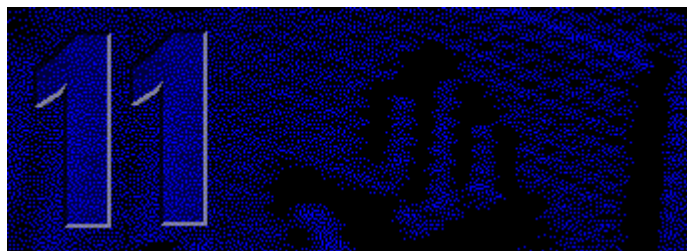


# Manual del programador, Parte 11: Lo nuevo en Visual FoxPro



Los capítulos siguientes describen las características nuevas de Visual FoxPro 6.0. Estas características aceleran y facilitan más que nunca la creación de aplicaciones de Visual FoxPro y le permiten crear aplicaciones para Internet e intranets.

## Capítulo 31 [Interoperabilidad e Internet](#)

Use arrastrar y colocar de OLE para programar aplicaciones que le permitan mover datos entre aplicaciones para Windows y dentro de una aplicación de Visual FoxPro. Cree aplicaciones y servidores de Visual FoxPro para su uso con Internet.

## Capítulo 32 [Programación de aplicaciones y productividad del programador](#)

La Galería de componentes y las Foundation Classes, la aplicación Analizador de trayecto, Enganches del Administrador de proyectos y Asistentes

## Capítulo 33 [Mejoras para la programación](#)

Nuevas características de programación diseñadas para mejorar la productividad del programador, entre las que se cuentan los métodos Access y Assign, la compatibilidad con más formatos gráficos y nuevas incorporaciones al lenguaje para simplificar las tareas de programación. Además, se han agregado a Visual FoxPro muchas de las funciones de manipulación de nombres de archivos disponibles en Foxttools.fll, una biblioteca API de Visual FoxPro.

# Capítulo 31: Interoperabilidad e Internet

Microsoft Visual FoxPro 6.0 admite la técnica arrastrar y colocar OLE, que permite mover datos entre Visual FoxPro y otras aplicaciones, y también dentro de las propias aplicaciones de Visual FoxPro.

Además, Visual FoxPro 6.0 facilita la creación de aplicaciones para su uso con Internet y otros programas para Windows, como Microsoft Excel y Microsoft Visual Basic. Visual FoxPro 6.0 permite crear [documentos activos](#) que pueden alojarse en contenedores de documentos activos, como exploradores de Internet.

Visual FoxPro 6.0 proporciona servidores de Automatización mejorados para trabajar con Internet, Microsoft Transaction Server y Active Desktop.

En este capítulo se tratan los temas siguientes:

- [Técnica arrastrar y colocar de OLE](#)
- [Documentos activos](#)
- [Mejoras de servidor](#)

## Técnica arrastrar y colocar de OLE

Visual FoxPro ya es compatible con la técnica arrastrar y colocar de OLE, una eficaz y útil herramienta que permite mover datos entre las aplicaciones que admiten dicha técnica (como Visual FoxPro, Visual Basic, el Explorador de Windows, Microsoft Word y Excel, etc.). En una aplicación de Visual FoxPro distribuida, puede mover datos de unos controles a otros o entre controles y otras aplicaciones para Windows compatibles con la técnica arrastrar y colocar de OLE.

Tenga en cuenta que las versiones anteriores de Visual FoxPro admitían la funcionalidad de arrastrar y colocar para los controles mediante programación, lo que permitía mover los controles en un formulario. Esta forma de arrastrar y colocar se mantiene en Visual FoxPro 6.0. Sin embargo, si elige implementar la técnica de arrastrar y colocar en sus aplicaciones, debe usar sólo una de las dos técnicas, arrastrar y colocar controles mediante programación o la técnica arrastrar y colocar de OLE, sin mezclar ambos tipos.

El conocimiento de los fundamentos de la técnica arrastrar y colocar de OLE facilita el aprovechamiento total de sus posibilidades.

### Arrastrar y colocar datos

Para arrastrar y colocar datos entre aplicaciones y controles se utiliza el *mouse* (ratón). Por ejemplo, puede seleccionar un conjunto de archivos en el Explorador de Windows. A continuación, puede mantener presionado el botón del *mouse* mientras los arrastra y después liberarlo para colocar los archivos en el Administrador de proyectos de Visual FoxPro; o bien, puede seleccionar texto en un documento de Word y colocarlo en un cuadro de texto de un formulario de Visual FoxPro. Durante la operación arrastrar y colocar de OLE, el cursor del *mouse* cambia de forma para indicar que la operación está en curso.

### Origen de arrastre

La aplicación o control desde el que se mueven los datos se denomina *origen de arrastre*.

### Propiedades, eventos y métodos del origen de arrastre

En la siguiente tabla se indican las propiedades, eventos y métodos disponibles para un origen de arrastre OLE.

| <b>Propiedad, evento o método</b>        | <b>Descripción</b>   |
|--|--|
| <a href="#">Evento OLECompleteDrag</a>   | Ocurre cuando se colocan los datos en el destino de colocación o cuando se cancela la operación OLE de arrastrar y colocar.  |
| <a href="#">Método OLEDrag</a>           | Inicia una operación OLE de arrastrar y colocar.   |
| <a href="#">Propiedad OLEDragPicture</a> | Especifica la imagen que aparece debajo del puntero del <i>mouse</i> durante una operación arrastrar y colocar de OLE. Puede especificar un archivo de imagen de tipo .bmp, .dib, .jpg, .gif, .ani, .cur o .ico. |
| <a href="#">Propiedad OLEDragMode</a>    | Especifica la forma en que un origen de arrastre administra las operaciones de arrastre OLE.   |
| <a href="#">Evento OLEGiveFeedBack</a>   | Ocurre después de cada evento OLEDragOver. Permite al origen de arrastre especificar el tipo de operación arrastrar y colocar de OLE, así como el resultado visual.  |
| <a href="#">Evento OLESetData</a>        | Ocurre cuando un destino para colocar llama al método GetData y no hay datos con un formato especificado en el objeto DataObject al que se refiere la operación OLE de arrastrar y colocar.                      |
| <a href="#">Evento OLEStartDrag</a>      | Ocurre cuando se llama al método OLEDrag.  |

## Destino para colocar

La aplicación o control al que se mueven los datos se denomina *destino para colocar*.

### Propiedades y eventos del destino para colocar

En la tabla siguiente se indican las propiedades, eventos y métodos disponibles para un destino para colocar OLE.

| <b>Propiedad o evento</b>                | <b>Descripción</b>   |
|--|--|
| <a href="#">Evento OLEDragDrop</a>       | Ocurre cuando se colocan datos en un destino para colocar y la propiedad OLEDropMode de éste tiene el valor 1 – Activado.  |
| <a href="#">Evento OLEDragOver</a>       | Ocurre cuando se arrastran datos a un destino para colocar y la propiedad OLEDropMode de éste tiene el valor 1 – Activado. |
| <a href="#">Propiedad OLEDropEffects</a> | Especifica el tipo de operaciones de colocación que admite un destino para colocar OLE.                                    |
| <a href="#">Propiedad OLEDropHasData</a> | Especifica la forma de administrar una operación de  |

colocación.

---

---

[Propiedad OLEDropMode](#)

Especifica la forma en que un destino para colocar administra las operaciones colocar de OLE.

---

---

## Mover datos

Para realizar una operación de arrastrar y colocar con el fin de mover datos con el botón predeterminado del *mouse* (principal), seleccione los datos que desee mover en el origen de arrastre. Una vez seleccionados los datos, mantenga presionado el botón del *mouse* mientras desplaza el puntero hasta el destino de colocación. Suelte el botón del *mouse* para colocar los datos en el destino. Durante la operación arrastrar y colocar de OLE, el cursor del *mouse* cambia de forma para indicar que la operación está en curso.

También puede hacer clic con el botón no predeterminado del *mouse* (secundario) en los datos del origen de arrastre y arrastrarlos hasta un destino de colocación. En función del destino, al colocar los datos puede aparecer un menú contextual con un conjunto de opciones que permiten elegir la forma en que se van a procesar los datos en el destino de colocación.

## Copiar datos

También puede copiar datos desde un origen de arrastre y pegarlos en un destino para colocar. Presione la tecla Ctrl mientras hace clic con el *mouse* en los datos seleccionados en el origen de arrastre. En el cursor del *mouse* aparecerá un signo más (+) mientras se arrastran los datos, para indicar que se está realizando una copia.

## Destinos y orígenes que no admiten la técnica arrastrar y colocar de OLE

Sólo se puede mover o copiar datos desde un origen de arrastre compatible con la técnica arrastrar y colocar de OLE hasta un destino de colocación que también admita dicha característica. Tenga en cuenta que aunque un destino de colocación admita la técnica arrastrar y colocar de OLE, ello no implica que acepte los datos que desee colocar en él. Por ejemplo, es posible que los datos que desea mover o copiar tengan un formato que no es compatible con el destino de colocación. En las operaciones de arrastrar y colocar, el cursor se transforma en el símbolo No colocar (un círculo tachado) para indicar que el *mouse* se encuentra en una área de una aplicación o de un control en el que no se pueden colocar los datos.

## Cancelar una operación

Para cancelar una operación arrastrar y colocar de OLE, presione ESC mientras la efectúa.

## Técnica arrastrar y colocar de OLE en tiempo de diseño

La técnica arrastrar y colocar de OLE en tiempo de diseño de Visual FoxPro hace que la programación de aplicaciones sea aún más rápida que en las versiones anteriores. Esta técnica permite colocar de forma sencilla en el Administrador de proyectos y en los diseñadores de Visual FoxPro archivos del Explorador de Windows. También se permite mover o copiar fácilmente texto desde otras aplicaciones a la ventana Comandos, a los editores de texto de Visual FoxPro y a la ventana

Propiedades.

