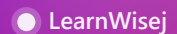


Wisej.NET-Icon-Packs und NuGet

Lektionsleitfaden · Lab- und Prüfungsleitfaden



Lektionsleitfaden

Was ein Icon-Pack wirklich ist: eine .NET-Assembly, die Hunderte SVG-Dateien als eingebettete Ressourcen trägt, die einzeln über resource.wx adressiert und vom Browser Icon für Icon zwischengespeichert werden, installiert aus den offiziellen Wisej-4-NuGet-Paketen, durchsucht im Bild-Explorer von Visual Studio und aus C# über den Icon-Katalog des Packs referenziert. 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

Was ein Icon-Pack wirklich ist: eine .NET-Assembly, die Hunderte SVG-Dateien als eingebettete Ressourcen trägt, die einzeln über resource.wx adressiert und vom Browser Icon für Icon zwischengespeichert werden, installiert aus den offiziellen Wisej-4-NuGet-Paketen, durchsucht im Bild-Explorer von Visual Studio und aus C# über den Icon-Katalog des Packs referenziert.

1. Das Problem im Produktivbetrieb

Nach drei Modulen kann IconDesk ein Bild bereits auf fünf verschiedene Arten zeigen, aber jedes Icon, das es besitzt, wurde von Hand gezeichnet oder heruntergeladen. Eine echte Anwendung braucht am ersten Tag ein Icon für Save, Delete, Search, User und Settings und bis zum Quartalsende weitere vierzig, alle in derselben visuellen Sprache. SVG-Dateien einzeln hineinzukopieren erzeugt einen Ordner voll zusammenhangloser Grafiken, uneinheitlicher Strichstärken und ohne Lizenznachweis.

2. Zentrale Konzepte

Konzept	Warum es wichtig ist
Icon-Pack	Eine gewöhnliche .NET-Assembly, deren Icon-Dateien, meist SVG, als eingebettete Ressourcen einkompiliert sind; sie zu referenzieren fügt Ihrem Deployment eine Datei hinzu, egal wie viele Icons darin stecken.
eingebettete Ressource	Eine Datei, die in die Assembly kompiliert statt daneben kopiert wird – die Icons können also beim Deployment auf einen anderen Server nicht verloren gehen, umbenannt oder vergessen werden.
resource.wx	Der Wisej.NET-Handler, der eine eingebettete Ressource über HTTP ausliefert; eine Icon-Pack-Bildquelle ist eine <code>resource.wx</code> -URL, die die Pack-Assembly und die Datei darin benennt.
Icon-Katalog	Die statische Klasse, die ein Pack veröffentlicht, damit Aufrufer IntelliSense und einen zur Compile-Zeit geprüften Namen statt eines getippten Pfads bekommen; Namespace und Klassenname unterscheiden sich von Pack zu Pack.
Anfrage pro Icon	Pack-Icons werden einzeln bei der ersten Verwendung geholt und danach aus dem Browser-Cache geliefert – das macht einen Katalog mit Tausenden Icons verkraftbar.
Vorladen der Theme-Bilder	Theme-Bilder reisen mit dem Theme und liegen beim Start der Anwendung bereits im Theme-Cache des Clients – sie verhalten sich beim Start also ganz anders als Pack-Icons.
primäre Icon-Familie	Das eine Pack, auf das sich ein Produkt für seine alltäglichen Befehle festlegt, bewusst gewählt nach Abdeckung, Stil und Lizenz statt Pack für Pack angesammelt.

3. Der rote Faden der Lektion

Ein Icon-Pack ist eine Assembly, kein Ordner

Als Erstes müssen Sie die Vorstellung ablegen, ein Icon-Pack sei ein ZIP voller Bilder. Ein Wisej.NET-Icon-Pack ist eine gewöhnliche .NET-Assembly. Ihre Icons – in den offiziellen Packs weit überwiegend SVG – sind als eingebettete Ressourcen einkompiliert, sodass ein Pack mit mehreren tausend Icons in Ihrem `bin`-Ordner trotzdem als eine einzige DLL ankommt. Nichts wird neben `Default.html` kopiert, nichts muss mit der Site veröffentlicht werden, und nichts kann beim Deployment versehentlich gelöscht werden.

