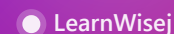


Recursos incrustados y resource.wx

Guía de la lección · Guía del laboratorio / examen



Guía de la lección

Distribuir imágenes dentro del ensamblado: la Build Action establecida en Embedded Resource, el controlador resource.wx con y sin calificador de ensamblado, las extensiones css, js, html, jpg, jpeg, png, gif, svg y bmp que puede servir, y el archivo desplegado que puede sobrescribir un recurso incrustado sin recompilar. 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

Distribuir imágenes dentro del ensamblado: la Build Action establecida en Embedded Resource, el controlador resource.wx con y sin calificador de ensamblado, las extensiones css, js, html, jpg, jpeg, png, gif, svg y bmp que puede servir, y el archivo desplegado que puede sobrescribir un recurso incrustado sin recompilar.

1. El problema en producción

Hasta ahora, todos los recursos de IconDesk han sido algo que la aplicación encontró en disco o le pidió al tema. Eso funciona hasta que alguien despliega el sitio y se olvida de una carpeta: el navegador pide un icono, el servidor responde 404 y el usuario ve un rectángulo vacío donde debería estar el distintivo de estado. Nadie lo detecta en la revisión, porque la máquina del desarrollador sigue teniendo el archivo.

Incrustar convierte la ilustración en parte de la salida de compilación: la imagen viaja con el ensamblado, versionada con el mismo número de compilación, imposible de perder en un paso de copia. Es además el mecanismo sobre el que está construido todo pack de iconos de Wisej.NET, que es lo que hace posible el Módulo 6.

2. Conceptos clave

Concepto	Por qué importa
Build Action = Embedded Resource	La propiedad de archivo de Visual Studio que compila un archivo dentro del manifiesto del ensamblado en lugar de copiarlo junto al ejecutable, de modo que el recurso se distribuye dentro de la DLL.
controlador resource.wx	El controlador HTTP de Wisej.NET que sirve recursos incrustados al navegador; lo que asignas es una cadena de origen de imagen, nunca un objeto de imagen del lado del servidor.
URL de recurso sin calificar	<code>resource.wx/company-logo.svg</code> le pide a Wisej.NET que busque en los ensamblados cargados un recurso con ese nombre, lo cual es cómodo y ligeramente ambiguo.
URL de recurso calificada con el ensamblado	<code>resource.wx/MyCompany.Branding/company-logo.svg</code> nombra el ensamblado (sin <code>.dll</code>) para que la búsqueda no pueda caer en el archivo de otro con el mismo nombre.
Tipos de recurso permitidos	El controlador documenta exactamente nueve extensiones: <code>.css</code> , <code>.js</code> , <code>.html</code> , <code>.jpg</code> , <code>.jpeg</code> , <code>.png</code> , <code>.gif</code> , <code>.svg</code> y <code>.bmp</code> ; cualquier otra cosa vuelve como un 404.
Sobrescritura por archivo desplegado	Un archivo desplegado con la aplicación se puede servir en lugar del recurso incrustado en la misma ruta, lo que permite a una instalación cambiar la marca o parchear una ilustración sin recompilar el ensamblado.
Nombre de recurso del manifiesto	El nombre que .NET almacena realmente, derivado del nombre del ensamblado, el espacio de nombres predeterminado y la carpeta en la que está el archivo; adivinarlo es la razón habitual de que una URL de recurso no devuelva nada.

3. Desarrollo de la lección

Incrustar es una acción de compilación, no una llamada a una API

No hay ningún método al que llamar ni ningún atributo con el que decorar. Añades el archivo al proyecto, abres la ventana Properties y cambias Build Action de su valor predeterminado a **Embedded Resource**. El compilador escribe los bytes en el manifiesto del ensamblado, el archivo deja de aparecer en la carpeta de salida y a partir de ahí el recurso forma parte del binario. El laboratorio lo hace cinco veces: `logo.svg`, `status-ok.svg`, `status-warning.svg`, `photo.png` y `badge.gif`, todos dentro de una carpeta `Resources` de la aplicación IconDesk.

La misma propiedad funciona en una biblioteca de clases, y ese es el caso interesante. Una biblioteca cuyo único trabajo es llevar ilustraciones es un saco de iconos redistribuible: la referencias y todas las pantallas que la referencien pueden direccionar esos archivos. Los packs de iconos de Wisej.NET son exactamente esto, varios cientos de SVG en un ensamblado, y en el Módulo 6 construirás el tuyo. La técnica no cambia en nada entre una aplicación y una biblioteca; lo único que cambia es el nombre que quizá tengas que poner en la URL.