En la tabla siguiente se indican las características de tiempo de diseño de Visual FoxPro que admiten la técnica arrastrar y colocar de OLE, junto con una descripción de su uso.

| Elemento de la interfaz    | Descripción   |
|----------------------------|---|
| Ventana Comandos           | <p data-bbox="602 405 1442 474">Destino para colocar archivos y origen de arrastre y destino para colocar texto.</p> <p data-bbox="602 516 1442 873">Si se coloca un archivo creado con Visual FoxPro en la ventana Comandos, se abrirá el archivo con el comando de Visual FoxPro correspondiente. Por ejemplo, si se coloca en la ventana Comandos una base de datos, Visual FoxPro ejecutará los comandos OPEN DATABASE y MODIFY DATABASE para abrir el archivo y poder modificarlo. Si se coloca en la ventana Comandos una tabla, ésta se abrirá con los comandos USE ... AGAIN y BROWSE. Si SET EXCLUSIVE tiene el valor ON, la tabla se abrirá para uso exclusivo. Si SET EXCLUSIVE tiene el valor OFF, se abrirá para uso compartido.</p> <p data-bbox="602 915 1442 1056">Otros archivos de Visual FoxPro se abren con el comando MODIFY correspondiente: los formularios se abren con MODIFY FORM, las consultas con MODIFY QUERY, los archivos de texto y de encabezado (.H) con MODIFY FILE, etc.</p> <p data-bbox="602 1098 1442 1272">Si se coloca en la ventana Comandos un archivo creado con otra aplicación, se abrirá en la aplicación a la que esté asociado. Por ejemplo, al colocar una hoja de cálculo de Microsoft Excel en la ventana Comandos se iniciará Excel y abrirá dicha hoja de cálculo.</p> |
| Administrador de proyectos | <p data-bbox="602 1304 992 1335">Destino para colocar archivos.</p> <p data-bbox="602 1377 1442 1560">Los archivos se agregan a las correspondientes categorías del Administrador de proyectos, en función de sus extensiones de archivo. Si Visual FoxPro no reconoce la extensión de un archivo que se coloca en el Administrador de proyectos, el archivo se agregará a la categoría Otros.</p> <p data-bbox="602 1602 1442 1881">Si se coloca en el Administrador de proyectos una tabla contenida en una base de datos, ésta se agregará a la categoría Bases de datos del elemento Datos y se marcará como Excluida. Si se coloca en el Administrador de proyectos una tabla libre, ésta se agregará a la categoría Tablas libres del elemento Datos y se marcará como Excluida. Si se coloca en el Administrador de proyectos una base de datos, ésta se agregará a la categoría Bases de datos del elemento Datos y se marcará como Excluida.</p>   |

Aunque con la técnica arrastrar y colocar de OLE es fácil agregar archivos al Administrador de proyectos, recuerde que el Administrador de proyectos agrega automáticamente al proyecto todos los archivos a los que se hace referencia cuando se genera. Por ejemplo, si un programa agregado al proyecto ejecuta otro programa, éste se agregará automáticamente al generar el proyecto. No es necesario agregar el segundo programa manualmente.

---



---

Editores  
de texto

Origen de arrastre y destino de colocación para texto.

Se consideran editores de texto las ventanas de edición abiertas con MODIFY COMMAND, MODIFY FILE y MODIFY MEMO; la ventana Comandos; las ventanas de edición de miniprogramas de los Diseñadores de formularios, de clases, de menús y de entornos de datos; y el editor de procedimientos almacenados del Diseñador de bases de datos.

---



---

Depurador

Origen de arrastre y destino para colocar texto.

El cuadro de texto de la ventana Inspección y la lista Nombre son orígenes de arrastre y destinos de colocación de texto. Las ventanas Seguimiento y Resultados del depurador son sólo orígenes de arrastre para texto.

---



---

Diseñador de bases de datos

Destino para colocar archivos.

Al colocar una tabla en el Diseñador de bases de datos, la tabla se agrega a la base de datos actual.

---



---

Diseñador de clases

Destino para colocar texto y archivos.

De forma predeterminada, al colocar texto en un objeto contenedor del Diseñador de clases se crea una etiqueta con ese texto como valor de la propiedad Caption. Puede cambiar el control predeterminado creado al colocar texto en el Diseñador de formularios. Para ello, utilice la ficha Asignación de campos del cuadro de diálogo Opciones.

Si coloca texto en un control no contenedor (CheckBox, CommandButton, Header, Label u OptionButton), el texto se asignará a la propiedad Caption del control.

Al colocar un archivo gráfico (.ani, .bmp, .cur, .gif, .ico o .jpg) en el Diseñador de clases se crea un control Image cuya propiedad Picture tiene como valor el nombre de ese archivo gráfico.

---



---

Diseñador de entornos de datos

Destino para colocar archivos.

Al colocar una tabla en el Diseñador de entornos de datos, se agrega la tabla al entorno de datos. Al colocar una base de datos en el Diseñador de entornos de datos aparece el cuadro de diálogo Agregar tabla o vista, que permite agregar una tabla o una vista al entorno de datos.

---



---

Diseñador de consultas

Destino para colocar archivos.

Al colocar una tabla en el Diseñador de consultas, se agrega la tabla a la consulta. Al colocar una base de datos en el Diseñador de consultas aparece el cuadro de diálogo Agregar tabla o vista, que permite agregar una tabla o una vista a la consulta.

---



---

Diseñador de vistas

Destino para colocar archivos.

Al colocar una tabla en el Diseñador de vistas, se agrega la tabla a la vista. Al colocar una base de datos en el Diseñador de vistas aparece el cuadro de diálogo Agregar tabla o vista, que permite agregar una tabla o una vista a la vista.

---



---

Ventana Propiedades

Destino para colocar texto.

Puede arrastrar texto al cuadro de texto que aparece en la parte superior de la ventana Propiedades al seleccionar una propiedad en tiempo de diseño.

---



---

Galería de componentes

Origen de arrastre y destino para colocar archivos.

Puede arrastrar objetos desde la Galería de componentes y colocarlos en el Diseñador de formularios. También puede arrastrar archivos desde la Galería de componentes y colocarlos en el Administrador de proyectos.

Además, puede colocar archivos en la Galería de componentes.

---



---

## Técnica arrastrar y colocar de OLE en tiempo de ejecución

La técnica arrastrar y colocar de OLE está disponible en tiempo de ejecución para los controles de Visual FoxPro y para el editor de texto. En tiempo de ejecución, los controles y el editor de texto admiten esta técnica de forma interactiva; además, los controles admiten la técnica mediante programación. El [objeto DataObject](#) permite el uso de la técnica arrastrar y colocar de OLE mediante programación para los controles.

Existen dos técnicas arrastrar y colocar de OLE disponibles para los controles de Visual FoxPro: el modo intrínseco y el modo manual. En el primero, Visual FoxPro controla intrínsecamente la operación arrastrar y colocar de OLE. En el modo manual, las operaciones arrastrar y colocar de OLE se controlan mediante programación. Los eventos que se producen están determinados por el modo utilizado. Si desea obtener más información, vea la sección "Modos intrínseco y manual Arrastrar y

colocar de OLE".

## Arrastrar y colocar en versiones anteriores de Visual FoxPro

En las versiones anteriores de Visual FoxPro se utilizaba la técnica de arrastrar y colocar mediante programación para los controles, lo que permitía mover los controles de un formulario. Este modo de arrastrar y colocar sigue siendo compatible. Si utiliza los valores predeterminados de las propiedades OLEDragMode y OLEDropMode, podrá ejecutar las aplicaciones existentes como antes, sin ningún cambio.

## El objeto DataObject

El [objeto DataObject](#) es un contenedor para los datos que se transfieren desde un origen de arrastre OLE hasta un destino para colocar OLE y sólo existe mientras se realiza la operación arrastrar y colocar de OLE. El objeto DataObject no se puede crear mediante programación y las referencias al mismo dejan de ser válidas al finalizar la operación de arrastrar y colocar. El objeto DataObject se pasa como parámetro oDataObject en los eventos OLEStartDrag, OLEDragOver, OLEDragDrop y OLESetData.

El objeto DataObject puede almacenar varios conjuntos de datos, cada uno con un formato distinto. Puede usar el método GetFormat para determinar si existe un formato específico en el objeto DataObject. Consulte [Método GetFormat](#) para ver una lista con los formatos que admite el objeto DataObject.

## Métodos del objeto DataObject

El objeto DataObject tiene métodos que permiten manipular mediante programación los datos que se arrastran y colocan. En la tabla siguiente se indican los métodos disponibles en tiempo de ejecución para el objeto DataObject.

| Método                    | Descripción  |
|---------------------------|--|
| <a href="#">ClearData</a> | Borra todos los datos y formatos del objeto DataObject de la operación arrastrar y colocar de OLE.                                 |
| <a href="#">GetData</a>   | Recupera los datos del objeto DataObject de la operación arrastrar y colocar de OLE.   |
| <a href="#">GetFormat</a> | Determina si hay datos con el formato especificado disponibles en el objeto DataObject de la operación arrastrar y colocar de OLE. |
| <a href="#">SetData</a>   | Establece los datos y el formato de los mismos en el objeto DataObject de la operación arrastrar y colocar de OLE.                 |
| <a href="#">SetFormat</a> | Establece un formato de datos, sin los datos, en el objeto DataObject de la operación arrastrar y colocar de OLE.                  |

## Modos intrínseco y manual arrastrar y colocar de OLE

Visual FoxPro admite dos modos de arrastrar y colocar de OLE para los controles: el modo intrínseco

y el modo manual. En el primero, las operaciones arrastrar y colocar de OLE las controla Visual FoxPro. En el modo manual, las operaciones se controlan mediante programación.

### **Modo intrínseco de arrastrar y colocar de OLE**

El modo intrínseco arrastrar y colocar de OLE se puede implementar en una aplicación para proporcionar compatibilidad con la técnica arrastrar y colocar de OLE sin necesidad de programación adicional.

#### **Para implementar la técnica intrínseca de arrastrar y colocar de OLE para un control**

1. Asigne el valor **1** – Automático a la propiedad `OLEDragMode` del control para permitir que éste actúe como origen de arrastre OLE.
2. Asigne el valor **1** – Activado a la propiedad `OLEDropMode` del control para permitir que éste actúe como destino para colocar OLE.

En las operaciones intrínsecas de arrastrar y colocar de OLE, Visual FoxPro determina si el destino para colocar admite el formato de los datos que se intentan colocar en él. Sólo se colocarán los datos si el destino es compatible.

En la tabla siguiente se indican los controles de Visual FoxPro con los formatos de datos que admiten como orígenes de arrastre en el modo intrínseco. Tenga en cuenta que `CF_TEXT` es texto, como el que escribe en un cuadro de texto, y `CFSTR_VFPSOURCEOBJECT` es una referencia a tipo de objeto para un control u objeto de Visual FoxPro. En el caso de los controles que admiten el formato de datos `CF_TEXT`, puede arrastrar texto desde la parte de texto del control.

#### **Formatos de datos de los orígenes de arrastre**

| <b>Control</b>  | <b>Formato de datos (definido en Foxpro.h)</b>  |
|---|---|
| Container, Image, Line, PageFrame y Shape               | <code>CFSTR_VFPSOURCEOBJECT</code>  |
| CommandButton y Label                                   | <code>CFSTR_VFPSOURCEOBJECT</code> y <code>CF_TEXT</code>                                 |
| CheckBox, ComboBox, EditBox, ListBox, Spinner y TextBox | <code>CFSTR_VFPSOURCEOBJECT</code> , <code>CF_TEXT</code> y <code>CFSTR_OLEVARIANT</code> |

En la tabla siguiente se indican los controles de Visual FoxPro con los formatos de datos que admiten como destinos para colocar en el modo intrínseco. En el caso de los controles, puede colocar texto en la parte de texto del control. El texto se sitúa en el punto de inserción.

