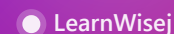


Packs de iconos de Wisej.NET y NuGet

Guía de la lección · Guía del laboratorio / examen



Guía de la lección

Qué es realmente un pack de iconos: un ensamblado .NET que lleva cientos de archivos SVG como recursos incrustados, direccionados uno a uno mediante resource.wx y almacenados en la caché del navegador icono a icono, instalado desde los paquetes NuGet oficiales de Wisej-4, explorado en el explorador de imágenes de Visual Studio y referenciado desde C# mediante el catálogo de iconos del pack. Qué significa esto para un desarrollador que conoce C# básico y ya ha trabajado algo con los controles y el diseñador de Wisej.NET — y cómo leer este módulo.



Enfoque de producción

Qué es realmente un pack de iconos: un ensamblado .NET que lleva cientos de archivos SVG como recursos incrustados, direccionados uno a uno mediante resource.wx y almacenados en la caché del navegador icono a icono, instalado desde los paquetes NuGet oficiales de Wisej-4, explorado en el explorador de imágenes de Visual Studio y referenciado desde C# mediante el catálogo de iconos del pack.

1. El problema en producción

Tres módulos después, IconDesk ya sabe mostrar una imagen de cinco formas distintas, pero todos los iconos que tiene se dibujaron o se descargaron a mano. Una aplicación real necesita un icono de Save, uno de Delete, uno de Search, uno de User y uno de Settings el primer día, y otros cuarenta antes de que acabe el trimestre, todos en el mismo lenguaje visual. Copiar archivos SVG de uno en uno produce una carpeta de ilustraciones inconexas, grosores de trazo incoherentes y ningún rastro de licencias.

2. Conceptos clave

Concepto	Por qué importa
pack de iconos	Un ensamblado .NET corriente cuyos archivos de icono, en su mayoría SVG, están compilados dentro como recursos incrustados; referenciarlo añade un solo archivo a tu despliegue por muchos iconos que contenga.
recurso incrustado	Un archivo compilado dentro del ensamblado en lugar de copiado junto a él, de modo que los iconos no se pueden perder, renombrar ni olvidar cuando la aplicación se despliega en otro servidor.
resource.wx	El controlador de Wisej.NET que sirve un recurso incrustado por HTTP; el origen de imagen de un pack de iconos es una URL <code>resource.wx</code> que nombra el ensamblado del pack y el archivo que hay dentro.
catálogo de iconos	La clase estática que publica un pack para que quien la usa obtenga IntelliSense y un nombre comprobado en compilación en lugar de una ruta escrita a mano; el espacio de nombres y el nombre exacto de la clase varían de pack a pack.
petición por icono	Los iconos de un pack se obtienen de forma individual, en el primer uso, y después se sirven desde la caché del navegador, que es lo que hace asumible un catálogo de miles de iconos.
precarga de imágenes de tema	Las imágenes de tema viajan con el tema y ya están en la caché de temas del cliente cuando arranca la aplicación, así que se comportan de forma muy distinta a los iconos de un pack en el arranque.
familia de iconos principal	El único pack al que se compromete un producto para sus comandos del día a día, elegido de forma deliberada por cobertura, estilo y licencia en lugar de acumulado pack a pack.

3. Desarrollo de la lección

Un pack de iconos es un ensamblado, no una carpeta

Lo primero que hay que desaprender es la idea de que un pack de iconos es un zip de imágenes. Un pack de iconos de Wisej.NET es un ensamblado .NET normal. Sus iconos, en los packs oficiales SVG de forma abrumadora, están compilados dentro como recursos incrustados, así que un pack con varios miles de iconos sigue llegando a tu carpeta `bin` como una única DLL. No se copia nada junto a `Default.html`, no hay que publicar nada con el sitio y nada se puede borrar por accidente durante un despliegue.

Aun así, cada icono sigue siendo direccionable de forma individual. En tiempo de ejecución el navegador pide un icono cada vez mediante el controlador `resource.wx`, que lee ese único recurso incrustado del ensamblado referenciado y lo devuelve. Por tanto, el origen de imagen que asignas no es más que una cadena: el nombre del controlador, el ensamblado en el que vive el recurso y el archivo que hay dentro. Como es una respuesta HTTP corriente, el navegador la cachea como cualquier otro archivo, así que el segundo control que use ese mismo icono, y la siguiente carga de página, no cuestan nada.

Esa entrega icono a icono es la razón por la que un pack de miles de iconos resulta asumible: referenciar el ensamblado no envía nada al navegador, usar un icono envía un archivo. Una imagen de tema con nombre

funciona al revés, precargada con el tema antes de que se pinte la primera pantalla.