Compáralo con las alternativas. Un archivo Content con *Copy to Output Directory* queda suelto: fácil de sustituir, fácil de perder. Una URL externa traslada el problema a otro servidor. Incrusta cuando el recurso pertenece al código y debe versionarse con él; la decisión es operativa, no estilística.

Direccionarlo: resource.wx con y sin nombre de ensamblado

Una vez incrustado el archivo, el navegador llega a él mediante el controlador `resource.wx`. El valor que asignas a `ImageSource` es una cadena, y el cliente la resuelve como una URL; el servidor nunca materializa un `System.Drawing.Image` para ella. Esa distinción importa sobre todo con SVG, que no es en absoluto un formato de `System.Drawing`: es marcado que renderiza el navegador, que es precisamente por lo que se mantiene nítido a 16, 24 o 96 píxeles.

```
// IconDesk - ResourceLab.cs
// every file below has Build Action = Embedded Resource

// unqualified: Wisej.NET searches the loaded assemblies
picStatusOk.ImageSource = "resource.wx/status-ok.svg";
picPhoto.ImageSource    = "resource.wx/photo.png";
picBadge.ImageSource    = "resource.wx/badge.gif";

// qualified with the assembly name, no ".dll" - no ambiguity
picLogo.ImageSource     = "resource.wx/IconDesk/logo.svg";

// the same URL works outside control properties, e.g. in CSS:
// .brand-mark { background-image: url('resource.wx/IconDesk/logo.svg'); }
```

Deja fuera el segmento del ensamblado y Wisej.NET buscará en los ensamblados cargados un recurso coincidente. Eso está bien mientras un solo ensamblado sea dueño del nombre. Añade una segunda biblioteca de iconos, o el ensamblado de marca de un cliente, y dos archivos llamados `logo.svg` podrán responder. Calificar la ruta elimina la duda y documenta la propiedad para quien lea después. El laboratorio te pide calificar exactamente un origen para que la diferencia se vea en tu propio código.

Como el controlador habla HTTP corriente, la misma forma de URL puede aparecer en una hoja de estilos, en un fragmento de HTML o en un trozo de JavaScript, no solo en una propiedad de imagen. Ten en cuenta además que el atributo `[assembly: WisejResources]` pertenece a una historia más amplia de descubrimiento y agrupación de scripts, estilos y temas incrustados; una URL de recurso funciona con él o sin él, así que no añadas atributos de forma mecánica solo para mostrar una imagen.

Qué sirve y qué no sirve el controlador

La lista documentada es corta y merece la pena memorizarla, porque el modo de fallo es silencioso. El controlador sirve `.css`, `.js`, `.html`, `.jpg`, `.jpeg`, `.png`, `.gif`, `.svg` y `.bmp`. Seis de las nueve son imágenes; las otras tres existen porque el ensamblado de una biblioteca suele llevar una hoja de estilos o un script junto a sus ilustraciones. Todo lo demás —un archivo de icono, una fuente, un PDF— vuelve como un 404 por muy correctamente que se incrustara.

Así que la extensión es una decisión que se toma antes de que el archivo entre en el proyecto: un logotipo vectorial pasa a ser `.svg`, una fotografía `.png` o `.jpg`, un marcador animado `.gif`. Si alguien te entrega un formato que el controlador no sirve, convertirlo forma parte del trabajo.

Cuándo gana un archivo desplegado y cómo diagnosticar un 404

Wisej.NET permite que un archivo desplegado con la aplicación tenga prioridad sobre el recurso incrustado en la misma ruta lógica. Esto es deliberado y útil: la instalación de un cliente puede colocar su propio `logo.svg` y la aplicación cambia de marca sin que nadie recompile ni vuelva a publicar el ensamblado de recursos, y la misma vía sirve para parchear un icono equivocado. El laboratorio te pide hacer una sobrescritura y capturar el antes y el después.

El riesgo es que el directorio de despliegue pasa a formar parte de tu frontera de confianza: cualquiera que pueda escribir archivos ahí puede cambiar lo que muestra la aplicación. El segundo riesgo es la deriva, porque la aplicación en ejecución ya no coincide con el ensamblado con el que se compiló. Trata la capacidad de sobrescritura como configuración, con un responsable y un registro.