#### **Formatos de datos de los destinos para colocar**

| <b>Control</b>  | <b>Formato de datos</b> |
|---|-------------------------|
| EditBox y ComboBox (cuando la propiedad Style del ComboBox es 0 - Cuadro desplegable) | CF_TEXT                 |
| Spinner y TextBox   | CFSTR_OLEVARIANT        |

### **Modo manual de arrastrar y colocar de OLE**

Es posible que en algunos casos desee controlar el tipo de datos que se pueden colocar en un destino para colocar o proporcionar una funcionalidad adicional a una operación de arrastrar y colocar. Por ejemplo, puede convertir los datos a un formato que el destino para colocar admita o puede mostrar un cuadro de diálogo que pregunte al usuario si desea colocar los datos en el destino. En estos casos, puede prescindir el modo intrínseco de arrastrar y colocar de OLE para lograr un mayor control sobre las operaciones de arrastrar y colocar.

Para implementar el modo manual de arrastrar y colocar de OLE para un control, anule los eventos o métodos de arrastrar y colocar que desee controlar; para ello, escriba su propio código para cada evento o método. Incluya la palabra clave NODEFAULT en el código del evento o método para pasar por alto el comportamiento intrínseco de Visual FoxPro al arrastrar y colocar.

La compatibilidad con aplicaciones existentes de versiones anteriores (sin arrastre OLE) se consigue cuando el valor de OLEDragMode es 0 (valor predeterminado) y no se incluye código adicional para operaciones arrastrar y colocar de OLE.

## **Documentos activos**

Visual FoxPro 6.0 permite crear documentos activos. Los documentos activos permiten ver documentos con formato distinto de HTML en un contenedor explorador de Web, como Microsoft Internet Explorer. La tecnología de documentos activos permite ver varios tipos de documentos de diversos orígenes en un mismo contenedor de documentos activos.

Un documento activo es un tipo específico de documento OLE que se puede incrustar. Se muestra en todo el área cliente de un contenedor de documentos activos y sus menús se combinan con los del contenedor. El documento activo ocupa todo el marco y está siempre activo en contexto.

Las siguientes son algunas de las características de los documentos activos:

- Los documentos activos siempre están activos en el contexto.
- Los comandos de menús y de barras de herramientas del documento activo se pueden dirigir al contenedor de documentos activos.
- Los documentos activos proporcionan integración directa con otras páginas Web cuando se ven en Internet Explorer.
- Los documentos activos suponen un paso adelante en la evolución desde las aplicaciones

clientes puras de Visual FoxPro a las aplicaciones Active Platform que utilizan una interfaz de cliente basada en HTML.

## Crear un documento activo

Los documentos activos de Visual FoxPro se crean fácilmente. Un documento activo de Visual FoxPro, como cualquier otra aplicación de Visual FoxPro, puede manipular datos, ejecutar formularios, informes y etiquetas, crear instancias de clases y ejecutar código.

Un documento activo de Visual FoxPro es una aplicación (.app) creada desde un proyecto de Visual FoxPro. Las versiones anteriores de Visual FoxPro ya permitían crear aplicaciones, por lo que es posible que esté familiarizado con el proceso. Si desea obtener más información sobre la creación de aplicaciones, consulte el capítulo 13, [Compilar una aplicación](#), en el *Manual del programador*.

Puede ejecutar cualquier aplicación en Internet Explorer. Sin embargo, sólo las aplicaciones basadas en la clase de base ActiveDoc, descrita más adelante, admiten las propiedades, eventos y métodos que permiten la comunicación con el contenedor de documentos activos.

## La clase de base ActiveDoc

Los documentos activos de Visual FoxPro son ligeramente diferentes de otras aplicaciones (.app). La diferencia más destacada es que el “archivo principal” de un documento activo debe ser una clase basada en la clase base ActiveDoc. Otros tipos de aplicaciones requieren que el archivo principal sea un programa o un formulario.

Una clase basada en la clase de base ActiveDoc se crea con el Diseñador de clases y sirve como base para todos los documentos activos de Visual FoxPro. La clase de base ActiveDoc proporciona las propiedades, eventos y métodos para un documento activo, así como la comunicación con el contenedor de documentos activos. Por ejemplo, el evento ContainerRelease se produce cuando un contenedor libera un documento activo. Puede incluir código en este evento para cerrar archivos, completar transacciones y realizar otras tareas de limpieza antes de liberar el documento activo.

## Para establecer como archivo principal una clase basada en la clase de base ActiveDoc

1. Agregue al proyecto la biblioteca de clases visuales (.vcx) que contiene la clase basada en la clase de base ActiveDoc.
2. Expanda la jerarquía de la biblioteca de clases visuales (.vcx); para ello, haga clic en el cuadro más (+) situado a la izquierda del nombre de la biblioteca o haga clic con el botón secundario del *mouse* en la biblioteca y elija **Expandir todo** en el menú contextual.
3. Seleccione la clase basada en la clase de base ActiveDoc. Haga clic con el botón secundario del *mouse* en la clase y elija **Establecer principal** en el menú contextual.

## El objeto ActiveDoc

Cuando se ejecuta un documento activo de Visual FoxPro Active en Internet Explorer, se crea un objeto ActiveDoc a partir de la clase de base ActiveDoc. Este objeto responde a los eventos y llamadas a métodos de la clase de base ActiveDoc.

## Propiedades, eventos y métodos del objeto ActiveDoc

En las tablas siguientes se indican las propiedades, eventos y métodos que admite el objeto ActiveDoc.

### Propiedades

|                              |                         |                                      |
|------------------------------|-------------------------|--------------------------------------|
| <a href="#">BaseClass</a>    | <a href="#">Caption</a> | <a href="#">Class</a>                |
| <a href="#">ClassLibrary</a> | <a href="#">Comment</a> | <a href="#">ContainerReleaseType</a> |
| <a href="#">Name</a>         | <a href="#">Parent</a>  | <a href="#">ParentClass</a>          |
| <a href="#">Tag</a>          |                         |                                      |

### Eventos

|                                   |                                    |                                  |
|-----------------------------------|------------------------------------|----------------------------------|
| <a href="#">CommandTargetExec</a> | <a href="#">CommandTargetQuery</a> | <a href="#">ContainerRelease</a> |
| <a href="#">Destroy</a>           | <a href="#">Error</a>              | <a href="#">HideDoc</a>          |
| <a href="#">Init</a>              | <a href="#">Run</a>                | <a href="#">ShowDoc</a>          |

### Métodos

|                                |                                |                                 |
|--------------------------------|--------------------------------|---------------------------------|
| <a href="#">AddProperty</a>    | <a href="#">ReadExpression</a> | <a href="#">ReadMethod</a>      |
| <a href="#">ResetToDefault</a> | <a href="#">SaveAsClass</a>    | <a href="#">WriteExpression</a> |

### Secuencia de eventos en los documentos activos

Cuando se abre una aplicación de tipo documento activo en Internet Explorer, se ejecuta el documento activo y se produce su evento Init. A continuación, se produce el evento ShowDoc del documento activo. Cuando Internet Explorer aloja correctamente el documento activo, se produce el evento Run del mismo. En general, el código del programa de documento activo debe situarse en este evento. Normalmente, el evento Run contiene código que ejecuta los menús, ejecuta el formulario principal de la aplicación y contiene READ EVENTS para iniciar el procesamiento de los eventos, como en una aplicación Visual FoxPro estándar.

Puede incluir código de configuración en el evento Init del documento activo, pero si tarda demasiado tiempo en ejecutarse, es posible que el contenedor del documento activo genere un error de tiempo de espera. Si incluye código de configuración en el evento Init, no debe requerir interacción del usuario ni crear una interfaz de usuario.

El evento HideDoc ocurre al desplazarse desde un documento activo y el evento ShowDoc se produce al volver al mismo.

Si se cierra Internet Explorer cuando aloja el documento activo, se producirá el evento HideDoc y, a continuación, el evento ContainerRelease. Este último evento también se produce si el documento activo queda fuera de la memoria caché de Internet Explorer 3.0.

Cuando se produce el evento ContainerRelease, el código de programa del evento puede realizar las siguientes acciones:

- Cerrar archivos, limpiar su rastro y ejecutar QUIT para cerrar el documento activo.
- Asignar el valor 0 (predeterminado) a la propiedad [ContainerReleaseType](#), con lo que se abre el documento activo en el entorno de tiempo de ejecución de Visual FoxPro. El documento activo continuará ejecutándose en la ventana principal de tiempo de ejecución de Visual FoxPro.

**Nota** El evento CommandTargetExec se produce cuando Internet Explorer 4.0 va a cerrar el documento activo o a explorar desde el mismo. En este caso, se asigna el valor 37 al parámetro *nIdComando* de CommandTargetExec y puede asignar el valor falso (.F.) al parámetro *eSalArg* para evitar que Internet Explorer cierre el documento activo. Internet Explorer 3.0 no admite el evento CommandTargetExec.

### Nuevas funciones de documentos activos

Se han agregado a Visual FoxPro dos nuevas funciones, [GETHOST\(\)](#) e [ISHOSTED\(\)](#), para proporcionar información sobre el contenedor de un documento activo. GETHOST() devuelve la referencia de un objeto al contenedor de un documento activo. ISHOSTED() devuelve un valor lógico que indica si un documento activo se encuentra en un contenedor.

### Cambios en el objeto Form

La interfaz de usuario de un documento activo de Visual FoxPro está definida por su código de programa. En general, un formulario de Visual FoxPro debe mostrarse como interfaz de usuario inicial. Las siguientes propiedades, eventos y métodos de formulario se han agregado a Visual FoxPro para que los formularios funcionen correctamente con documentos activos.

#### Propiedades

---

---

[AlwaysOnBottom](#)

[ContinuousScroll](#)

[HscrollSmallChange](#)

---

---

[Scrollbars](#)

[TitleBar](#)

[ViewPortHeight](#)

---

---

[ViewPortLeft](#)

[ViewPortTop](#)

[ViewPortWidth](#)

---

---

[VscrollSmallChange](#)

---

---

#### Eventos

---

---

[Scrolled](#)

---

---

## Métodos

---

### [SetViewPort](#)

---

## Formularios de documentos activos

Los formularios de un documento activo se muestran en el área cliente que ofrece Internet Explorer. Para hacer que un formulario se muestre completamente en el área cliente de Internet Explorer, asigne los siguientes valores a las propiedades de formulario que se indican a continuación:

BorderStyle = 0 (Sin borde)  
 TitleBar = 0 (Desactivado)  
 WindowState = 2 (Maximizado)

Además, si se van a mostrar barras de desplazamiento cuando el área de cliente de Internet Explorer sea más pequeña que el área de visualización del documento activo (el área determinada por un rectángulo que incluye todos los controles del formulario), debe asignar el siguiente valor a la propiedad Scrollbars:

ScrollBars = 3 (Barras de desplazamiento horizontal y vertical)

## Menús en documentos activos

Si se ejecuta código de menús en un documento activo de Visual FoxPro, los menús se combinan con los menús de Internet Explorer, conforme a unas reglas de combinación de menús específicas. Cuando se hayan combinado los menús del documento activo con los de Internet Explorer, los primeros se verán como en una aplicación normal de Visual FoxPro.

