Brindando un nombre seguro a un ensamblado

Hoy en día la mayor parte de los ciudadanos del mundo cuentan con un documento de identidad, el cual les permite identificarse. Usted no es una excepción, ya que cuando realiza algún tipo de trámite, dicha información es vital para acreditar que usted es quien dice ser. Otra u otras personas podrían tener su mismo nombre, apellido, y hasta en algunos casos similar apariencia, pero usted seguirá siendo una persona única[3] y totalmente identificable. Toda tarea que realice será identificada por los demás en forma exclusiva, y si bien este libro no intenta ahondar sobre aspectos más profundos que los que expone Visual Basic .NET, la comprensión de esta característica le ayudará a entender algunos conceptos que utilizaremos a continuación. Podemos decir entonces que su nacionalidad y documento de identidad lo hacen único en el mundo, como si fuese un identificador o nombre global.

Algo similar pasa con los ensamblados que desean ser parte de esta «comunidad» global. Un ensamblado privado no puede ser compartido por el sencillo hecho de que no cuenta con una identidad única, o lo que es igual, no aporta la suficiente información para poder ser identificado de forma exclusiva. Dos ensamblados podrían tener el mismo nombre, tamaño, etc., y CLR podría no saber cuál de ellos utilizar. Esto es lo suficientemente grave como para que la plataforma .NET tenga que proveer un mecanismo adicional para darle a los mismos un nombre o identificador único y seguro.

Como vimos anteriormente, cada un nuevo ensamblado factible de ser compartido es expuesto a muchos riesgos, como el que existan otros ensamblados con iguales características, lo que podría no permitir a CLR identificar con certeza cuál de ellos emplear. Pero entonces...

¿Cómo se podría identificar cada uno de ellos en forma única y segura?

Bien, la respuesta es similar a lo que hablamos al comienzo, mediante la utilización de un identificador/nombre único o global. El primer paso para darle la posibilidad a un ensamblado de que sea compartido es brindarle un nombre de este tipo, el cual hará posible que el mismo pueda diferenciarse de los demás aunque existan otros con similares características.

Visual Studio podría generar un número al azar teniendo en cuenta varios factores únicos, como el número de placa de red, la fecha y hora, etc., y así lograr un valor al azar, el cual podría ser considerado exclusivo, y posteriormente guardarlo dentro del ensamblado en el momento de su compilación. Esta aproximación no está errada, y de hecho fue la técnica utilizada por los componentes creados en versiones anteriores. No obstante, la plataforma .NET emplea una tecnología aún más interesante, la cual permite que una vez que una versión de ensamblado compartido sea compilada, solamente quien disponga de cierta información pueda gestar nuevas versiones del mismo. Para ello hace uso de una característica muy conocida denominada criptografía de clave pública.

La criptografía de clave pública es una tecnología que utiliza algoritmos sofisticados para encriptar un conjunto de datos mediante el uso de claves. A diferencia de los métodos estándares de cifrado, en ésta se debe emplear una clave para cifrar la información (clave pri-

---

[3] Por supuesto que las características personales también identifican a una persona de forma única, pero eso no nos es de utilidad en este momento.

vada) y otra para descifrar la misma (clave pública). De acuerdo con esta aproximación, la clave privada es utilizada para cifrar los datos, los cuales podrán ser únicamente descifrados por la clave pública. Ambas claves constan de un conjunto de letras y números. De esta forma sólo existe una clave pública capaz de descifrar información cifrada con la privada. Pero...

¿Qué relación mantiene todo esto con los ensamblados compartidos?

Vimos anteriormente que podían existir dos ensamblados con características similares, como nombre, tamaño, versión, etc., por lo que era esencial el poder identificar a cada uno de estos en forma exclusiva. Para ello se debía proveer algún tipo de dato dentro de cada uno de ellos que fuese considerado único, y que esto fuese válido sin importar el número de ensamblados bajo el cual el mismo estuviese expuesto. Para dicho fin la plataforma .NET ofrece la posibilidad de crear un nombre único global (o seguro), el cual es internamente usado para identificar al componente en forma exclusiva, en vez de hacer uso de su nombre u otra característica que puede ser repetible. Básicamente, un nombre único es la cédula de identidad de un ensamblado, o lo que es igual, un conjunto de valores que siguen cierta secuencia única que lo identifica. Veremos a continuación como es almacenada esta información.

