // ein String, den der Client auflöst
pictureBox1.SizeMode = PictureBoxSizeMode.Zoom;

pictureBox2.ImageSource = "resource.wx/IconDesk/logo.svg";
pictureBox2.SizeMode = PictureBoxSizeMode.Cover;

// eine URL, die die Benutzerin gewählt hat: holen, ohne die Seite anzuhalten
pictureBox3.LoadAsync(remoteJpegUrl);
```

Die Einstiegspunkte für URLs stehen neben diesen Eigenschaften. `Load(url)` aktualisiert das Bild synchron aus einer URL; `LoadAsync(url)` startet stattdessen ein asynchrones Laden. Ihre Seite ImageLab verwendet `LoadAsync` für das entfernte JPEG genau deshalb, damit ein langsamer Fremdhost zu „das Bild kommt einen Moment später“ führt und nicht zu „die Seite ist eingefroren“.

Sechs Size Modes und was jeder mit dem Bild macht

Wisej.NET dokumentiert sechs Werte für `PictureBoxSizeMode`, und das dritte Ziel ist, bewusst zwischen ihnen zu wählen, statt hinzunehmen, was der Designer hinterlassen hat. `Normal` zeichnet das Bild in Originalgröße ab der linken oberen Ecke und schneidet ab, was nicht passt. `AutoSize` dreht das Verhältnis um: Die PictureBox wächst oder schrumpft auf das Bild. `CenterImage` behält die Originalgröße, zentriert das Bild aber, sodass der Beschnitt gleichmäßig auf allen Seiten anfällt. `StretchImage` füllt die Box, ob das Seitenverhältnis das übersteht oder nicht. `Zoom` skaliert das Bild unter Wahrung des Seitenverhältnisses, bis das gesamte Bild hineinpasst. `Cover` wahrt das Seitenverhältnis ebenfalls, füllt aber das Rechteck und schneidet die überstehenden Ränder ab.

In der Praxis ist `Zoom` die sichere Wahl für Logos und Fotos, die vollständig sichtbar bleiben müssen, `Cover` die richtige für Vorschaukacheln und Banner, bei denen das Füllen des Rechtecks wichtiger ist als jeder Rand, und `StretchImage` sollte eine bewusst getroffene Entscheidung sein, denn es ist der Modus, der eine Wortmarke still verzerrt. Das Lab zeigt ein Rasterbild viermal, in `Normal`, `Zoom`, `Cover` und `StretchImage`, damit Beschnitt und Verzerrung nebeneinander sichtbar sind statt nur beschrieben.

SVG für Oberflächen-Icons, Raster für Fotos

Das vierte Ziel ist die Gestaltungsregel, die aus den zwei Pipelines folgt. SVG ist für Oberflächen-Icons meist die erste Wahl: Es ist Vektor-Markup, bleibt also bei 16, 24, 32 und 48 Pixeln und bei jedem Skalierungsfaktor scharf, einfache einfarbige Grafiken sind klein, und Wisej.NET kann ein geeignetes SVG über den Bildquellen-String umfärben. Bei Fotos und Screenshots ist es umgekehrt. PNG ist die Allzweckwahl im Rasterbereich, wenn Transparenz oder exakte Pixel zählen, JPEG ist für Fotos mit vertretbarer verlustbehafteter Kompression in der Regel besser, GIF bleibt für einfache oder animierte

Grafiken nützlich, und BMP wird zwar unterstützt, ist aber meist deutlich größer als ein komprimiertes Webformat.

Bei High-DPI hört der Unterschied auf, theoretisch zu sein. Ein Vektor-Icon braucht keine zweite Kopie für 150 % oder 200 % Skalierung; eine auf 32 Pixel vergrößerte 16-Pixel-Bitmap sieht schlicht vergrößert aus. Wenn Sie Raster-Icons ausliefern müssen, gehen Sie von genügend Auflösung für die größte gezeichnete Größe aus und testen Sie im Browser bei erhöhter Anzeigeskalierung.

- Befehls-, Navigations- und Status-Glyphen: SVG über `ImageSource` .
- Produktfotos, Screenshots und Scans: PNG oder JPEG, meist mit `Zoom` oder `Cover` .
- Pixel, die Ihr eigener Code zeichnet: ein `System.Drawing.Image` auf `Image` .

Die Größe ist eine Gestaltungsentscheidung, kein Zufall

Eine Bildressource hat eigene Maße, aber Ihre Oberfläche hat zusätzlich eine Ziel-Icon-Größe, und beides ist nicht dieselbe Zahl. Wisej-Themes steuern die Icon-Metriken üblicherweise, und eine `ImageList` kann eine explizite gemeinsame Größe festlegen und die von den Steuerelementen gezeichneten Bilder anpassen. Für ein Label hält die Dokumentation fest, dass die Originalgröße des Bildes verwendet wird, sofern keine `ImageList` beteiligt ist.

Das letzte Ergebnis schließt den Kreis zu Modul 1: Ein Report-Handler geht jede `PictureBox` der Seite `ImageLab` durch und gibt an, ob ihr Bild über `Image` oder über `ImageSource` gekommen ist. Dieser Handler ist der schnellste Weg, ein Steuerelement zu finden, auf dem jemand beide Eigenschaften gesetzt hat und nur eine davon gewinnt.