## Negociar menús

En Visual FoxPro 6.0 y en las versiones anteriores, puede especificar el comportamiento de la negociación de menús cuando un control ActiveX contenido en un formulario de Visual FoxPro se modifica de forma visual mediante OLE. En Visual FoxPro 6.0, se ha mejorado la negociación de menús para permitir controlar el lugar de Internet Explorer en el que deben aparecer los menús del documento activo.

Cuando se abre un documento activo en Internet Explorer, el espacio para menús de Internet Explorer se comparte y los menús se combinan. Los menús combinados se dividen en seis grupos y cada uno de ellos pertenece a Internet Explorer, al documento activo, o a ambos.

| Grupo            | Propietario       |
|------------------|-------------------|
| Grupo Archivo    | Internet Explorer |
| Grupo Edición    | Documento activo  |
| Grupo Contenedor | Internet Explorer |
| Grupo Objeto     | Documento activo  |

|               |                                      |
|---------------|--------------------------------------|
| Grupo Ventana | Internet Explorer                    |
| Grupo Ayuda   | Documento activo o Internet Explorer |

### Combinar el menú Ayuda

El documento activo comparte su menú Ayuda con Internet Explorer. Si Internet Explorer tiene un menú Ayuda, el documento activo puede agregar el suyo al final del menú Ayuda de Internet Explorer.

### Mejoras del lenguaje para la negociación de menús

Se ha mejorado la cláusula DEFINE PAD NEGOTIATE para permitir especificar la forma en que se produce la negociación de los menús en un documento activo. Una nueva segunda opción, *cPosiciónObjeto*, especifica la ubicación del título de un menú en la barra de menús de Internet Explorer.

Para obtener más información, vea [DEFINE PAD](#) en la *Referencia del lenguaje*.

### Negociación de menús y el Diseñador de menús

Se ha mejorado el cuadro de diálogo [Opciones de la acción](#) del Diseñador de menús para permitir especificar la negociación de los menús creados en el Diseñador de menús e incluidos en documentos activos. Se ha agregado un cuadro desplegable **Objeto**, que especifica la forma de negociar el título del menú cuando Internet Explorer actúa como contenedor de un documento activo de Visual FoxPro.

### Información de negociación de menús

La información sobre la negociación de menús se almacena en el campo Location del archivo .mnx de cada menú. En la tabla siguiente se indican los valores de este campo y el tipo de negociación que representa cada uno. Para obtener más información sobre *cPosiciónContenedor* y *cPosiciónObjeto*, vea [DEFINE PAD](#).

| Valor | <i>cPosiciónContenedor</i> | <i>cPosiciónObjeto</i> |
|-------|----------------------------|------------------------|
| 0     | Ninguna                    | Ninguna                |
| 1     | Izquierda                  | Ninguna                |
| 2     | Centro                     | Ninguna                |
| 3     | Derecha                    | Ninguna                |
| 4     | Ninguna                    | Izquierda              |
| 5     | Izquierda                  | Izquierda              |
| 6     | Centro                     | Izquierda              |
| 7     | Derecha                    | Izquierda              |

|    |           |         |
|----|-----------|---------|
| 8  | Ninguna   | Centro  |
| 9  | Izquierda | Centro  |
| 10 | Centro    | Centro  |
| 11 | Derecha   | Centro  |
| 12 | Ninguna   | Derecha |
| 13 | Izquierda | Derecha |
| 14 | Centro    | Derecha |
| 15 | Derecha   | Derecha |

Tenga en cuenta que el tamaño del campo Location se ha aumentado de 1 a 2 dígitos en Visual FoxPro 6.0. Éste es el único cambio en Visual FoxPro 6.0 en las [estructuras de las tablas](#), incluidas las tablas de bases de datos (.dbc), de formularios (.scx), de etiquetas (.lbx), de proyectos (.pjx), de informes (.frx) y bibliotecas de clases visuales (.vcx).

### Eventos CommandTargetExec y CommandTargetQuery

Dos eventos de documentos activos, CommandTargetExec y CommandTargetQuery, permiten administrar las selecciones de los menús de Internet Explorer (y otros eventos de Internet Explorer) desde un documento activo. El evento CommandTargetExec se produce cuando Internet Explorer notifica a un documento activo que se va a ejecutar un comando (que puede ser un comando de menú). El evento CommandTargetQuery ocurre cuando Internet Explorer actualiza su interfaz de usuario. Si desea obtener más información sobre estos eventos, consulte [CommandTargetExec \(Evento\)](#) y [CommandTargetQuery \(Evento\)](#) en la *Referencia del lenguaje*.

### Ejecutar documentos activos

Para ejecutar los documentos activos de Visual FoxPro son necesarios los archivos Vfp6.exe y Vfp6run.exe, o Vfp6run.exe, Vfp6r.dll y Vfp6res.dll (es indica que se trata de la versión en español). Estos archivos deben estar instalados y registrados en el equipo en el que está instalado Internet Explorer. Cuando se instala Visual FoxPro, Vfp6.exe se instala en el directorio de Visual FoxPro y los archivos restantes en el directorio Windows\System de Windows 95 o en el directorio WinNT\System32 de Windows NT.

### Ejecutar documentos activos desde el menú Herramientas

El menú Herramientas de Visual FoxPro contiene el comando **Ejecutar documento activo** que muestra el cuadro de diálogo **Ejecutar documento activo**, en el que puede especificar cómo se va a ejecutar un documento activo. Están disponibles las siguientes opciones:

| <b>Opción</b>                     | <b>Descripción</b>  |
|-----------------------------------|---|
| En el explorador (Predeterminada) | El documento activo se ejecuta en Internet Explorer en el entorno de tiempo de ejecución de Visual FoxPro.  |
| Independiente                     | El documento activo se ejecuta como una aplicación independiente con el entorno de tiempo de ejecución de Visual FoxPro.  |
| En el explorador (Depuración)     | El documento activo se ejecuta en Internet Explorer con el ejecutable de Visual FoxPro (Vfp6.exe). Estarán disponibles las herramientas de depuración, la ventana Comandos y todas las características del entorno de programación de Visual FoxPro.              |
| Independiente (Depuración)        | El documento activo se ejecuta como una aplicación independiente con el ejecutable de Visual FoxPro (Vfp6.exe). Estarán disponibles las herramientas de depuración, la ventana Comandos y todas las características del entorno de programación de Visual FoxPro. |
|                                   | Esta opción equivale a ejecutar DO <Nombre de Active Doc> en la ventana Comandos.   |

También puede ejecutar un documento activo si lo abre en el cuadro de diálogo Abrir archivo de Internet Explorer o si se desplaza hasta el documento activo desde otra página Web que contenga un vínculo al mismo.

### **El entorno de tiempo de ejecución de Visual FoxPro y los documentos activos**

Desde Visual FoxPro puede ejecutar un documento activo si hace doble clic en el icono del mismo en el Explorador de Windows. También puede ejecutar un documento activo desde una aplicación en el entorno de tiempo de ejecución de Visual FoxPro. El entorno de tiempo de ejecución de Visual FoxPro consta de dos archivos, Vfp6run.exe y Vfp6r.dll. Ambos deben estar instalados y registrados para poder ejecutar documentos activos. También se puede utilizar el entorno de tiempo de ejecución para ejecutar otros archivos Visual FoxPro distribuibles, como programas de Visual FoxPro compilados (archivos .fxp), por ejemplo.

Una vez registrado, puede utilizar Vfp6run.exe para ejecutar directamente documentos activos (y otros archivos Visual FoxPro distribuibles).

### **Sintaxis de Vfp6run.exe**

```
VFP6RUN [/embedding] [/regserver] [/unregserver] [/security]
[/s] [/version] [NombreArchivo]
```

### **Argumentos**

/embedding

Carga Vfp6run.exe como un servidor de documentos activos. En este modo, Vfp6run.exe se registra como servidor COM capaz de crear un objeto de tipo documento activo de Visual FoxPro ("Visual.FoxPro.Application.6"). Sin este argumento, Vfp6run.exe no actúa como servidor COM.

/regserver

Registra Vfp6run.exe.

/unregserver

Elimina Vfp6run.exe del registro.

/security

Muestra el cuadro de diálogo **Configuración de seguridad de aplicaciones**, que permite especificar las opciones de seguridad para los documentos activos y para otros archivos de aplicación (.app). Si desea obtener más información, consulte la siguiente sección, "Seguridad de documentos activos".

/s

Silenciosa. Especifica que se generará un error si Vfp6run.exe no puede cargar el componente de tiempo de ejecución Vfp6r.dll.

/version

Muestra la información de versión de Vfp6run.exe y Vfp6r.dll.

*NombreArchivo*

Especifica el archivo Visual FoxPro que se va a ejecutar.

Vfp6run.exe requiere que la biblioteca de vínculos dinámicos de soporte en tiempo de ejecución, Vfp6r.dll, esté instalada y registrada. Para registrar Vfp6r.dll, ejecute Regsvr32 con el nombre del entorno de tiempo de ejecución:

```
Regsvr32 Vfp6r.dll
```

## Seguridad de documentos activos

La opción /security del entorno de tiempo de ejecución de Visual FoxPro Vfp6run.exe permite establecer niveles de seguridad para los documentos activos y para otros archivos de aplicación (.app). Al ejecutar Vfp6run.exe /security aparece el cuadro de diálogo **Configuración de seguridad de aplicaciones**, en el que puede establecer niveles de seguridad para los documentos activos y para otros archivos .app.

En el cuadro de diálogo **Configuración de seguridad de aplicaciones** están disponibles las siguientes opciones:

**Alojado**

Elija esta opción de modo de aplicación para especificar el nivel de seguridad de un documento activo o de una aplicación (.app) que se ejecute desde un contenedor de documentos activos, como Internet Explorer.

**No alojado**

Elija esta opción de modo de aplicación para especificar el nivel de seguridad de un documento activo o de una aplicación (.app) que se ejecute desde el Explorador de Windows al hacer doble clic en su icono o que se ejecute con el entorno de tiempo de ejecución de Visual FoxPro, Vfp6run.exe.

**Alta (seguridad máxima)**

Elija esta opción para impedir que se ejecute un documento activo o una aplicación (.app).

**Media (más seguro)**

Elija esta opción para que aparezca una advertencia antes de que se ejecute un documento activo o una aplicación (.app). Ésta es la opción predeterminada para los documentos activos y aplicaciones que no se ejecutan en un contenedor.

**Baja (ninguna seguridad)**

Elija esta opción para ejecutar un documento activo o una aplicación (.app) sin que aparezca ninguna advertencia. Ésta es la opción predeterminada para los documentos activos y aplicaciones que se ejecutan en un contenedor.