Instalar los paquetes oficiales de Wisej-4

Wisej.NET publica sus familias de iconos como paquetes NuGet, y para una aplicación Wisej.NET 4 quieres la familia `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` y `Wisej-4-IonicIcons`. Wisej.NET 4.1 añade además Lucide al panorama; confirma cómo se distribuye consultando el perfil oficial de NuGet de Wisej y las notas de la versión 4.1, en lugar de instalar un paquete de terceros sin relación que casualmente comparta el nombre.

Añádelos como añades cualquier dependencia, desde la interfaz de NuGet de Visual Studio o con `dotnet add package Wisej-4-BootstrapIcons`. Mantén la línea principal del pack alineada con la del framework: una aplicación Wisej 4 usa packs Wisej-4, y los ejemplos copiados de un paquete antiguo de Wisej 2 o Wisej 3 deben revisarse contra el actual en lugar de pegarse tal cual. En el laboratorio de IconDesk referenciarás dos de estos paquetes, que son exactamente dos entradas `PackageReference` en el archivo del proyecto y dos ensamblados más en la historia de dependencias del control de versiones.

Hay un paso que casi todo el mundo se salta: compilar después de añadir la referencia. El explorador de imágenes en tiempo de diseño descubre los ensamblados de packs de iconos que el proyecto referencia de verdad y contra los que ha compilado, así que un selector abierto antes de la primera compilación sencillamente no mostrará la nueva familia.

Del explorador de imágenes a C#

Con los paquetes ya instalados, el selector de imágenes de Visual Studio incorpora las nuevas familias junto a las imágenes de tu proyecto y las del tema, y exploras y previsualizas los iconos ahí en lugar de adivinar nombres de archivo. Cuando eliges uno, el diseñador escribe por ti el origen de imagen en el `Designer.cs` generado, y merece la pena leer al menos una vez esa cadena generada porque muestra todo el mecanismo en una sola línea.

```
// designer-generated: the picker wrote the resource.wx URL for you
buttonSave.ImageSource =
    "resource.wx/Wisej.Ext.FontAwesome/android.svg?color=Coral";

// from C#: the pack's own icon catalog, so no path is typed by hand
// (the exact catalog namespace and class vary from pack to pack)
buttonSave.ImageSource = FontAwesome.Icons.Save;

// recolour a monochrome pack SVG with a theme colour name
buttonDelete.ImageSource =
    "resource.wx/Wisej.Ext.BootstrapIcons/trash.svg?color=error";
```

Ambas formas acaban en el mismo sitio. El catálogo es el que conviene preferir en el código de negocio: da IntelliSense, sobrevive a un cambio de nombre dentro del pack e impide que los nombres de archivo literales se repartan por manejadores donde nadie puede auditarlos. El sufijo `?color=` funciona con los SVG de los packs exactamente igual que con tus propios SVG monocromos del módulo anterior, y

funciona por la misma razón, así que una ilustración rellena o multicolor lo ignorará. Eso es un hecho que hay que descubrir en el laboratorio con dos iconos reales, no una suposición que llevarse a producción.

Pack, imagen de tema o tu propio pack

Ahora hay tres mecanismos compitiendo por el mismo botón. Usa una **imagen de tema con nombre** cuando el icono represente una semántica estándar del framework y deba cambiar cuando cambie el tema. Usa un **pack de iconos publicado** cuando necesites una cobertura amplia de los conceptos corrientes de una aplicación. Construye un **pack de iconos de la aplicación**, que es el Módulo 6, cuando los símbolos sean propios, de marca o un subconjunto seleccionado que quieras versionar tú mismo. Los mecanismos conviven sin problema; lo que rompe un producto es usar los tres al azar para el mismo tipo de comando.

Elegir la familia principal es una revisión corta y con respuesta:

- ¿Cubre el pack los conceptos de negocio que el producto nombra de verdad?
- ¿Sus estilos de contorno o de relleno encajan bien junto al tema?
- ¿Son los iconos SVG monocromos que aceptan el recolorado?
- ¿Encaja la licencia de origen con la forma en que se distribuye la aplicación?
- ¿Reconocerán diseñadores y desarrolladores la convención de nomenclatura?

Por último, trata el pack como la dependencia que es. Es un ensamblado que distribuyes, así que registra su origen y su licencia, mantén el `PackageReference` en el control de versiones y prueba un cambio de versión antes de producción. No instales todas las familias porque instalar sea fácil. El laboratorio se cierra con una recomendación de un párrafo justo por eso: la prueba del panel de red, una petición por icono y después un acierto de caché, es lo que la hace defendible.



Nombra primero el mecanismo

Antes de asignar cualquier imagen, di qué mecanismo estás usando: un objeto `System.Drawing.Image` que el servidor renderiza como PNG, una cadena de origen de imagen que resuelve el navegador, una imagen de tema con nombre, un recurso incrustado servido por `resource.wx` o una fuente de iconos dibujada como HTML. Asigna una sola propiedad de imagen por control para que las demás no puedan restablecerla en silencio, mantén la cadena de origen con significado en lugar de una ruta mágica pegada, prefiere SVG para los iconos de interfaz para que se mantengan nítidos en cualquier tamaño y escala, y dale una alternativa textual a todo icono que transmita significado.