Asegurando un nombre único

Para generar un nombre o identificador único se conjugan varios datos, como el nombre programático del ensamblado, la versión del mismo, la cultura, una clave pública que deberá ser generada previamente y un valor de verificación o *hash*, el cual tendrá que ser calculado empleando el contenido del ensamblado y la lista de componentes de los cuales éste depende. Por último, el resultado será cifrado utilizando una clave privada (correspondiente al par de clave pública-privada), y almacenado en el manifiesto, en una sección que posteriormente no será utilizada para recalcular el valor de verificación o *hash*. A esto se le llama nombre seguro (*Strong Name*), y es el dato que internamente emplea la infraestructura para identificar a un componente. Cuando se compila un ensamblado, el compilador se encargará de guardar en el manifiesto la información sobre las bibliotecas que éste necesita para ejecutarse correctamente. Para ello, el compilador se encargará de grabar la referencia a éstos, y adicionalmente el nombre único generado anteriormente para cada uno de ellos[4], a los efectos de que posteriormente pueda ser localizado empleando dicha información. Debido a la cantidad de factores que se utilizan para generar el nombre único, la posibilidad de que el mismo se repita es prácticamente imposible.

En tiempo de ejecución, el CLR leerá dicha marca e intentará buscar un ensamblado con igual nombre y características. Este proceso involucra también tareas de validación, a los efectos de corroborar que el mismo no haya sido adulterado. Básicamente, el valor de validación o *hash* es calculado nuevamente, y comprobado mediante el uso de la clave pública para descifrar el mismo. Si ambas marcas son iguales, entonces el ensamblado será cargado, de lo contrario se generará una excepción.

[4] La clave no es almacenada en su totalidad para que el resultado final no sea demasiado extenso, pero la parte guardada es suficiente para que no existan conflictos. Puede verse dicha información mediante la herramienta *ILdasm* bajo la entrada *PublicToken* (símbolo de clave pública) del manifiesto.

Como vemos, la utilización de las técnicas de cifrado por clave pública ofrece una ventaja adicional, la cual radica en que sólo existirá una clave pública capaz de descifrar datos cifrados con una clave privada, por lo que solamente quien disponga de la misma podrá generar nuevas versiones del componente. Si se intentase utilizar un conjunto de claves diferentes para construir una versión posterior de un ensamblado, el mismo daría como resultado uno diferente desde el punto de vista de CLR, sin ninguna relación con el anterior. Debido a que se utilizan varios factores para generar un nombre seguro, es posible utilizar una misma clave para generar los nombres de un conjunto de ensamblados.

¿Cómo se genera un nombre seguro?

Para dar nombre a un ensamblado se necesita obtener un conjunto de claves, a los efectos de realizar el proceso citado anteriormente. El par de claves empleadas son almacenadas en archivos físicos, por lo que es relativamente fácil su posterior manipulación y guardado. Debido a ello, sería posible restringir la creación de nuevas versiones de un ensamblado, simplemente limitando el acceso a las mismas.

Esta tecnología asegura que quien implemente una nueva versión sea la misma entidad que realizó las anteriores, pero no afirma que la empresa sea confiable o tenga buenas intenciones. Para ello se utiliza un proceso llamado «firmado», el cual no veremos en este capítulo, pero podrá encontrar más información acerca del mismo en las ayudas del producto.

Si bien anteriormente detallamos los procesos necesarios para lograr un nombre seguro (*Strong Name*), la mayor parte de los mismos son totalmente transparentes al desarrollador, y han sido aquí presentados con el fin de explicar el alcance de esta tecnología. Vamos entonces a ver los pasos necesarios desde el punto de vista de un desarrollador, los cuales imagino que querrá conocer:

1. Obtener un par de claves (pública y privada).
2. Guardar el par de claves en un lugar seguro.
3. Darle nombre al ensamblado utilizando el par de claves.