**Restablecer**

Restaura el nivel de seguridad predeterminado para el modo de aplicación seleccionado (con contenedor o sin contenedor).

**Aceptar**

Guarda las opciones elegidas en el cuadro de diálogo.

**Notas sobre Internet Explorer**

Con el fin de aumentar el rendimiento, Internet Explorer 3.0 guarda en memoria caché las cuatro últimas páginas visitadas. Esto significa que un documento activo puede quedar fuera de la memoria caché de Internet Explorer 3.0 y se producirá el evento ContainerRelease. Internet Explorer 4.0 no mantiene una memoria caché de páginas, de forma que el evento ContainerRelease se produce en cuanto abandona el documento activo.

**Ejemplo de documento activo**

La aplicación de ejemplo Solutions de Visual FoxPro incluye un ejemplo llamado “Crear documentos activos para el Web” que ilustra muchas de las características de los documentos activos.

### Para ejecutar la aplicación de ejemplo Solutions

- Escriba la siguiente línea en la ventana **Comandos**:

```
DO (HOME(2) + 'solution\solution')
```

– o bien –

1. En el menú **Programa**, elija **Ejecutar**.
2. Elija la carpeta ...\**Samples\Vfp98\Solution**.
3. Haga doble clic en **Solution.app**.

### Para ejecutar el ejemplo “Crear documentos activos para el Web”

1. Después de iniciar Solution.app, haga doble clic en **Nuevas características de Visual FoxPro 6.0**.
2. Haga clic en **Crear documentos activos para el Web** y, a continuación, haga clic en el botón **Ejecutar ejemplo**.

El ejemplo “Crear documentos activos para el Web” permite abrir un proyecto que contiene todos los archivos necesarios para crear un documento activo desde un proyecto. Cuando el proyecto está abierto, puede examinar el código de la clase Actdoc para ver cómo se administran los eventos de documentos activos y cómo se ejecutan los formularios. Tenga en cuenta que Actdoc, una clase basada en la clase de base ActiveDoc, es el archivo principal del proyecto. Un documento activo debe tener como archivo principal una clase basada en la clase de base ActiveDoc.

También puede generar un documento activo desde el proyecto; para ello, elija **Generar** en el Administrador de proyectos. Una vez generado el documento activo, elija **Ejecutar documento activo** en el menú **Herramientas** para ejecutarlo.

## Mejoras en servidores de Automatización

En este tema se describen las mejoras realizadas en los servidores de Automatización de Visual FoxPro 6.0 y se incluyen explicaciones sobre la forma en que los servidores de Automatización de Visual FoxPro pueden funcionar con productos y tecnologías como Microsoft Transaction Server y Microsoft Visual Basic.

Visual FoxPro permite crear servidores de Automatización: aplicaciones de componentes que ofrecen funcionalidad que otras aplicaciones pueden usar y reutilizar a través de Automatización. Por ejemplo, con Visual FoxPro puede crear un servidor de Automatización que muestre formularios reutilizables (en un ejecutable fuera de proceso) o que empaquete una rutina compleja en un

componente simple disponible para otros programadores. Además, puede crear una o más clases para controlar las normas de negocios de toda una compañía. Una aplicación cliente que utilice los objetos de normas de negocios pasará los parámetros de entrada de una llamada a un método y el servidor de Automatización realizará numerosas operaciones para recuperar o almacenar datos de diversas fuentes y realizar cálculos complejos antes de devolver los resultados.

En Visual FoxPro puede crear servidores de Automatización [fuera de proceso](#) o [en proceso](#). Un componente *fuera de proceso* es un archivo ejecutable (.exe) que se ejecuta en su propio proceso. La comunicación entre una aplicación cliente y un servidor fuera de proceso se denomina, por lo tanto, comunicación *entre procesos*. Un componente *en proceso* es un archivo de biblioteca de vínculos dinámicos (.dll) que se ejecuta en el mismo espacio de direcciones que el proceso del cliente que lo llama o en un proceso de Microsoft Transaction Server.

Para obtener más información sobre la creación de servidores de Automatización en Visual FoxPro, vea [Crear servidores de Automatización](#), en el capítulo 16 del *Manual del programador*.

## Mejoras de los servidores de Automatización en Visual FoxPro 6.0

En los temas siguientes se describen las características nuevas y mejoradas de los servidores de Automatización en Visual FoxPro 6.0.

### Modelo de subprocesamiento controlado

Los servidores de Automatización de Visual FoxPro son compatibles ahora con el modelo de subprocesamiento controlado. Microsoft Transaction Server aprovecha la funcionalidad de los servidores marcados como servidores de subprocesamiento controlado y ofrece una mejor protección y escalabilidad de subprocesos.

En cada objeto del modelo controlado (por ejemplo, un servidor de Automatización de Visual FoxPro) sólo puede entrar el subproceso que lo creó (por ejemplo, con la llamada a CoCreateInstance en Microsoft Visual C++). Sin embargo, un servidor de objetos, como Microsoft Transaction Server, puede admitir múltiples objetos, en cada uno de los cuales entran simultáneamente subprocesos diferentes. Los datos comunes que mantiene el servidor de objetos se deben proteger contra colisiones entre subprocesos. El servidor de objetos crea un objeto del modelo de subprocesamiento controlado en el mismo subproceso que llamó a CoCreateInstance. Las llamadas al objeto desde el subproceso controlado no están resueltas.

Para obtener más información sobre el modelo de subprocesamiento controlado, busque “Apartment-Model Threading in Visual Basic” en la biblioteca MSDN.

### Interfaces de usuario y servidores en proceso

La nueva característica del modelo de subprocesamiento controlado requiere que los servidores de Automatización .dll en proceso no tengan interfaces de usuario. En Visual FoxPro 5.0 era posible (aunque no recomendable) crear un servidor de Automatización .dll en proceso con una interfaz de usuario, como un formulario. El formulario sólo se podía mostrar ya que los eventos del formulario no eran compatibles. En Visual FoxPro 6.0, cualquier intento de crear una interfaz de usuario en un servidor de Automatización .dll en proceso producirá un error.

Por el contrario, un servidor de Automatización .exe fuera de proceso sí puede tener una interfaz de usuario. Se ha agregado una nueva función a Visual FoxPro 6.0, [SYS\(2335\)](#), que permite desactivar los eventos modales de un servidor de Automatización .exe fuera de proceso de forma que se pueda ejecutar de forma remota sin intervención del usuario. Los eventos modales se crean mediante formularios modales definidos por el usuario, cuadros de diálogo del sistema, la función MESSAGEBOX( ), el comando WAIT, etc.

### **Enlace en tiempo de compilación (vtable)**

Visual FoxPro 6.0 permite ahora el enlace en tiempo de compilación (vtable) además de la interfaz IDispatch existente (lo que se conoce conjuntamente como interfaz dual). El enlace en tiempo de ejecución (vtable) mejora el rendimiento de los controladores de Automatización que admiten esta técnica, como Visual Basic y Microsoft Transaction Server.

### **Entorno de tiempo de ejecución de Visual FoxPro Vfp6r.dll**

Ya no hay un único entorno de tiempo de ejecución de Visual FoxPro 6.0, Vfp6r.dll, que dé servicio a múltiples servidores de Automatización .dll en proceso. Cada .dll en proceso utiliza ahora una instancia independiente del entorno de tiempo de ejecución Vfp6r.dll. Las reglas siguientes determinan la forma en que las .dll en proceso utilizan el entorno de tiempo de ejecución Vfp6r.dll:

- La .dll en proceso que se llamó en primer lugar tiene el uso exclusivo de la biblioteca del entorno de tiempo de ejecución Vfp6r.dll (normalmente instalada en la carpeta System de Windows 95 o en la carpeta System32 de Windows NT).
- Si una .dll en proceso ya tiene el uso exclusivo de Vfp6r.dll, por cada nueva .dll en proceso que se llame se creará en el disco y se cargará en memoria una copia de Vfp6r.dll con un nombre diferente. El nombre asignado al entorno de tiempo de ejecución Vfp6r.dll se basa en el nombre de la biblioteca .dll en proceso. Por ejemplo, si se llama a una .dll en proceso denominada Miservidor.dll, se asignará a la copia de Vfp6r.dll el nombre Miservidorr.dll (observe la “r” anexada al nombre), y se cargará en memoria para dar servicio a la .dll en proceso.
- Los entornos de tiempo de ejecución de Visual FoxPro sólo cambian de nombre para las .dll en proceso que se ejecutan en el mismo proceso. Esto significa que dos clientes independientes, cada uno ejecutándose en su propio proceso, pueden cargar dos .dll en proceso diferentes de Visual FoxPro sin cambiar el nombre del entorno de tiempo de ejecución. En este caso, ambas .dll en proceso de Visual FoxPro utilizarán Vfp6r.dll ya que los clientes se cargan en procesos separados.
- Cuando hay múltiples servidores de Automatización (creados con OLEPUBLIC en DEFINE CLASS) en una misma .dll en proceso, todos compartirán el mismo entorno de tiempo de ejecución Vfp6r.dll. En este caso, es posible que los servidores de Automatización se afecten mutuamente al compartir variables de memoria públicas, al establecer los mismos comandos SET, etc. Evite que los servidores de Automatización de una misma .dll en proceso interfieran entre sí.

### **Bibliotecas de tipos**

Visual FoxPro 6.0 admite ahora propiedades, eventos y métodos intrínsecos (de Visual FoxPro) en bibliotecas de tipos de servidores de Automatización. Solamente las propiedades declaradas como Public se incluyen en la biblioteca de tipos. Las propiedades protegidas y ocultas no aparecen en la biblioteca. Tenga en cuenta que el método Release de Visual FoxPro no está incluido en la biblioteca de tipos, pues ya existe como método COM.

Tanto los métodos como las propiedades PUBLIC personalizadas definidas por el usuario aparecen en las bibliotecas de tipos de Visual FoxPro, siempre que estén marcadas como Public. Para los métodos, Visual FoxPro incluye también un tipo de valor de retorno (de tipo variant) y una lista de parámetros (de tipo variant) que se analizan a partir de la definición del método original.

Tenga en cuenta que en Visual FoxPro 6.0 sólo puede designarse un archivo de Ayuda como biblioteca de tipos.

## Tratamiento de excepciones

Los servidores de Automatización de Visual FoxPro son ahora más robustos, por lo que pueden terminar una operación con menos problemas cuando se produce una excepción. Cuando ocurre una excepción en un servidor de Automatización de Visual FoxPro 6.0, el servidor establece el objeto COM ErrorInfo (a través de IErrorInfo) y cancela el método actual. El cliente de Automatización tiene la opción de liberar el servidor de Automatización de Visual FoxPro o tratar la excepción, dependiendo de la información del objeto COM ErrorInfo (y el cliente tiene acceso al objeto COM ErrorInfo).