Objetivos de aprendizaje

Al terminar este módulo serás capaz de:

- ✓ Explicar qué es un pack de iconos de Wisej.NET y cómo obtiene el navegador sus iconos.
- ✓ Instalar y usar los paquetes NuGet oficiales de packs de iconos Wisej-4.
- ✓ Usar los recursos de un pack de iconos en el diseñador de Visual Studio y desde C#.
- ✓ Comparar los recursos de un pack de iconos con las imágenes de tema y elegir el mecanismo adecuado para una aplicación.

Guía del laboratorio / examen

Lo que construirás en el laboratorio práctico, el enfoque sugerido y los entregables requeridos.



Laboratorio práctico

Añade a IconDesk dos paquetes NuGet oficiales de packs de iconos Wisej-4 y construye una página IconCompare que muestre los mismos cinco conceptos, Save, Delete, Search, User y Settings, tomados de ambas familias para que la diferencia visual sea imposible de pasar por alto. Elige al menos cinco de los iconos con el explorador de imágenes de Visual Studio en lugar de escribir la ruta, asigna el resto desde el catálogo del pack en C#, recolorea dos de ellos con el sufijo `?color=` y observa el panel de red para ver cómo los archivos de icono llegan petición a petición y después vienen de la caché del navegador. Cierra con una recomendación breve que nombre la familia de iconos principal del proyecto.

Punto de partida

Abre la solución IconDesk tal como la dejaste después del Módulo 3: el panel de comandos, la página ImageLab y la página IconGallery funcionan, y todas las imágenes del proyecto son un archivo local, una imagen de tema o una URL que escribiste tú. El proyecto todavía no referencia ningún pack de iconos, así que el selector de imágenes del diseñador muestra por ahora solo imágenes del proyecto y del tema.

Enfoque sugerido

- ✓ Elige dos familias oficiales de Wisej-4 que contrasten entre sí, añade ambas con la interfaz de NuGet o con `dotnet add package` y compila el proyecto antes de abrir ningún selector.
- ✓ Añade una página `IconCompare` con dos columnas etiquetadas y una fila por concepto: Save, Delete, Search, User y Settings.
- ✓ Rellena la primera columna desde el explorador de imágenes de Visual Studio, después abre `IconCompare.Designer.cs` y lee las cadenas `resource.wx` que generó el selector.
- ✓ Rellena la segunda columna en C# desde el catálogo de iconos de cada pack y recolorea dos iconos añadiendo el sufijo `?color=` con un nombre de color del tema.
- ✓ Recarga con el panel de red del navegador abierto, anota las peticiones por icono y los aciertos de caché en la segunda carga, y escribe después la recomendación a partir de esa prueba.

Entregables requeridos

- ✓ Dos paquetes NuGet oficiales de packs de iconos Wisej-4 referenciados por IconDesk
- ✓ Página IconCompare que muestra Save, Delete, Search, User y Settings de ambos packs
- ✓ Al menos cinco iconos elegidos con el explorador de imágenes de Visual Studio en lugar de escritos a mano
- ✓ Dos iconos recoloreados con el sufijo de color y una nota sobre qué ilustración aceptó el recoloreado
- ✓ Recomendación de un párrafo que nombre la familia de iconos principal del proyecto, con las pruebas de caché del navegador que la respaldan

Preguntas de repaso

- ✓ Referenciar un pack de tres mil iconos añade un ensamblado al despliegue. ¿Qué le cuesta eso al navegador, y cuándo?
- ✓ Uno de tus dos iconos recoloreados ignoró el sufijo de color. ¿Qué tiene esa ilustración y qué cambiarías en los criterios de selección del pack por ese motivo?
- ✓ Un compañero quiere que todos los iconos de comando sigan al tema cuando la aplicación cambia de tema. ¿Cuáles de los cinco iconos de IconCompare deberían pasar a ser imágenes de tema y cuáles no?



Nombra primero el mecanismo

Antes de asignar cualquier imagen, di qué mecanismo estás usando: un objeto `System.Drawing.Image` que el servidor renderiza como PNG, una cadena de origen de imagen que resuelve el navegador, una imagen de tema con nombre, un recurso incrustado servido por `resource.wx` o una fuente de iconos dibujada como HTML. Asigna una sola propiedad de imagen por control para que las demás no puedan restablecerla en silencio, mantén la cadena de origen con significado en lugar de una ruta mágica pegada, prefiere SVG para los iconos de interfaz para que se mantengan nítidos en cualquier tamaño y escala, y dale una alternativa textual a todo icono que transmita significado.