La obtención de un par de claves se realiza generalmente una sola vez en la vida de un ensamblado, y esto incluye también sus posteriores versiones. Para conseguir las mismas no es necesario pagarle a ninguna entidad, ni realizar la solicitud a compañía alguna. Visual Studio incluye una nueva herramienta, la cual genera un archivo conteniendo dos claves (pública y privada) generadas al azar. Si bien el proceso puede resultar a simple vista complejo, veremos que es más sencillo de lo que inicialmente aparenta. Vamos entonces a convertir el ensamblado privado que construimos en la sección anterior de este capítulo (*MensajeAlMundo.dll*) en uno compartido, y para dicho fin obtendremos en primera instancia un par de claves. Para ello, Visual Studio cuenta con una herramienta llamada *Sn.exe* (*Strong Name*), la cual genera un archivo con la información mencionada en el punto 1.

Sintaxis:
Sn <opción> <parámetros>

Sn –k <archivo de salida>

Como vemos, se le debe suministrar la ruta en donde depositar la información, adicionando el parámetro *k*. El siguiente ejemplo genera un par de claves para un archivo llamado *claves.key*.

Sn –k claves.key

El resultado tendrá 596 bytes, y contendrá toda la información necesaria para generar uno o varios nombres únicos.

```
Volume in drive K is VS.NET
 Volume Serial Number is A8E4-FAAA

 Directory of K:\PROGRA~1\MICROS~1.NET\FRAMEW~1\Bin

22/10/2001  12:38a                596 claves.key
               1 File(s)            596 bytes
               0 Dir(s)  11.318.988.800 bytes free
```

Es recomendable siempre que el próximo paso sea respaldar este archivo en algún lugar seguro.

A continuación vamos a indicar cuál será el ensamblado al que le daremos nombre empleando el par de claves. Esta tarea debe realizarse desde el entorno de desarrollo, utilizando una etiqueta llamada *AssemblyKeyFile*, la cual es generalmente incluida dentro del archivo de configuración de proyecto *AssemblyInfo.vb*. La misma toma la ruta al documento conteniendo las claves, que serán posteriormente utilizadas por el compilador para escribir el nombre seguro (*Strong Name*) en el manifiesto del ensamblado.

Sintaxis:

AssemblyKeyFile(<ruta al archivo conteniendo las claves> As String)

```
<Assembly: AssemblyTitle("Mensaje del planeta tierra")>
<Assembly: AssemblyDescription("Envia un mensaje de los terrícolas")>
<Assembly: AssemblyCompany("")>
<Assembly: AssemblyProduct("")>
<Assembly: AssemblyCopyright("(c) Corp. Tierra Unida SRL")>
<Assembly: AssemblyTrademark("")>
<Assembly: AssemblyKeyFile("C:\MiRuta\Claves.key")>
```

En el momento de la compilación se detectará la etiqueta, y se comprobará si la ruta y el documento existen, y de ser así se gestará y escribirá en la sección de manifiesto el nombre seguro (*Strong Name*).

Si observamos el mismo mediante la herramienta *ILdasm.exe* después de la compilación, podremos apreciar que se incluye una nueva entrada llamada *publickey*, la cual permite identificar que el ensamblado cuenta ahora con un nombre único y seguro.

```
.publickey = (00 24 00 00 04 80 00 00 94 00 00 00 06 02 00 00   // .$..............
              00 24 00 00 52 53 41 31 00 04 00 00 01 00 01 00   // .$..RSA1........
              9B AC 9F 18 A0 1E 31 C7 AB 55 CC 95 13 D7 6E E1   // ......1..U....n.
              A3 B9 9C B1 5C 1B 20 72 34 FF FA 47 7A 6B 2A 33   // ....\. r4..Gzk*3
              BF 97 9E 10 1B AF 48 22 7B 3B 90 A7 A9 D1 02 15   // ......H"{;......
              F3 1D EF 17 20 9B 50 16 C4 B2 34 04 B5 2C 8E 12   // .... .P...4..,..
              6C 36 74 BC 05 F5 99 FF 55 E2 78 F5 69 12 8D 82   // l6t.....U.x.i...
              25 04 72 A6 1F D2 77 0C AB 53 00 9D 43 A1 99 6A   // %.r...w..S..C..j
              92 E3 AA 9A C4 CE F8 37 55 C6 6A 99 8E 91 E3 0D   // .......7U.j.....
              39 78 39 12 B3 4B 03 D0 B2 18 CC 0F 5B D5 E6 C8 ) // 9x9..K......[...
```

Figura 18.9.

Por supuesto que el valor de la misma dependerá del conjunto de claves utilizadas, por lo que puede ser sustancialmente diferente al que usted pueda tener. El ensamblado puede estar ahora compartido, de forma similar a las bibliotecas que usted creaba en versiones previas del producto.

