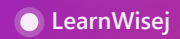


Icon-Fonts, HTML-Inhalte und Icon-Strategie für den Produktivbetrieb

Lektionsleitfaden · Lab- und Prüfungsleitfaden



Lektionsleitfaden

Der letzte Mechanismus und die Policy, die alle anderen zusammenhält: ein Icon-Font-Stylesheet, das aus Default.html geladen wird, AllowHtml auf den Steuerelementen, die Markup interpretieren sollen, Icon-Glyphen in DataGridView-Zellen mit rollenbasierten Element-Klicks und eine Checkliste für den Produktivbetrieb zu Barrierefreiheit, Lizenzen, Caching, externen Abhängigkeiten und visueller Konsistenz. 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

Der letzte Mechanismus und die Policy, die alle anderen zusammenhält: ein Icon-Font-Stylesheet, das aus Default.html geladen wird, AllowHtml auf den Steuerelementen, die Markup interpretieren sollen, Icon-Glyphen in DataGridView-Zellen mit rollenbasierten Element-Klicks und eine Checkliste für den Produktivbetrieb zu Barrierefreiheit, Lizenzen, Caching, externen Abhängigkeiten und visueller Konsistenz.

1. Das Problem im Produktivbetrieb

Nach sechs Modulen kann IconDesk ein Icon auf fünf verschiedene Arten zeichnen, und jede davon funktioniert. Was dem Projekt noch fehlt, ist eine Entscheidungsregel: Wenn ein neuer Bildschirm ein Save-Symbol braucht, sollte niemand raten müssen, ob es aus dem Theme, aus einem offiziellen Wisej-4-Pack oder aus IconDesk.Icons kommt. Dieses Modul ergänzt die letzte Möglichkeit – Icon-Fonts, die als HTML gerendert werden – und schreibt dann die Policy, die sagt, wozu man greift und was vor der Auslieferung eines Bildschirms zu prüfen ist.

2. Zentrale Konzepte

Konzept	Warum es wichtig ist
Icon-Font	Eine Webschrift, deren Glyphen Bilder statt Buchstaben sind, ausgewählt über CSS-Klassennamen und vom Browser in HTML-Inhalten gezeichnet statt über eine Bildeigenschaft.
AllowHtml	Der Schalter pro Steuerelement, der Wisej.NET anweist, den Text eines Steuerelements als Markup zu interpretieren; ohne ihn erscheinen die Tags als wörtliche Zeichen, denn Text ist standardmäßig reiner Text.
Head von Default.html	Die Hostseite der Anwendung, in der ein einziger Stylesheet-Link die CSS- und Font-Dateien des Icon-Fonts für jeden Bildschirm der Session verfügbar macht.
rollenbasierter Element-Klick	Ein Klick in HTML-Inhalt kann melden, welches Inline-Element getroffen wurde – eine Grid-Zelle kann also ein Ziel für Edit und eines für Delete anbieten, ohne zwei getrennte Steuerelemente.
CDN gegenüber Selbst-Hosting	Den Link auf ein öffentliches CDN zu richten ist eine Entscheidung für den Produktivbetrieb über Verfügbarkeit, Datenschutz, Content Security Policy und Lizenzen; interne Anwendungen hosten die Dateien oft selbst.
gestufte Icon-Policy	Eine schriftliche Regel, die jeder Art von Symbol einen Ort gibt: Theme-Bilder für Semantiken, ein primäres Pack für Befehle, das eigene Pack für Produktgrafiken, Rasterdateien für Fotos.
Icon-QA-Durchgang	Ein Gang durch die fertigen Bildschirme in einem hellen und einem dunklen Theme und bei mehr als einer Anzeigeskalierung, mit der Prüfung, dass jedes Icon noch sichtbar ist und noch eine Textalternative hat.

3. Der rote Faden der Lektion

Das Stylesheet laden und Markup einschalten

Ein Icon-Font gehört überhaupt nicht zum Bildsystem von Wisej.NET. Nichts wird `Image` zugewiesen, nichts in `ImageSource` geschrieben, und der Server erzeugt nie ein PNG. Stattdessen lädt der Browser ein Stylesheet und eine Font-Datei, und ein Klassenname auf einem Inline-Element sagt der Schrift, welches Glyph sie zeichnen soll. Damit wird das Laden zu einer Angelegenheit der Seite und nicht des Steuerelements: Der Stylesheet-Link gehört in den Head der Datei `Default.html` der Anwendung, wo er einmal pro Session geholt wird und für jedes Fenster und jeden Dialog gilt, den die Benutzerin öffnet.