Trotzdem bleibt jedes Icon einzeln adressierbar. Zur Laufzeit fordert der Browser ein Icon nach dem anderen über den Handler `resource.wx` an, der genau diese eine eingebettete Ressource aus der referenzierten Assembly liest und zurückgibt. Die Bildquelle, die Sie zuweisen, ist deshalb nur ein String: der Name des Handlers, die Assembly, in der die Ressource liegt, und die Datei darin. Weil es eine gewöhnliche HTTP-Antwort ist, achtet der Browser sie wie jede andere Datei – das zweite Steuerelement mit demselben Icon und der nächste Seitenaufruf kosten also nichts.

Diese Auslieferung pro Icon ist der Grund, warum ein Pack mit Tausenden Icons verkraftbar ist: Die Assembly zu referenzieren sendet gar nichts an den Browser, ein Icon zu verwenden sendet eine Datei. Ein benanntes Theme-Bild funktioniert umgekehrt und wird mit dem Theme vorgeladen, bevor der erste Bildschirm erscheint.

Die offiziellen Wisej-4-Pakete installieren

Wisej.NET veröffentlicht seine Icon-Familien als NuGet-Pakete, und für eine Wisej.NET-4-Anwendung brauchen Sie die Familie `Wisej-4-*`: `Wisej-4-FontAwesome`, `Wisej-4-MaterialDesign`, `Wisej-4-ModernUI`, `Wisej-4-ElegantIcons`, `Wisej-4-VisualStudioIcons`, `Wisej-4-BootstrapIcons`, `Wisej-4-VaadinIcons`, `Wisej-4-FeatherIcons`, `Wisej-4-CSSGGIcons`, `Wisej-4-TablerIcons` und `Wisej-4-IonicIcons`. Mit Wisej.NET 4.1 kommt außerdem Lucide ins Spiel; prüfen Sie im offiziellen Wisej-NuGet-Profil und in den Release Notes zu 4.1, wie es verteilt wird, statt ein nicht verwandtes Fremdpaket zu installieren, das zufällig denselben Namen trägt.

Fügen Sie sie hinzu wie jede andere Abhängigkeit, über die NuGet-Oberfläche von Visual Studio oder mit `dotnet add package Wisej-4-BootstrapIcons`. Halten Sie die Hauptlinie des Packs deckungsgleich mit der des Frameworks: Eine Wisej-4-Anwendung verwendet Wisej-4-Packs, und Beispiele aus einem älteren Wisej-2- oder Wisej-3-Paket sollten gegen das aktuelle geprüft und nicht einfach eingefügt werden. Im IconDesk-Lab referenzieren Sie zwei dieser Pakete – das sind genau zwei `PackageReference`-Einträge in der Projektdatei und zwei weitere Assemblys in der Abhängigkeitsgeschichte Ihrer Versionsverwaltung.

Ein Schritt erwischt fast jeden: Bauen Sie, nachdem Sie die Referenz hinzugefügt haben. Der Bild-Explorer zur Entwurfszeit findet Icon-Pack-Assemblys, die das Projekt tatsächlich referenziert und gegen die es kompiliert wurde – ein vor dem ersten Build geöffneter Selector listet die neue Familie schlicht nicht.