Esta aproximación ofrece varias ventajas, como que el sistema operativo hará uso de una única instancia para servir a todas las aplicaciones que la utilicen, así como también que existirá un único archivo físico. Emplearemos este ensamblado en las subsiguientes secciones, por lo que todavía no se deshaga de él.

Si bien no es una opción excluyente, la plataforma .NET ofrece una nueva característica denominada *Caché de ensamblados global*, la cual permite —entre otras— situar todos los ensamblados compartidos en un único lugar físico. Aprenderemos más de esto a continuación.

La caché de ensamblados global

Con el fin de centralizar todos los ensamblados compartidos, la arquitectura .NET cuenta con una característica llamada *Caché de ensamblados global o GAC (Global Assembly Cache)*. Básicamente la caché es una carpeta «especial» que puede contener ensamblados compartidos, y la cual facilita a las aplicaciones su posterior localización. Generalmente la misma se encuentra bajo el nombre *Assembly*, y está localizada por debajo del directorio del sistema operativo.

```
<unidad>:<sistema operativo>\Assembly
```

Por ejemplo, podría ser la siguiente:

```
c:\Windows\Assembly
```

Una vez que se le ha dado nombre único a un ensamblado, el mismo puede ser copiado a esta carpeta, con el fin de hacerlo visible a todas las aplicaciones de la plataforma .NET.

Copiar el ensamblado a este directorio ofrece características similares a las que brindaba el registro de un componente en versiones anteriores de Visual Basic, pero sin la imperiosa dependencia del archivo de registro. La utilización de la caché no es obligatoria, pero puede darle un buen punto de partida para la administración de los diferentes componentes. Los ensamblados compartidos son generalmente agregados a la misma por los instaladores, o por usted si sabe cómo hacerlo.

Como vimos anteriormente, un ensamblado compartido debe especificar de forma muy estricta la información de versión, cultura, compatibilidad y dependencias, la cual son utilizadas para definir su nombre seguro (*Strong Name*). Adicionalmente vimos que se conjugaba la tecnología de cifrado, la cual aseguraba al autor de un componente como único gestor de las subsiguientes versiones del mismo.

Aunque sería más sencillo, no es posible agregar un ensamblado a la caché simplemente copiando el mismo a dicha carpeta, debido a que el proceso exige que se realicen algunas validaciones de integridad adicionales en el momento en que éste sea incluido a la misma. La adición de un ensamblado a la caché de ensamblados global cuenta con algunos beneficios, los cuales vemos a continuación:

Chequeo de integridad

Cuando un ensamblado es agregado a la caché, un chequeo de integridad es realizado sobre todos los archivos contenidos por éste, a los efectos de asegurar que el mismo podrá ser ejecutado.

Seguridad de archivos

Solamente aquellos usuarios con permisos de administrador podrán elimina archivos de la caché; de acuerdo con esta característica, los instaladores deberán ser siempre ejecutados con permisos de administrador. Esto elimina la posibilidad de que un usuario final pueda borrar ensamblados de una aplicación.

Versiones

Pueden existir varias versiones de un mismo ensamblado en la caché.

Múltiples nombres

Pueden existir varios ensamblados con igual nombre físico (de archivo), sin que ello afecte a quienes los utilizan.

Cada vez que un ensamblado es agregado a la caché, se realiza un chequeo de integridad sobre el mismo, el cual verifica que éste no haya sido adulterado, y que adicionalmente se cuente con todos los módulos necesarios para que el mismo pueda ser ejecutado exitosamente. Por otra parte, sólo el administrador puede eliminar ensamblados de la caché, por lo que se descarta la posibilidad de que un usuario final borre por error una biblioteca, cosa bastante común bajo la arquitectura anterior. Adicionalmente, múltiples versiones de un mismo componente pueden existir al mismo tiempo, aunque estos denominen de igual forma. Este es un punto muy significativo, ya que elimina uno de los pro-

blemas más importantes que causaban «El infierno de las DLL», el cual radicaba en que sólo una versión podía existir por cada componente en un sistema operativo.

Ahora múltiples versiones de un mismo componente pueden estar instaladas, y de hecho, diferentes aplicaciones podrían hacer uso de distintas versiones del mismo ensamblado de forma simultánea. Esto permite que la instalación de una nueva versión de un ensamblado no afecte a las aplicaciones que utilizan otra versión.