Wohin dieser Link zeigt, ist eine Entscheidung für den Produktivbetrieb. Ein öffentliches CDN ist bequem, aber die Oberfläche hängt dann davon ab, dass ein Dritter erreichbar ist, sie gibt die Anfragen Ihrer Benutzer an diesen Host preis, sie muss Ihre Content Security Policy passieren, und sie unterliegt der Lizenz der Icon-Bibliothek. Für eine interne Anwendung ist es oft die bessere Wahl, die CSS- und Font-Dateien selbst auszuliefern: Verfügbarkeit und Datenschutz hängen dann nur von Ihrem eigenen Deployment ab.

Mit geladenem Stylesheet erscheint noch nichts. Wisej.NET-Steuerelemente behandeln `Text` als reinen Text, das Markup kommt also maskiert an und die Benutzerin sieht spitze Klammern. `AllowHtml` auf

`true` zu setzen ändert das, und zwar bewusst pro Steuerelement: Label, Button, DataGridView-Zellen, CheckBox, RadioButton und LinkLabel sind die HTML-fähigen Beispiele. Den Schalter einzuschalten verschiebt zugleich eine Vertrauensgrenze: Ab diesem Moment ist der Text des Steuerelements Markup – halten Sie ihn also in der Hand der Anwendung und setzen Sie die Worte der Benutzerin in ein getrenntes Steuerelement mit reinem Text.

Glyphen in einer Toolbar, einer Karte und einer Grid-Zelle

In IconDesk ist die erste Icon-Font-Oberfläche eine kleine Toolbar oder Übersichtskarte. Jeder Button oder jedes Label bekommt `AllowHtml = true` und einen Textwert, der ein Inline-Element vor ein Wort stellt – genau da wirken Icon-Fonts natürlich: Das Glyph fließt mit der Beschriftung, statt in einem eigenen Bildplatz zu sitzen, und es erbt Schriftgröße und Farbe der Umgebung. Dieselbe Idee skaliert in eine `DataGridView` mit eingeschaltetem `AllowHtml`, in der eine kompakte Aktionsspalte zwei Glyphen in einer Zelle hält statt zwei eingebetteter Steuerelemente.

Eine einzelne Zelle mit zwei Symbolen funktioniert nur, wenn die Anwendung erkennen kann, welches angeklickt wurde. Dafür ist ein `role`-Attribut auf dem Inline-Element da: Der Klick meldet das getroffene Element, und der Handler verzweigt darauf. Prüfen Sie die Maus-Ereignisdaten, die Ihre Frameworkversion tatsächlich bereitstellt, bevor Sie sich darauf verlassen, und erfinden Sie keinen Vertrag, den das Ereignis nicht anbietet.

```
labelActions.AllowHtml = true;
labelActions.Text =
    "<i class='fa-solid fa-pen' role='edit'></i> " +
    "<i class='fa-solid fa-trash' role='delete'></i>";

labelActions.MouseClick += labelActions_MouseClick;
```

Kompaktheit hat ihren Preis. Eine Zelle voller Markup ist mit der Tastatur schwerer erreichbar als ein echter Button, Screenreader bekommen ein Glyph ohne Namen, und ein Icon, das nicht lädt, hinterlässt eine leere Spalte. Geben Sie jeder Aktion einen Tooltip oder ein sichtbares Wort und greifen Sie zu einer gewöhnlichen Bildspalte oder einem eingebetteten Steuerelement, sobald die Interaktion mehr ist als eine Abkürzung mit zwei Icons.

Icon-Fonts neben SVG und Icon-Packs

Keiner der Mechanismen gewinnt komplett. Das eigene System von Wisej.NET ist ressourcen- und SVG-basiert, und genau dort ist es am stärksten: Ein Quell-String kann ein Theme-Bild, einen `resource.wx`-Pfad oder ein Pack-Asset benennen; einfarbige SVGs nehmen das Suffix `?color=` mit einem Theme-Farbnamen an; der Image Selector des Designers durchsucht jedes installierte Pack; das Caching ist um diese Ressourcen herum gebaut; und jedes Steuerelement mit einer Bildeigenschaft kann sie direkt verwenden.