Una nueva función agregada a Visual FoxPro 6.0, COMRETURNERROR( ), permite tratar los errores que se producen en un servidor de Automatización. COMRETURNERROR( ) se puede usar en el método Error; rellena la estructura de la excepción COM con información que los clientes de Automatización pueden utilizar para determinar el origen de los errores de servidor de Automatización. Para obtener más información, consulte [COMRETURNERROR\(\)](#) en la *Referencia del lenguaje*.

## Pasar matrices

Visual FoxPro 5.0 pasa por valor las matrices a los objetos COM (por ejemplo, los servidores de Automatización creados con Visual FoxPro, Visual Basic o Visual C); los elementos de las matrices son los mismos después de una llamada a un método y los cambios en el objeto COM no se propagan a los elementos del cliente. Esta restricción evita el paso de grandes cantidades de datos entre Visual FoxPro 5.0 y los objetos COM.

Además, se asume que la matriz pasada al objeto COM hace referencia al primer elemento, fila y columna con el número uno (por ejemplo, Mimatriz[1]). Sin embargo, algunos objetos COM requieren que la matriz pasada haga referencia al primer elemento, fila y columna con el número cero, como Mimatriz [0].

Una nueva función de Visual FoxPro 6.0, COMARRAY( ), permite especificar la forma de pasar una matriz a un objeto COM y si está basada en cero o en uno. Para obtener más información, consulte [COMARRAY\(\)](#) en la *Referencia el lenguaje*.

Tenga en cuenta que COMARRAY( ) sólo se utiliza cuando se pasan matrices a objetos COM con la sintaxis siguiente:

```
oObjetoCom.Metodo(@MiMatriz)
```

Si se omite el símbolo @, sólo se pasará al objeto COM el primer elemento de la matriz y COMARRAY( ) no tendrá ningún efecto. Éste es el comportamiento en las versiones anteriores de Visual FoxPro.

## Generar archivos .dll y .exe a partir de proyectos

Debido a que los servidores de Automatización .dll en proceso y .exe fuera de proceso se invocan a través de la creación de instancias de clases, no es necesario especificar un [archivo principal](#) para los mismos. En Visual FoxPro 6.0 puede generar un servidor de Automatización .dll en proceso o .exe fuera de proceso sin tener que especificar primero un archivo principal en el Administrador de proyectos.

## Lenguaje

En la tabla siguiente se muestran las propiedades y funciones que se han agregado a Visual FoxPro 6.0 para facilitar la administración de clientes y servidores de Automatización. Para obtener más información, consulte los temas indicados.

| <b>Nuevos elementos del lenguaje para mejoras en el servidor</b> | <b>Descripción</b>   |
|--|--|
| <a href="#">Función COMARRAY( )</a>                              | Especifica cómo se pasan las matrices a los objetos COM.   |
| <a href="#">Función COMCLASSINFO( )</a>                          | Devuelve información del registro sobre un objeto COM, como un servidor de Automatización de Visual FoxPro.  |
| <a href="#">Función CREATEOBJECTEX( )</a>                        | Crea una instancia de un objeto COM registrado (por ejemplo, de un servidor de Automatización de Visual FoxPro) en un equipo remoto. Puede utilizar Microsoft Transaction Server para crear una instancia de una .dll en proceso de Visual FoxPro en un equipo remoto. |
| <a href="#">Función COMRETURNERROR( )</a>                        | Rellena la estructura de excepción COM con información que los clientes de Automatización pueden utilizar para determinar el origen de los errores de un servidor de Automatización.   |
| <a href="#">Propiedad ServerName</a>                             | Contiene la ruta completa y el nombre de archivo de un servidor de Automatización. La propiedad ServerName corresponde al objeto Application.  |
| <a href="#">Propiedad StartMode</a>                              | Contiene un valor numérico que indica cómo se inició la instancia de Visual FoxPro.  |
| <a href="#">SYS(2334) – Modo de invocación de</a>                | Devuelve un valor que indica cómo se ha invocado un  |

---

|  |   |
|--|---|
| <a href="#">servidor de Automatización</a>                   | método de un servidor Automation de Visual FoxPro.  |
| <a href="#">SYS(2335) – Modo de servidor sin supervisión</a> | Activa o desactiva el uso de estados modales en los servidores de Automatización .exe distribuibles de Visual FoxPro. |

---

## Notas sobre la programación de servidores de Automatización

En esta sección se ofrece información adicional sobre la programación de servidores de Automatización.

### El objeto Application

El [objeto Application](#) no se expone en una biblioteca de tipos de un servidor de Automatización. De este modo se impide el acceso a los métodos [DoCmd](#) y [Eval](#) del objeto Application, que potencialmente podrían dar acceso al lenguaje completo de Visual FoxPro. Puede exponer el objeto Application si crea una propiedad personalizada y le asigna el valor de objeto Application. También puede proporcionar un método con acceso al objeto Application.

### Ejemplos de servidores de Automatización

Visual FoxPro 6.0 incluye dos servidores de Automatización ISAPI de ejemplo, FoxWeb y FoxIS. Estos ejemplos administran el envío de registros seleccionados de datos de Visual FoxPro con formato HTML a un explorador de Internet. Para obtener más información sobre estos ejemplos, vea [FoxISAPI: ejemplo de servidor OLE](#).

# Capítulo 32: Programación de aplicaciones y productividad del programador

Microsoft FoxPro siempre ha proporcionado las herramientas necesarias para la programación de aplicaciones con la aplicación FoxPro y el lenguaje XBase. Visual FoxPro ha agregado el lenguaje y los comportamientos para la orientación a objetos. Esta versión de Visual FoxPro incluye un marco de trabajo mejorado, así como herramientas de creación y mantenimiento de objetos diseñadas para ayudar a una programación de aplicaciones más rápida y facilitar las tareas de mantenimiento.

En este capítulo se tratan los temas siguientes:

[Galería de componentes](#)

[Aplicación Analizador de trayecto](#)

[Enganches del Administrador de proyectos](#)

[Asistentes nuevos y mejorados](#)

[Marco de trabajo de aplicación mejorado](#)

## Galería de componentes

La Galería de componentes es un contenedor de catálogos de objetos de software tales como bibliotecas de clases, formularios, botones, etcétera. También contiene nuevas clases de Visual FoxPro. Puede utilizar la Galería de componentes para organizar los componentes por objetos, proyectos, aplicaciones u otros agrupamientos. Estos agrupamientos visuales pueden personalizarse dinámicamente de forma que pueda utilizar, duplicar o reorganizar los componentes de varias clasificaciones en la Galería de componentes. Puede tener acceso a un elemento específico desde cualquier punto de la Galería de componentes en el que coloque una referencia al mismo. También puede tener varias referencias a un mismo objeto en distintos catálogos o carpetas. Por ejemplo, un botón puede aparecer en una o más categorías de proyecto de la Galería de componentes (representadas como carpetas), pero también puede ser visible en una categoría llamada "Herramientas" que contenga referencias a todos los botones que utilice.

Puede utilizar la Galería de componentes con todas las funciones que proporcionan los componentes independientes [Administrador de proyectos](#), [Examinador de clases](#) y la [barra de herramientas](#) [Controles de formularios](#). Cada uno de los restantes componentes de Visual FoxPro proporciona un enfoque muy específico de los proyectos o clases desde el entorno especial del archivo de proyecto o de la biblioteca de clases. La Galería de componentes permite administrar las relaciones entre los componentes y muchos de sus comportamientos, desde un nivel abstracto de diseño y también desde una perspectiva de programación más concreta.

Puede arrastrar y colocar componentes dentro de la Galería de componentes y también desde la Galería de componentes a proyectos o formularios. También puede cambiar las propiedades de los objetos o clases desde la Galería de componentes.

La Galería de componentes puede contener cualquier elemento visual de Visual FoxPro, como documentos locales y remotos, archivos o carpetas, servidores de Automatización del tipo Microsoft Excel y Word, y ubicaciones y archivos HTML. También puede incluir archivos .prg con código de miniprogramas, clases, asistentes, generadores o elementos gráficos.

### Para abrir la Galería de componentes

- En el menú **Herramientas**, haga clic en la **Galería de componentes**.
- O bien -
- Escriba **DO** ([\\_GALLERY](#)) en la ventana **Comandos**.

### Administrar proyectos con la Galería de componentes

Puede utilizar la Galería de componentes para crear proyectos y aplicaciones, y también para administrar su desarrollo. La Galería de componentes permite organizar los componentes que contiene y puede utilizar sus plantillas, generadores y asistentes para crear el proyecto o la aplicación deseados.

### Para crear un proyecto o una aplicación desde la Galería de componentes

- Utilice el **Asistente para aplicaciones** o la **plantilla Nueva aplicación** de la carpeta **Aplicaciones** del catálogo de **Visual FoxPro**.

En el caso de los catálogos y carpetas, seleccione las fichas y opciones para realizar el cambio que desee. Si desea más información, vea el [cuadro de diálogo Opciones de la Galería de componentes](#).

### Mover y ver elementos en la Galería de componentes

Puede mover los elementos del panel derecho (Objeto) de la ventana de la Galería de componentes hasta el escritorio o hasta un proyecto o formulario abierto. El Administrador de proyectos reconoce el elemento al que se refiere la Galería de componentes y lo coloca en el lugar apropiado. Los elementos de la Galería de componentes colocados en el escritorio no son funcionales. No hay ninguna representación en el escritorio para las bases de datos, carpetas y elementos de la Galería de componentes que representan archivos no visuales.

### Para mover elementos desde la Galería de componentes

1. En el panel derecho, haga clic en el elemento que desee mover.

El [icono Mover](#), situado en la esquina superior izquierda de la ventana **Galería de componentes**, cambia de según el elemento seleccionado.

2. Arrastre y coloque el icono **Mover** en el escritorio o en un proyecto o formulario abierto.

Cuando la Galería de componentes no encuentra el elemento original representado por el elemento de galería, se abre un cuadro de diálogo Buscar en el que puede ayudarle a encontrarlo.