Existe una excepción a las reglas, que se da cuando se requiere que una versión sea suplantada por otra de forma obligatoria, con el fin de solucionar alguna falla (*bug*). Hablaremos de esto más adelante.

Visualizando los ensamblados de la caché

Existen varias formas de visualizar los ensamblados contenidos por la caché, y la más sencilla es a través del explorador de Windows. Cuando se instala Visual Studio, éste agrega a la aplicación una extensión para poder explorar la caché, la cual es utilizada cuando se hace clic sobre la carpeta *Assembly*. Esto permite exhibir de forma natural toda la información de los mismos, como su nombre, versión, clave pública, etc., usando la misma interfaz a la cual está ya acostumbrado.

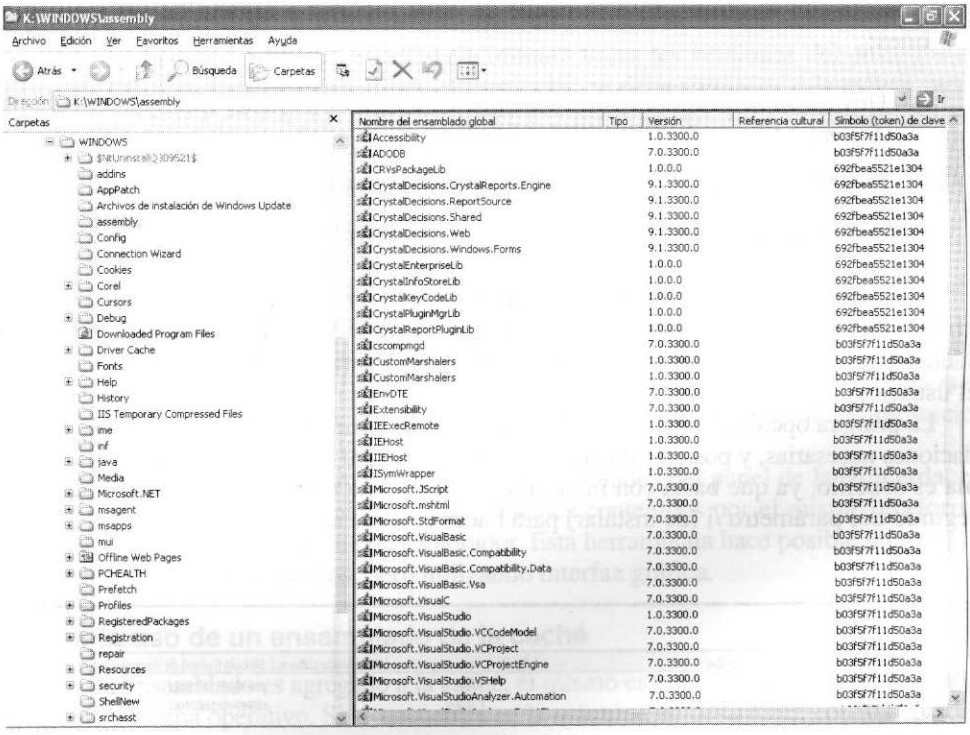

Figura 18.10.

La organización física interna que se emplea para guardar los ensamblados es diferente a la exhibida por el explorador, y ello es principalmente debido a que se implementan las estructuras necesarias para poder contener ensamblados con igual nombre físico (véase Fig. 18.11), cosa que no es naturalmente soportada por los sistemas operativos de Microsoft.

Sin embargo, no es necesario interactuar con la estructura interna, y solamente mencionamos esta característica con el fin de que conozca que la misma es diferente a la estructura lógica. En todos los casos emplearemos herramientas para agregar o quitar ensamblados de la caché.

Agregando ensamblados a la caché de ensamblados global

Como vimos anteriormente, es una tarea muy común la de querer que todos los ensamblados compartidos sean centralizados bajo un mismo sitio, a los efectos de que éstos puedan ser localizados de forma rápida, ya sea por una aplicación o por usted. Debido a ello, es una tarea muy común la de interactuar con los ensamblados de la caché, por ejemplo para agregar o quitar uno. Lamentablemente, no es posible agregar un ensamblado a esta estructura simplemente copiando el mismo en la carpeta *Assembly*, como se podría imaginar inicialmente. Ello es debido a que se realizan ciertas validaciones sobre el ensamblado en el momento de que éste sea adicionado. Para insertar un ensamblado a la caché se debe recurrir a alguna de las siguientes opciones:

a) Utilizar la herramienta .NET de línea de comando llamada *GacUtil.exe (Global Assembly Cache Utility).*

b) Emplear la aplicación Windows de configuración de la plataforma .NET.

c) Hacer uso de un instalador que coloque el ensamblado en el GAC.