Vom Bild-Explorer zu C#

Mit den installierten Paketen erhält der Image Selector von Visual Studio die neuen Familien neben Ihren Projektbildern und den Theme-Bildern, und Sie durchsuchen und betrachten die Icons dort, statt Dateinamen zu raten. Wenn Sie eines auswählen, schreibt der Designer die Bildquelle für Sie in die generierte `Designer.cs`, und diesen erzeugten String sollten Sie mindestens einmal lesen, weil er den gesamten Mechanismus in einer Zeile zeigt.

```
// vom Designer erzeugt: Der Selector hat die resource.wx-URL für Sie geschrieben
buttonSave.ImageSource =
    "resource.wx/Wisej.Ext.FontAwesome/android.svg?color=Coral";

// aus C#: der Icon-Katalog des Packs, sodass kein Pfad von Hand getippt wird
// (Namespace und Klasse des Katalogs unterscheiden sich von Pack zu Pack)
buttonSave.ImageSource = FontAwesome.Icons.Save;

// ein einfarbiges Pack-SVG mit einem Theme-Farbnamen umfärben
buttonDelete.ImageSource =
    "resource.wx/Wisej.Ext.BootstrapIcons/trash.svg?color=error";
```

Beide Formen landen am selben Ort. Im Fachcode ist der Katalog vorzuziehen: Er bietet IntelliSense, er übersteht eine Umbenennung innerhalb des Packs, und er verhindert, dass literale Dateinamen über Handler verstreut werden, wo sie niemand prüfen kann. Das Suffix `?color=` wirkt bei Pack-SVGs genauso wie im vorigen Modul bei Ihren eigenen einfarbigen SVGs, und aus demselben Grund – eine gefüllte oder mehrfarbige Grafik ignoriert es also. Das ist eine Tatsache, die Sie im Lab an zwei echten Icons entdecken sollen, und keine Annahme für den Produktivbetrieb.

Pack, Theme-Bild oder eigenes Pack

Drei Mechanismen konkurrieren nun um denselben Button. Verwenden Sie ein **benanntes Theme-Bild**, wenn das Icon eine Standardsemantik des Frameworks darstellt und sich mit dem Theme ändern soll. Verwenden Sie ein **veröffentlichtes Icon-Pack**, wenn Sie eine breite Abdeckung gewöhnlicher Anwendungskonzepte brauchen. Bauen Sie ein **eigenes Icon-Pack der Anwendung** – das ist Modul 6 –, wenn die Symbole proprietär, markenbezogen oder eine kuratierte Auswahl sind, die Sie selbst versionieren wollen. Die Mechanismen vertragen sich problemlos; was ein Produkt kaputtmacht, ist, alle drei für dieselbe Art von Befehl wahllos zu verwenden.

Die primäre Familie zu wählen ist ein kurzes, beantwortbares Review:

- Deckt das Pack die fachlichen Konzepte ab, die das Produkt tatsächlich benennt?
- Passen seine Outline- oder Fill-Stile gut zum Theme?
- Sind die Icons einfarbige SVGs, die sich umfärben lassen?
- Passt die Lizenz des Ursprungs dazu, wie die Anwendung verteilt wird?
- Werden Designer und Entwickler die Namenskonvention wiedererkennen?

Behandeln Sie das Pack schließlich als das, was es ist: eine Abhängigkeit. Es ist eine Assembly, die Sie ausliefern – halten Sie also Quelle und Lizenz fest, belassen Sie die `PackageReference` in der Versionsverwaltung und testen Sie einen Versionswechsel vor dem Produktivbetrieb. Installieren Sie nicht jede Familie, nur weil das Installieren einfach ist. Genau deshalb endet das Lab mit einer Empfehlung von einem Absatz: Der Beleg aus dem Netzwerk-Panel – eine Anfrage pro Icon und danach ein Cache-Treffer – macht sie vertretbar.



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:

- ✓ Erklären, was ein Wisej.NET-Icon-Pack ist und wie der Browser an seine Icons kommt.
- ✓ Offizielle Wisej-4-Icon-Pack-NuGet-Pakete installieren und verwenden.
- ✓ Icon-Pack-Ressourcen im Visual Studio Designer und aus C# nutzen.
- ✓ Icon-Pack-Ressourcen mit Theme-Bildern vergleichen und den richtigen Mechanismus für eine Anwendung wählen.