Cuando una imagen no aparece, no adivines nombres del manifiesto. Recorre en su lugar una lista fija; cada paso descarta una causa:

- Confirma que Build Action es Embedded Resource, no Content ni None.
- Confirma que la extensión es una de las nueve que sirve el controlador.
- Confirma que el ensamblado está realmente cargado en el proceso en ejecución.
- Confirma el ensamblado y los segmentos de la ruta de la URL, carácter a carácter.
- Comprueba si un cambio de espacio de nombres predeterminado, de carpeta o de nombre lógico ha movido el nombre de recurso del manifiesto.
- Recompila, pide después la URL exacta en el navegador y lee el código de estado.



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:

- ✓ Incrustar los archivos de imagen admitidos en el ensamblado de una aplicación o de una biblioteca de clases.
- ✓ Usar `resource.wx` con y sin nombre de ensamblado.
- ✓ Enumerar los tipos de recurso que admite el controlador de recursos incrustados de Wisej.NET.
- ✓ Explicar cómo los archivos desplegados pueden sobrescribir recursos incrustados y por qué eso puede ser útil.

Guía del laboratorio / examen

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



Laboratorio práctico

Añade a la aplicación IconDesk una carpeta Resources con logo.svg, status-ok.svg, status-warning.svg, photo.png y badge.gif, y marca todos ellos con Build Action Embedded Resource. Construye una página ResourceLab que muestre los cinco mediante orígenes de imagen resource.wx, escribe uno de esos orígenes con el nombre del ensamblado de la aplicación delante del archivo para que la búsqueda no pueda ser ambigua, y deja el resto sin calificar. Después coloca un archivo de reemplazo junto a la aplicación desplegada para sobrescribir un recurso incrustado y captura el antes y el después. Termina escribiendo mal a propósito una URL de recurso y anotando los pasos que identificaron el fallo.

Punto de partida

Abre IconDesk tal como está después del Módulo 4: el panel de comandos, la página ImageLab, la IconGallery dirigida desde el diseñador y la página IconCompare que pone dos packs de iconos oficiales uno al lado del otro. Hasta ahora todos los recursos llegan del tema, de un archivo suelto del proyecto, de una URL o del ensamblado de un pack; todavía no se distribuye nada dentro del propio binario de IconDesk, y no hay ninguna carpeta **Resources** en el proyecto.

Enfoque sugerido

- ✓ Añade primero la carpeta **Resources** y los cinco archivos, después establece Build Action en Embedded Resource en cada uno y recompila antes de escribir nada de C#.
- ✓ Añade una página **ResourceLab** con cinco controles PictureBox o Label y asigna a cada **ImageSource** una cadena **resource.wx**, calificando solo el logotipo con el nombre de ensamblado **IconDesk**.
- ✓ Ejecuta la página, abre el panel de red del navegador y comprueba que cada recurso devuelve 200 y que los SVG siguen nítidos al ampliarlos.
- ✓ Publica o copia la aplicación a una carpeta de despliegue, coloca un archivo de reemplazo sobre uno de los recursos incrustados, recarga sin recompilar y haz una captura de ambos estados.
- ✓ Rompe una URL a propósito, recorre después la lista de resolución de problemas en orden y anota qué paso reveló la causa.

Entregables requeridos

- ✓ Carpeta Resources con logo.svg, status-ok.svg, status-warning.svg, photo.png y badge.gif, todos marcados como Embedded Resource
- ✓ Página ResourceLab que muestra cada recurso incrustado mediante un origen de imagen resource.wx
- ✓ Un origen de imagen calificado con el nombre del ensamblado de la aplicación y el resto sin calificar
- ✓ Archivo desplegado que sobrescribe un recurso incrustado, capturado antes y después

- ✓ Nota de resolución de problemas con los pasos que identificaron una URL de recurso mal escrita

Preguntas de repaso

- ✓ Dos ensamblados referenciados incrustan ambos un archivo llamado `logo.svg`. ¿A qué se resuelve un origen sin calificar y qué cambiarías para que la respuesta deje de depender del orden de carga?
- ✓ Tu sobrescritura funciona en el servidor de pruebas pero no después de la siguiente versión. ¿Qué partes del proceso de despliegue podrían haberla eliminado y dónde debería haberse registrado?
- ✓ Un compañero incrusta `icon.ico` e informa de que resource.wx está roto. ¿Qué está pasando en realidad y cuál es el cambio más pequeño que lo arregla?



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.