Evidentemente, las primeras dos opciones son realizadas principalmente por desarrolladores, mientras que la última puede ser llevada a cabo por un usuario final. Más adelante veremos cómo crear paquetes de instalación para que puedan ser instalados por el usuario.

La primera opción plantea el uso de la herramienta *GacUtil.exe*, la cual realiza las validaciones necesarias, y posteriormente copia el ensamblado a la caché. El empleo de la misma es sencillo, ya que basta con invocarla incluyendo la ruta y nombre del ensamblado, seguido del parámetro /i (de instalar) para hacer efectiva su inclusión.

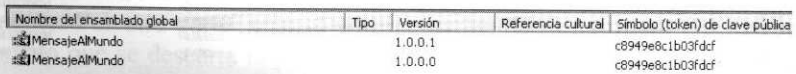

Figura 18.11.

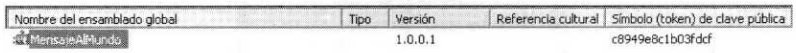

Figura 18.12.

Sintaxis:
gacutil <opción> <parámetros>

gacutil /i MensajeAlMundo.dll

Cuando esta línea sea ejecutada, el ensamblado será copiado a la caché, o se genera un error si el mismo no cuenta con un nombre seguro, o no se encuentra debajo de la ruta especificada. Para verificar que el mismo haya sido agregado, basta con examinar la caché mediante el explorador, como muestra la Figura 18.12.

El ensamblado está ahora disponible de forma global para todas las aplicaciones de la plataforma .NET que deseen acceder a él. Si se quiere eliminar a uno o varios elementos de la caché, basta con invocar la misma herramienta utilizando el parámetro /u.

gacutil /u MensajeAlMundo

Debe ser cauteloso, ya que esta línea eliminará todas las versiones del ensamblado *MensajeAlMundo*. Para especificar una en particular, se debe suministrar la mayor cantidad de información posible, tal como se exhibe a continuación:

gacutil /u MensajeAlMundo,Version=1.0.0.0,Culture=en,PublicKeyToken=134c32ab372e13ca

El resultado final también puede ser corroborado mediante el explorador, situando el mismo sobre la carpeta conteniendo la caché de ensamblados global.

Anteriormente comentamos que se contaba con una segunda opción, la cual podía resultar más sencilla para aquellos desarrolladores que no gustasen de las herramientas de línea de comando. La misma se ofrece a través de la aplicación de configuración de la plataforma .NET (mscorcfg.msc), que se encuentra dentro de la carpeta de *Herramientas administrativas* en el *Panel de control*, o bajo la misma carpeta en el menú de inicio, dependiendo de la versión de sistema operativo.

Haciendo clic sobre la opción de caché de ensamblados del panel de la izquierda, es posible acceder a la lista completa de componentes contenidos por el mismo, en forma similar a lo que se hacía mediante el explorador. Esta herramienta hace posible tanto agregar como eliminar ensamblados, pero utilizando interfaz gráfica.

Haciendo uso de un ensamblado de la caché

Cuando un ensamblado es agregado a la caché, el mismo es copiado a la carpeta *Assembly* debajo del sistema operativo. Si se encuentra en la máquina de desarrollo, contará con dos copias del mismo:

a) Aquella que fue generada en el momento de la compilación.

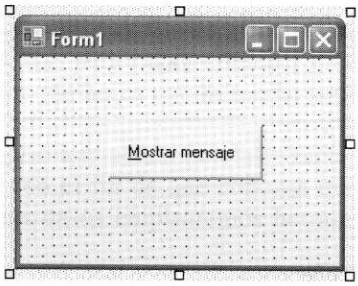

Figura 18.13.

b) Aquella que se encuentra dentro de la caché o GAC.