En la tabla siguiente se identifican los elementos de galería incluidos en Visual FoxPro, así como sus comportamientos predeterminados.

| Tipo de elemento de la Galería de componentes | Destinos de arrastrar y colocar |            |          |           |
|---|---------------------------------|------------|----------|-----------|
|   | Proyecto                        | Formulario | Pantalla | Controles |
| Clase (_ClassItem)                            |                                 |            | 6        |           |
| Archivo (_FileItem)                           |                                 |            |          |           |
| Dirección URL (_UrlItem)                      | 1                               |            |          |           |
| Formulario (_FormItem)                        | 9                               |            | 11       |           |
| Informe (_ReportItem)                         | 9                               |            | 11       |           |
| Programa (_ProgramItem)                       |                                 |            | 11       |           |
| Menú (_MenuItem)                              | 10                              |            | 11       |           |
| Imagen (_ImageItem)                           | 2                               |            | 7        | 2         |
| Sonido (_SoundItem)                           | 3                               |            |          |           |

|                           |    |
|---------------------------|----|
| Vídeo (_VideoItem)        | 3  |
| ActiveX (_ActiveXItem)    |    |
| Datos (_DataItem)         | 4  |
| Plantilla (_TemplateItem) | 5  |
| Catálogo (_CatalogItem)   | 8  |
| Ejemplo (_SampleItem)     |    |
| Proyecto (_ProjectItem)   | 11 |

- 1 – Agrega una clase de hipervínculo
- 2 – Agrega una clase de imagen o establece una propiedad Picture
- 3 – Agrega una clase multimedia
- 4 – Agrega una clase cuadrícula
- 5 – Según el tipo (por ejemplo, en un formulario) crea un nuevo archivo y lo agrega al proyecto
- 6 – Crea una instancia en Pantalla
- 7 – Establece el papel tapiz de Visual FoxPro
- 8 – Inicia una nueva ventana de la Galería con ese catálogo
- 9 – Agrega una clase de botones para iniciar un formulario o informe
- 10 – Agrega un menú contextual a un formulario
- 11 – Se abre en un diseñador (para modificación)

### Usar menús contextuales en la Galería de componentes

Puede hacer clic con el botón secundario en un elemento seleccionado del panel derecho (Objeto) para que aparezca un [menú contextual de elemento](#) con todas las acciones correspondientes, como **Agregar al proyecto** o **Agregar al formulario**. Puede utilizar el menú contextual para modificar o, en algunos casos, ejecutar el elemento de galería. Los menús contextuales son característicos de cada tipo de elemento de la galería. Puede cambiar algunas de las propiedades del elemento seleccionado con la opción **Propiedades** del menú contextual, que abrirá el [cuadro de diálogo Propiedades de elemento](#).

### Organizar los componentes de Visual FoxPro o Windows en grupos definidos por el usuario

Las carpetas de la Galería de componentes representan un agrupamiento arbitrario de los elementos de galería. Puede reorganizarlas con la técnica de arrastrar y colocar, o bien puede duplicarlos en otras carpetas. También puede copiar y cambiar el nombre de un catálogo o carpeta, o reordenar los elementos que contiene. Hay pocos límites, si es que hay alguno, en el modo de utilizar, modificar o crear catálogos o carpetas.

### Revisar y modificar clases

Al representar los elementos de la Galería de componentes elementos reales que pueden ser objetos o clases, para revisarlos o modificarlos puede lograr el acceso al objeto original a través de la Galería de componentes.

### Para revisar una clase

1. En la Galería de componentes, haga clic con el botón secundario en una clase.
2. En el menú contextual, haga clic en **Ver en el examinador**.

Se abrirá el [Examinador de clases](#), en el que puede ver las propiedades y métodos de la clase seleccionada.

### Para modificar una clase

1. En la Galería de componentes, haga clic con el botón secundario en una clase.
2. En el menú contextual, haga clic en **Modificar**.

Se abrirá la clase en el **Diseñador de clases**.

### Crear y modificar formularios

Puede utilizar la Galería de componentes para duplicar o modificar formularios y para agregar formularios y otros elementos de galería a un proyecto.

#### Para crear un formulario desde la Galería de componentes

- Haga doble clic en cualquier plantilla, o bien seleccione **Nuevo formulario** en el menú contextual de cualquier plantilla de la carpeta Formularios de la Galería de componentes.  
- O bien -
- Haga doble clic en el **Asistente para formularios** en la carpeta Formularios de la Galería de componentes.  
- O bien -
- Seleccione **Crear formulario** en el menú contextual de los elementos de la carpeta Formularios de la Galería de componentes.

#### Características de edición avanzadas de la Galería de componentes

La configuración predeterminada de los catálogos y formularios permite realizar tareas básicas de revisión y administración de elementos de la galería. Si desea modificar las características de los catálogos o carpetas, o si desea tener un mayor acceso a las propiedades de galería, seleccione **Modificación avanzada habilitada** en el [cuadro de diálogo Opciones de la Galería de componentes](#).

#### Catálogos de la Galería de componentes

Al abrir la Galería de componentes, el panel izquierdo (Catálogo) muestra el catálogo predeterminado

incluido en la Galería de componentes. Un catálogo es una representación visual de los elementos que pertenecen a un grupo de Visual FoxPro o a un grupo definido por el usuario. Dentro de cada catálogo, puede crear carpetas para organizar con más detalle subgrupos de elementos. Los elementos pueden ser formularios, consultas, programas, plantillas, archivos gráficos, archivos de sonido u otros objetos. El catálogo predeterminado de Visual FoxPro en la Galería de componentes incluye elementos agrupados en varias categorías, como formularios y controles, entre otros. También incluye una carpeta vacía llamada Favoritos, que puede utilizar para crear o copiar en ella elementos de la galería. Puede copiar y cambiar el nombre del catálogo predeterminado, o crear los suyos propios.

Las opciones de catálogo de la Galería de componentes determinan el contenido del catálogo y su comportamiento al abrirlo. Los catálogos *globales* pueden contener cualquier tipo de elemento de la Galería de componentes. Los *predeterminados* se abren automáticamente al iniciar la Galería de componentes. Si desea más información, vea la [ficha Catálogos](#) del cuadro de diálogo Opciones de la Galería de componentes .

La Galería de componentes incluye los catálogos siguientes.

| <b>Catálogo</b>    | <b>Descripción</b>  |
|--------------------|---|
| VFPGLRY            | Contiene componentes que utilizan otros catálogos de la galería. Contiene todos los catálogos incluidos con Visual FoxPro. Catálogo predeterminado y global.                    |
| Visual FoxPro      | Contiene las clases Visual FoxPro Foundation Classes. Catálogo predeterminado.  |
| Favoritos          | Carpeta vacía. Catálogo global.   |
| Mis clases de base | Contiene clases derivadas de las clases de base de Visual FoxPro. Catálogo predeterminado.  |
| ActiveX            | Catálogo dinámico que contiene una lista de todos los controles ActiveX registrados, o bien una lista de todos los controles ActiveX de Visual FoxPro. Catálogo predeterminado. |
| World Wide Web     | Colección de direcciones URL de sitios Web.   |
| Multimedia         | Diversos elementos con imágenes, sonidos y vídeos que puede utilizar en sus aplicaciones.   |
| Ejemplos           | Referencias a los ejemplos Solutions, Tastrade, servidores ActiveX y cliente-servidor.  |

Al hacer clic en un catálogo en la vista de lista, el panel derecho (Objeto) muestra su contenido. Puede abrir otros catálogos si hace doble clic en ellos en uno de los dos paneles. En la carpeta Galería se incluyen varios catálogos.

## Personalizar la Galería de componentes

Si desea personalizar la Galería de componentes, puede cambiar el comportamiento predeterminado de los elementos de los catálogos, carpetas y galerías a través de los cuadros de diálogo Propiedades correspondientes.

### Para crear un catálogo de la Galería de componentes

1. Seleccione el botón **Opciones** de la [barra de herramientas de la Galería de componentes](#).
2. Haga clic en la [ficha Catálogos](#) del cuadro de diálogo **Opciones de la Galería de componentes**.
3. Haga clic en **Nuevo** y dé nombre al nuevo catálogo en el cuadro de diálogo **Abrir**.
4. Haga clic en **Aceptar**.
5. La **Galería de componentes** agregará el catálogo al árbol para que pueda empezar a utilizarlo como lo haría con cualquier otro catálogo existente.

### Para cambiar la configuración de un catálogo o carpeta

1. Haga clic con el botón secundario en el catálogo o carpeta.
2. En el menú contextual, haga clic en **Propiedades**.
3. En el [cuadro de diálogo Propiedades de catálogo](#) o en el [cuadro de diálogo Propiedades de carpeta](#), seleccione la ficha que contenga las opciones que desea configurar.

Los catálogos y las carpetas mostrados en el panel **Catálogo** de la [ventana Galería de componentes](#) pueden representar direcciones URL, carpetas o archivos del disco duro. Puede ver una carpeta como **vista de Web** o como **vista a nivel de explorador**, dependiendo de la forma en que especifique su nombre en la ficha General del cuadro de diálogo Propiedades de carpeta.

### Vistas de Web

Puede especificar direcciones URL o archivos como catálogos o elementos de galería. Al configurar un elemento como carpeta de galería, se abrirá automáticamente como vista de Web en el panel **Objeto** (derecho) al seleccionarlo en el panel **Catálogo**.

### Para configurar un catálogo o carpeta de galería como vista de Web

1. En el [cuadro de diálogo Propiedades de carpeta](#), seleccione la ficha **Nodo**.
2. En el campo **Carpeta dinámica**, especifique el nombre de la página Web o del archivo como se hace en los ejemplos siguientes:

`http:\\www.microsoft.com\\`

archivo:\\c:\mis documentos\paginatest.htm

archivo:\\c:\mis documenotss\archword.doc

Al resaltar el icono de vista Web en el **panel Catálogo**, la barra de herramientas se modifica para incluir los botones de navegación por el Web. La vista de Web reflejará la configuración del Explorador de Windows.

### Vistas a nivel de explorador

Puede especificar un directorio como carpeta o catálogo de galería que tiene las características del Explorador de Windows.

#### Para configurar un catálogo o carpeta de galería como vista a nivel de explorador

1. En el [cuadro de diálogo Propiedades de carpeta](#), seleccione la ficha **Nodo**.
2. En el campo **Carpeta dinámica**, especifique el nombre de una carpeta o de un archivo y una barra inversa (\), como en el ejemplo siguiente:

C:\Mis documentos\

**Nota** Esta especificación crea una vista de archivos reales, a diferencia de otras vistas de la Galería de componentes. En esta vista *sí* puede eliminar archivos del disco.

Para crear una vista a nivel de explorador que mantenga la protección de los archivos mostrados, especifique el destino con una designación de comodín, como en el ejemplo siguiente:

C:\Mis documentos\\*.\*

Evite el uso de comodines para crear carpetas dinámicas cuando prevea encontrar más de 512 elementos, a no ser que disponga de un equipo rápido con gran cantidad de RAM.