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:

- ✓ Die Bildformate benennen, die Wisej.NET über `PictureBox`, Themes und `resource.wx` akzeptiert.
- ✓ `PictureBox.Image` und `PictureBox.ImageSource` korrekt verwenden.
- ✓ Size Modes der `PictureBox` wählen, darunter Normal, `AutoSize`, `CenterImage`, `StretchImage`, `Zoom` und `Cover`.
- ✓ Erklären, warum SVG für skalierbare UI-Icons meist vorzuziehen ist und warum Fotos in der Regel Rasterbilder sind.

Lab- und Prüfungsleitfaden

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



Praktisches Lab

Bauen Sie in IconDesk eine Seite ImageLab, die die zwei Pipelines nebeneinander sichtbar macht. Zeigen Sie ein Rasterbild viermal mit SizeMode auf Normal, Zoom, Cover und StretchImage, damit die Unterschiede bei Zuschchnitt und Verzerrung offensichtlich werden, ergänzen Sie eine PictureBox, die ein entferntes JPEG mit LoadAsync holt und die Seite währenddessen bedienbar hält, und platzieren Sie ein über ImageSource zugewiesenes SVG in 16, 24, 32 und 48 Pixeln, um zu belegen, dass es scharf bleibt. Schließen Sie mit einem Handler ab, der für jede PictureBox der Seite meldet, ob ihr Bild aus Image oder aus ImageSource stammt.

Ausgangspunkt

Öffnen Sie die Projektmappe IconDesk aus Modul 1, in der das Befehls-Panel bereits vier Button-Icons über vier verschiedene Mechanismen trägt. Fügen Sie dem Projekt eine neue Page namens **ImageLab** hinzu und legen Sie ein Rasterbild, ein fotoartiges JPEG auf einem entfernten Host und ein einfarbiges SVG-Icon bereit, bevor Sie Steuerelemente platzieren.

Vorgeschlagenes Vorgehen

- ✓ Platzieren Sie auf ImageLab vier gleich große PictureBox-Steuerelemente und geben Sie ihnen dasselbe Rasterbild, setzen Sie dann **SizeMode** auf Normal, Zoom, Cover und StretchImage und beschriften Sie jedes mit dem Modus, den es zeigt.
- ✓ Fügen Sie eine fünfte PictureBox hinzu und rufen Sie aus dem Ladepfad der Seite **LoadAsync** mit der URL des entfernten JPEG auf; lassen Sie einen sichtbaren Platzhalter oder eine Bildunterschrift stehen, damit erkennbar ist, dass die Seite bedienbar blieb, während das Bild unterwegs war.
- ✓ Platzieren Sie vier weitere PictureBox-Steuerelemente mit 16, 24, 32 und 48 Pixeln, weisen Sie jedem über **ImageSource** dasselbe SVG zu und vergleichen Sie sie mit dem auf dieselben vier Größen skalierten Rasterbild.
- ✓ Schreiben Sie einen Handler **btnReport_Click**, der die Steuerelemente der Seite durchläuft und für jede PictureBox eine Zeile ausgibt, die das Steuerelement benennt und angibt, ob sein Bild aus **Image** oder aus **ImageSource** kam.
- ✓ Schreiben Sie die Format-Notiz: Listen Sie jedes Asset der Seite auf, geben Sie an, welche über System.Drawing laufen und welche ein Ressourcen-String bleiben, und halten Sie fest, welchen Size Mode Sie im Review für ein Logo, eine Vorschaukachel und ein Foto verteidigen würden.

Geforderte Ergebnisse

- ✓ Seite ImageLab, die ein Rasterbild in den Size Modes Normal, Zoom, Cover und StretchImage zeigt
- ✓ PictureBox, die ein entferntes JPEG mit LoadAsync holt, ohne die Seite zu blockieren

- ✓ Ein über ImageSource zugewiesenes SVG, gerendert in 16, 24, 32 und 48 Pixeln
- ✓ Report-Handler, der pro PictureBox benennt, ob Image oder ImageSource das Bild geliefert hat
- ✓ Format-Notiz, die auflistet, welche Lab-Assets über System.Drawing laufen und welche Ressourcen-Strings bleiben

Wiederholungsfragen

- ✓ Ihr Rasterbild ist breiter als hoch, und die Box ist quadratisch. Welcher der beiden Modi Zoom und Cover verliert einen Teil des Bildes, welcher lässt freien Raum, und welchen würden Sie für ein Kundenlogo verteidigen?
- ✓ Der Host des entfernten JPEG ist zehn Sekunden lang nicht erreichbar. Was erleben die Benutzer Ihrer Seite mit `LoadAsync`, und was hätten sie mit dem synchronen `Load` erlebt?
- ✓ Ihr Report-Handler meldet, das Bild eines Steuerelements stamme aus `ImageSource`, aber die Designer-Datei weist auf demselben Steuerelement auch `Image` zu. Welches wird tatsächlich gezeichnet, und wie würden Sie das belegen, statt zu raten?



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.