Esto es solamente válido en el ordenador de desarrollo, ya que cuando finalmente se instale la aplicación en el ordenador del usuario, sólo una de estas copias prevalecerá, la cual será colocada en la caché por el instalador.

Vamos entonces a ver cómo emplear la copia de la caché en el ordenador de desarrollo, y para ello crearemos un nuevo proyecto de aplicación para Windows llamado *PruebaMensajeAlMundoGAC*, el cual diseñaremos de forma similar a la Figura 18.13.

Objeto	Propiedad	Valor
Button	Name/Nombre	Mensaje
	Text	&Mostrar mensaje

Una vez hecho esto, agregaremos una referencia a la biblioteca *MensajeAlMundo.dll*. Para ello, iremos a la opción del menú *Proyecto*, y luego daremos clic en *Agregar Referencia*. Posteriormente haremos clic en *Explorar*, y a continuación nos desplazaremos hasta la carpeta */bin*, agregando el ensamblado *MensajeAlMundo.dll*, el cual agregamos a la caché en secciones anteriores. Por último, escribiremos el código necesario para realizar la llamada al método *Mensaje*, de forma similar a como se exhibe a continuación:

```
Private Sub Mensaje_Click(...
    Dim msj As New MensajeAlMundo.Mensajes.PlanetaTierra()

    msj.MostrarMensaje()

End Sub
```

El resultado obtenido es similar al implementado al comienzo de este capítulo, salvo que hicimos una referencia a un ensamblado que también existe en la caché.

Cada vez que el evento clic sea iniciado por el botón, el mensaje de saludo terrícola será exhibido por una caja estándar, empleando la biblioteca localizada en el directorio */bin*, pero no la residente en la caché.

Vamos entonces a compilar la aplicación, y luego a copiar solamente el archivo ejecutable (no el DLL) a una carpeta llamada *MensajeTerrícolas*, debajo del raíz. El resultado final debería verse similar a la Figura 18.14.

Si ejecutamos la aplicación nuevamente, veremos que la misma funciona correctamente, exhibiendo el mensaje obtenido del ensamblado. Pero...

¿Cómo es esto posible si solamente se ha copiado el EXE pero no el ensamblado DLL?

Bien, en realidad la misma está utilizando la versión guardada en la caché, y cuando ella es ejecutada, CLR detecta que el ensamblado no existe bajo la misma carpeta de la aplicación, por lo que intenta localizarlo en otro sitio, y esto incluye la caché de ensamblados global. Debido a que existe en este último, el mismo es automáticamente cargado y utilizado. Esto indica que una aplicación que requiera un ensamblado podrá ejecutarse normalmente si sus dependencias residen en la caché. Veremos más adelante que cuando se construye un instalador, es posible dar instrucciones precisas para que determinados componentes sean instalados dentro de la caché, en vez de que sea en la misma carpeta de la aplicación.

Como es posible apreciar, siempre que se trabaje en el entorno de desarrollo se deberá hacer referencia al ensamblado residente en la carpeta original, aunque, no obstante, CLR se encargue posteriormente de buscar el mismo en el GAC, en el caso de que éste no exista bajo el mismo directorio de la aplicación.

INCLUYENDO ENSAMBLADOS EN LA VENTANA DE REFERENCIAS

En versiones anteriores de Visual Basic, todos los componentes registrados en el sistema eran exhibidos en la ventana de referencias, a los efectos de que pudiesen ser agregados al proyecto. Si se registraba un nuevo componente, el mismo era automáticamente agregado a la lista. En esta versión, la sección de componentes COM de la ventana de referencias continúa funcionando de la misma forma, pero la lista de ensamblados .NET ofrece un comportamiento distinto.

A diferencia de lo que se podría esperar inicialmente, cuando un nuevo componte es agregado a la caché, el mismo no es mostrado por la lista de referencias disponibles para .NET. Ello es debido a que la inclusión de un ensamblado a la misma debe realizarse en forma explícita, y de hecho, no mantiene relación con los ensamblados que se encuentren compartidos. Si duda, esto es de mucha utilidad, ya que permite seleccionar qué ensamblados serán visualizados en la misma y cuáles no, en vez de que una lista interminable sea presentada. Para especificar que un conjunto de ensamblados deben ser mostrados por la misma, debe recurrirse a la siguiente entrada del archivo de registro:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework\AssemblyFolders
```

Figura 18.14.