Icon-Fonts verdienen ihren Platz anderswo. Sie sind Webentwicklern vertraut, sie fügen sich in ein bestehendes HTML- und CSS-Designsystem ein, ihre Kataloge sind riesig, und sie stehen inline im Text auf eine Weise, die eine Bildeigenschaft nicht leisten kann. Wenn die Designsprache des Unternehmens bereits als Font-Klassen ausgedrückt ist, kostet es einen Stylesheet-Link, sie zu treffen, statt jedes Asset neu zu zeichnen.

Die ehrliche Antwort ist also, dass sie nebeneinander bestehen und die Wahl der Oberfläche gehört und nicht dem Projekt. Befehle auf Steuerelementen mit Bildeigenschaften gehören in die Ressourcen-Pipeline; Inline-Symbole im Text oder in einer dichten Grid-Zelle sind ein vernünftiger Fall für einen Icon-Font. Nicht hinnehmbar ist das Auseinanderdriften, bei dem dasselbe Delete-Konzept auf einem Bildschirm ein Pack-SVG und auf dem nächsten ein Font-Glyph ist.

Die Checkliste für den Produktivbetrieb und die IconDesk-Policy

Das eigentliche Ergebnis des Abschlussprojekts ist eine Seite Policy. Eine gestufte Regel funktioniert gut: Theme-Bilder für gängige Semantiken, ein primäres offizielles Wisej-4-Pack für die breiten Befehle, das eigene Pack IconDesk.Icons für proprietäre Grafiken, komprimierte Rasterdateien für Fotos und Icon-Fonts nur dort, wo die HTML-Integration den Aufwand wert ist. Weil die Policy für jede Situation eine erste Wahl benennt, ist ein neuer Bildschirm von Haus aus konsistent und nicht erst nach dem Review.

Darum herum steht die Checkliste, die das Projekt vor der Auslieferung eines Bildschirms abarbeitet:

- Jedes bedeutungstragende Icon hat ein sichtbares Wort, einen Tooltip oder eine andere Textalternative, und Status wird nie allein über Farbe signalisiert.
- Jede externe URL oder CDN-Referenz ist ausdrücklich freigegeben, und die Lizenzbedingungen der Bibliothek sind festgehalten.
- Icon-Font-Markup gehört der Anwendung oder ist kodiert; nicht vertrauenswürdige Werte erreichen nie ein Steuerelement mit `AllowHtml`.
- Asset-URLs sind stabil, damit der Browser-Cache arbeiten kann, und kein zufälliger Query-String hebt ihn aus.
- Der Bildschirm wurde in einem hellen und einem dunklen Theme und bei höherer Anzeigeskalierung geöffnet, und die Ergebnisse sind festgehalten.

In diesem QA-Durchgang stehen die fünf Mechanismen endlich nebeneinander. Auf der Übersichtsseite von IconDesk erscheinen ein Theme-Bild, ein Icon aus einem offiziellen Pack, ein Icon aus dem eigenen Pack, ein über `resource.wx` erreichtes eingebettetes Asset und ein umgefärbtes SVG gleichzeitig, und ein Theme-Wechsel zeigt, welche davon gefolgt sind und welche exakt so geblieben sind, wie sie gezeichnet wurden. Für jedes sichtbare Bild erklären zu können, wo es gespeichert ist und welche API es aufgelöst hat, ist die Fähigkeit, auf die dieser Kurs hingearbeitet hat.



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:

- ✓ Ein Icon-Font-Stylesheet laden und Icons mit AllowHtml in Wisej.NET-Steuerelementen rendern.
- ✓ Icon-Fonts in Inhalten von Label, Button und DataGridView verwenden und rollenbasierte Element-Klicks verstehen.
- ✓ Icon-Fonts mit den SVG- bzw. Icon-Pack-Ressourcen von Wisej.NET vergleichen.
- ✓ Eine Checkliste für den Produktivbetrieb zu Barrierefreiheit, Sicherheit, Caching, externen Abhängigkeiten und Konsistenz anwenden.

Lab- und Prüfungsleitfaden

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



Praktisches Lab

Schließen Sie das Abschlussprojekt IconDesk ab. Referenzieren Sie ein Icon-Font-Stylesheet aus Default.html, setzen Sie AllowHtml auf einer kleinen Toolbar oder Karte und rendern Sie deren Icons als Font-Glyphen; bauen Sie dann eine DataGridView-Spalte, deren Zellen Icon-Font-Aktionen enthalten, und lesen Sie aus, welches Element angeklickt wurde, damit Edit und Delete Unterschiedliches tun. Bauen Sie eine Übersichtsseite, die alle Mechanismen des Kurses auf einmal zeigt: ein Theme-Bild, ein Icon aus einem offiziellen Pack, ein Icon aus dem eigenen Pack, ein eingebettetes Logo und ein umgefärbtes SVG. Schreiben Sie anschließend die Icon-Policy des Projekts für den Produktivbetrieb und führen Sie einen QA-Durchgang in einem hellen und einem dunklen Theme durch.