Lab- und Prüfungsleitfaden

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



Praktisches Lab

Fügen Sie IconDesk zwei offizielle Wisej-4-Icon-Pack-NuGet-Pakete hinzu und bauen Sie eine Seite IconCompare, die dieselben fünf Konzepte – Save, Delete, Search, User und Settings – aus beiden Familien zeigt, sodass der optische Unterschied nicht zu übersehen ist. Wählen Sie mindestens fünf der Icons über den Bild-Explorer von Visual Studio statt den Pfad zu tippen, weisen Sie die übrigen in C# aus dem Katalog des Packs zu, färben Sie zwei davon mit dem Suffix `?color=` um und beobachten Sie im Netzwerk-Panel, wie die Icon-Dateien Anfrage für Anfrage eintreffen und danach aus dem Browser-Cache kommen. Schließen Sie mit einer kurzen Empfehlung ab, die die primäre Icon-Familie für das Projekt benennt.

Ausgangspunkt

Öffnen Sie die Projektmappe IconDesk so, wie Sie sie nach Modul 3 hinterlassen haben: Das Befehls-Panel, die Seite ImageLab und die Seite IconGallery funktionieren, und jedes Bild im Projekt ist entweder eine lokale Datei, ein Theme-Bild oder eine URL, die Sie selbst getippt haben. Das Projekt hat noch keine Icon-Pack-Referenz, der Image Selector des Designers zeigt also derzeit nur Projekt- und Theme-Bilder.

Vorgeschlagenes Vorgehen

- ✓ Wählen Sie zwei deutlich unterschiedliche offizielle Wisej-4-Familien, fügen Sie beide über die NuGet-Oberfläche oder `dotnet add package` hinzu und bauen Sie das Projekt, bevor Sie einen Selector öffnen.
- ✓ Fügen Sie eine Seite `IconCompare` mit zwei beschrifteten Spalten und einer Zeile pro Konzept hinzu: Save, Delete, Search, User und Settings.
- ✓ Füllen Sie die erste Spalte aus dem Bild-Explorer von Visual Studio, öffnen Sie dann `IconCompare.Designer.cs` und lesen Sie die `resource.wx`-Strings, die der Selector erzeugt hat.
- ✓ Füllen Sie die zweite Spalte in C# aus dem Icon-Katalog des jeweiligen Packs und färben Sie zwei Icons um, indem Sie das Suffix `?color=` mit einem Theme-Farbnamen anhängen.
- ✓ Laden Sie bei geöffnetem Netzwerk-Panel des Browsers neu, halten Sie die Anfragen pro Icon und die Cache-Treffer beim zweiten Laden fest und schreiben Sie die Empfehlung aus diesem Beleg.

Geforderte Ergebnisse

- ✓ Zwei offizielle Wisej-4-Icon-Pack-NuGet-Pakete, die IconDesk referenziert
- ✓ Seite IconCompare, die Save, Delete, Search, User und Settings aus beiden Packs zeigt
- ✓ Mindestens fünf Icons, die über den Bild-Explorer von Visual Studio ausgewählt und nicht von Hand getippt wurden
- ✓ Zwei mit dem Farbsuffix umgefärbte Icons und eine Notiz dazu, welche Grafik das Umfärben angenommen hat

- ✓ Ein Absatz Empfehlung, der die primäre Icon-Familie des Projekts benennt, mit dem Beleg aus dem Browser-Cache

Wiederholungsfragen

- ✓ Ein Pack mit dreitausend Icons zu referenzieren fügt dem Deployment eine Assembly hinzu. Was kostet das den Browser, und wann?
- ✓ Eines Ihrer beiden umgefärbten Icons hat das Farbsuffix ignoriert. Was ist das Besondere an dieser Grafik, und was würden Sie deshalb an den Auswahlkriterien für Packs ändern?
- ✓ Ein Teammitglied möchte, dass jedes Befehls-Icon dem Theme folgt, wenn die Anwendung das Theme wechselt. Welche der fünf IconCompare-Icons sollten stattdessen Theme-Bilder werden und welche nicht?



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.