Ausgangspunkt

Öffnen Sie die Projektmappe IconDesk in dem Zustand, in dem Modul 6 sie hinterlassen hat: das Befehls-Panel, die Seiten ImageLab und IconGallery, zwei offizielle Wisej-4-Icon-Packs, der eingebettete Ordner Resources, der über `resource.wx` ausgeliefert wird, und die Klassenbibliothek IconDesk.Icons mit ihrem Katalog `AppIcons`. Nichts im Projekt verwendet bislang einen Icon-Font, und die Datei `Default.html` des Projekts hat außer dem Titel noch einen leeren Head.

Vorgeschlagenes Vorgehen

- ✓ Entscheiden Sie zuerst, ob der Icon-Font aus einem CDN referenziert oder neben der Anwendung gehostet wird, und halten Sie die Gründe aus Verfügbarkeit, Datenschutz, Content Security Policy und Lizenz fest, bevor Sie den Link in `Default.html` eintragen.
- ✓ Fügen Sie eine kleine Toolbar oder Karte hinzu, setzen Sie `AllowHtml` auf jedem Label oder Button und prüfen Sie, dass ein Steuerelement ohne diesen Schalter das Markup als wörtlichen Text zeigt, damit Sie beide Verhalten gesehen haben.
- ✓ Schalten Sie `AllowHtml` für das Grid ein, bauen Sie die Aktionsspalte mit zwei Inline-Elementen und unterschiedlichen Rollen, prüfen Sie dann die Maus-Ereignisdaten, die Ihre Wisej.NET-Version bereitstellt, und verzweigen Sie Edit und Delete am getroffenen Element.
- ✓ Bauen Sie die Übersichtsseite so, dass jeder der fünf Mechanismen einmal vorkommt, beschriftet mit dem Speicherort des Bildes und der API, die es auflöst, und halten Sie ein Steuerelement pro Bildeigenschaft.
- ✓ Führen Sie den QA-Durchgang durch: Wechseln Sie zwischen einem hellen und einem dunklen Theme, probieren Sie eine höhere Anzeigeskalierung, prüfen Sie Tooltips und Tastaturerreichbarkeit der Aktionen, die nur aus einem Icon bestehen, und notieren Sie alles, was sich geändert hat oder verschwunden ist.

Geforderte Ergebnisse

- ✓ Icon-Font-Stylesheet, das aus Default.html referenziert wird, und eine Toolbar oder Karte mit AllowHtml, die Font-Glyphen rendert

- ✓ DataGridView-Spalte, deren Zellen Icon-Font-Aktionen rendern und auf das angeklickte Element reagieren
- ✓ IconDesk-Übersichtsseite mit einem Theme-Bild, einem Icon aus einem offiziellen Pack, einem Icon aus dem eigenen Pack, einem eingebetteten Asset und einem umgefärbten SVG
- ✓ Schriftliche Icon-Policy für den Produktivbetrieb zu Barrierefreiheit, Lizenzen, Caching und externen Abhängigkeiten
- ✓ QA-Durchgang in einem hellen und einem dunklen Theme mit festgehaltenen Ergebnissen

Wiederholungsfragen

- ✓ Der Stylesheet-Link steht in `Default.html` und die Klassennamen stimmen, aber ein Label zeigt die rohen Tags, während der Button daneben das Glyph zeigt. Was unterscheidet die beiden Steuerelemente, und warum gilt die Einstellung pro Steuerelement und nicht pro Anwendung?
- ✓ Ihre Aktionsspalte funktioniert, und ein Kollege möchte den Notiztext der Kundin in dieselbe Zelle setzen, damit er sich gestalten lässt. Was sagen Sie ihm, und was würden Sie stattdessen bauen?
- ✓ Welche der fünf Bilder auf der Übersichtsseite ändern sich beim Theme-Wechsel der Anwendung, und welche bleiben exakt so, wie sie gezeichnet wurden? Was sagt Ihnen das darüber, wo jedes davon gespeichert ist?



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.