### Objetos miembros de la Galería de componentes

La Galería de componentes consta de una interfaz, cuyas clases están contenidas en Vfpglry.vcx y de elementos que hacen referencia a las clases Visual FoxPro Foundation Classes siguientes:

| Objeto                           | Descripción  | Biblioteca de clases |
|----------------------------------|--|----------------------|
| <a href="#">About Dialog</a>     | Proporciona un sencillo cuadro de diálogo Acerca de... para aplicaciones personalizadas. | _dialogs.vcx         |
| <a href="#">ActiveX Calendar</a> | Control calendario que puede asociarse a un campo de fecha.                              | _datetime.vcx        |
| <a href="#">Array Handler</a>    | Proporciona métodos para tratar  | _utility.vcx         |

|   |  |                |
|---|--|----------------|
|   | operaciones con matrices que no realizan las funciones de matrices nativas.  |                |
| <a href="#">Cancel Button</a>           | Libera un formulario y descarta los datos que queden almacenados en búfer.   | _miscbtns.vcx  |
| <a href="#">Clock</a>                   | Control de reloj simple para un formulario o contenedor.   | _datetime.vcx  |
| <a href="#">Conflict Catcher</a>        | Cuadro de diálogo para resolver los conflictos de filas que aparezcan al modificar con almacenamiento optimista en búfer.        | _dataquery.vcx |
| <a href="#">Cookies Class</a>           | Clase simple de Web para el tratamiento de cookies entre páginas Web.  | _internet.vcx  |
| <a href="#">Cross Tab</a>               | Genera una tabla de referencias cruzadas.  | _utility.vcx   |
| <a href="#">Data Edit Buttons</a>       | Conjunto completo de botones de modificación (como los que utilizan los Asistentes para formularios).                            | Wizbtns.vcx    |
| <a href="#">Data Navigation Buttons</a> | Grupo de botones de exploración Top, Next, Prev, Bottom y clase DataChecker para comprobar si hay conflictos al mover registros. | _datanav.vcx   |
| <a href="#">Data Navigation Object</a>  | Objeto de exploración no visual que otras clases pueden utilizar.  | _table.vcx     |
| <a href="#">Data Session Manager</a>    | Administra sesiones de datos y se ocupa de las actualizaciones.  | _app.vcx       |
| <a href="#">Data Validation</a>         | Intercepta conflictos de datos almacenados en búfer.   | _datanav.vcx   |
| <a href="#">DBF -&gt; HTML</a>          | Convierte un cursor de Visual FoxPro (.dbf) a HTML.  | _internet.vcx  |
| <a href="#">Distinct Values Combo</a>   | Realiza una búsqueda de valores únicos en el campo origen del control para rellenar un cuadro combinado.                         | _dataquery.vcx |
| <a href="#">Error Object</a>            | Tratamiento genérico de errores que funciona con código de objeto y también con código procedimental.                            | _app.vcx       |
| <a href="#">Field Mover</a>             | Cuadro de lista supermover que carga automáticamente campos del origen de datos actual.  | _movers.vcx    |
| <a href="#">File Registry</a>           | Proporciona un conjunto de funciones de  | Registry.vcx   |

|  |   |               |
|--|---|---------------|
|  | registro que devuelven información específica de la aplicación.   |               |
| <a href="#">File Version</a>             | Recupera información de los datos de versión de un archivo.   | _utility.vcx  |
| <a href="#">Filter Button</a>            | Muestra un cuadro de diálogo en el que especificar un filtro de datos para un campo determinado.                          | _table2.vcx   |
| <a href="#">Filter Dialog</a>            | Cuadro de diálogo que permite especificar condiciones de filtrado de los datos.   | _table.vcx    |
| <a href="#">Filter Expression Dialog</a> | Crea un cuadro de diálogo para expresiones de filtro avanzadas.   | _table.vcx    |
| <a href="#">Find (Findnext) Buttons</a>  | Conjunto de botones Buscar/Buscar siguiente genérico.   | _table.vcx    |
| <a href="#">Find Button</a>              | Busca un registro que satisfaga criterios específicos.  | _table.vcx    |
| <a href="#">Find Dialog</a>              | Cuadro de diálogo Buscar con opciones simples, tales como elección de campos.   | _table.vcx    |
| <a href="#">Find Files/Text</a>          | Utiliza el objeto COM Filer.DLL para buscar archivos.   | _utility.vcx  |
| <a href="#">Find Object</a>              | Crea un objeto genérico que busca un registro con criterios específicos.  | _table.vcx    |
| <a href="#">Font Combobox</a>            | Cuadro combinado que contiene las fuentes disponibles. También lo utilizan las clases tbrEditing y rtfControls.           | _format.vcx   |
| <a href="#">FontSize Combobox</a>        | Cuadro combinado que contiene los tamaños de fuente disponibles. También lo utilizan las clases tbrEditing y rtfControls. | _format.vcx   |
| <a href="#">Format Toolbar</a>           | Proporciona una barra de herramientas para aplicar formato de fuente al texto del control activo.                         | _format.vcx   |
| <a href="#">FRX -&gt; HTML</a>           | Convierte el resultado de un informe de Visual FoxPro (.frx) al formato HTML.   | _internet.vcx |
| <a href="#">GetFile and Directory</a>    | Recupera un nombre de archivo y de carpeta.   | _controls.vcx |
| <a href="#">Goto Dialog Button</a>       | Crea un botón que muestra el cuadro de diálogo Ir a.  | _table2.vcx   |
| <a href="#">Goto Dialog</a>              | Crea un cuadro de diálogo Ir a registro.  | _table.vcx    |

|  |   |                |
|--|---|----------------|
| <a href="#">Graph By Record Object</a> | Grupo de botones de exploración que permite actualizar un nuevo gráfico por cada registro instantáneamente.     | _utility.vcx   |
| <a href="#">Graph Object</a>           | Genera un gráfico con el motor del Asistente para gráficos.   | Autgraph.vcx   |
| <a href="#">Help Button</a>            | Muestra el archivo de Ayuda mientras comienza a buscar el HelpContextID especificado.                           | _miscbtns.vcx  |
| <a href="#">Hyperlink Button</a>       | Inicia un explorador de Web desde un botón.   | _hyperlink.vcx |
| <a href="#">Hyperlink Image</a>        | Inicia un explorador de Web desde una imagen.   | _hyperlink.vcx |
| <a href="#">Hyperlink Label</a>        | Inicia un explorador de Web desde una etiqueta.   | _hyperlink.vcx |
| <a href="#">INI Access</a>             | Conjunto de funciones de registro que permiten el acceso a las configuraciones de archivo del antiguo tipo INI. | Registry.vcx   |
| <a href="#">Item Locator</a>           | Este botón abre un cuadro de diálogo con el que puede buscar un registro.                                       | _dialogs.vcx   |
| <a href="#">Keywords Dialog</a>        | Crea un cuadro de diálogo similar al cuadro de palabras clave de la Galería de componentes.                     | _dialogs.vcx   |
| <a href="#">Launch Button</a>          | Inicia una aplicación con un documento opcional.  | _miscbtns.vcx  |
| <a href="#">Locate Button</a>          | Muestra un cuadro de diálogo con el que buscar un registro.   | _table2.vcx    |
| <a href="#">Lookup Combobox</a>        | Realiza una búsqueda de valores en un campo para rellenar un cuadro combinado.                                  | _dataquery.vcx |
| <a href="#">Mail Merge Object</a>      | Genera una combinación de Word Mail con el motor del Asistente para combinar correspondencia.                   | Mailmerge.vcx  |
| <a href="#">MessageBox Handler</a>     | Envoltura simple de la función MessageBox.  | _dialogs.vcx   |
| <a href="#">MouseOver Effects</a>      | Resalta un control cuando pasa el mouse sobre él.   | _ui.vcx        |
| <a href="#">Mover</a>                  | Proporciona una clase sencilla de cuadro de lista con movimiento y botones mover/quitar.                        | _movers.vcx    |

|  |  |                |
|--|--|----------------|
| <a href="#">Navigation Shortcut Menu</a> | Menú contextual que puede colocarse en un formulario.  | _table2.vcx    |
| <a href="#">Navigation Toolbar</a>       | Conjunto de botones de navegación en una barra de herramientas.  | _table2.vcx    |
| <a href="#">Object State</a>             | Determina el estado de un objeto y guarda o restablece la configuración de sus propiedades.                            | _app.vcx       |
| <a href="#">ODBC Registry</a>            | Conjunto de funciones de registro que devuelven información específica de ODBC.  | Registry.vcx   |
| <a href="#">Offline Switch</a>           | Proporciona una vista de datos con conexión para su uso sin conexión.  | _dataquery.vcx |
| <a href="#">OK Button</a>                | Realiza una liberación simple de formulario.   | _miscbtns.vcx  |
| <a href="#">Output Control</a>           | Muestra un cuadro de diálogo complejo que solicita al usuario una opción de resultado de informe.                      | _reports.vcx   |
| <a href="#">Output Dialog</a>            | Muestra un cuadro de diálogo que solicita al usuario una opción de resultado de informe.                               | _reports.vcx   |
| <a href="#">Output Object</a>            | Diversas opciones de resultado de informe.   | _reports.vcx   |
| <a href="#">Password Dialog</a>          | Sencillo cuadro de diálogo Contraseña para aplicaciones personalizadas.  | _dialogs.vcx   |
| <a href="#">Pivot Table</a>              | Genera una tabla dinámica de Microsoft Excel con el motor del Asistente para tablas dinámicas.                         | Pivtable.vcx   |
| <a href="#">Preview Report</a>           | Botón genérico para ejecutar un informe.   | _miscbtns.vcx  |
| <a href="#">QBF</a>                      | Proporciona un conjunto de botones para consultas de tipo Consulta por formulario.                                     | _dataquery.vcx |
| <a href="#">Registry Access</a>          | Proporciona acceso a la información del Registro de Windows.   | registry.vcx   |
| <a href="#">Resize Object</a>            | Hace que los objetos de un formulario cambien de tamaño y posición cuando se produce el evento Resize del objeto Form. | _controls.vcx  |
| <a href="#">RTF Controls</a>             | Proporciona un conjunto de botones para aplicar formato al texto del control activo.                                   | _format.vcx    |
| <a href="#">Run Form Button</a>          | Botón que ejecuta un formulario.   | _miscbtns.vcx  |

|   |   |                 |
|---|---|-----------------|
| <a href="#">SCX -&gt; HTML</a>                    | Convierte un formulario .scx al formato HTML.   | _internet.vcx   |
| <a href="#">SendMail Buttons</a>                  | Utiliza el control ActiveX de MAPI ActiveX para enviar un mensaje de correo desde un formulario.  | _miscbtns.vcx   |
| <a href="#">Shell Execute</a>                     | Proporciona el comportamiento de doble clic del Explorador de Windows.  | _environ.vcx    |
| <a href="#">Shortcut Menu Class</a>               | Esta clase de envoltura crea dinámicamente menús contextuales emergentes.   | _menu.vcx       |
| <a href="#">Simple Edit Buttons</a>               | Proporciona sencillos botones Agregar, Modificar, Eliminar, Duplicar, Guardar y Cancelar (como los de los Asistentes para formularios). | Wizbtns.vcx     |
| <a href="#">Simple Navigation Buttons</a>         | Proporciona un conjunto de botones de exploración Siguiente y Anterior.   | _table.vcx      |
| <a href="#">Simple Picture Navigation Buttons</a> | Conjunto de botones de exploración con imágenes sencillas.  | _table2.vcx     |
| <a href="#">Sort Button</a>                       | Muestra un cuadro de diálogo que permite ordenar los datos de un campo determinado de forma ascendente o descendente.                   | _table2.vcx     |
| <a href="#">Sort Dialog</a>                       | Permite realizar una ordenación ascendente o descendente de los datos de un campo determinado.  | _table2.vcx     |
| <a href="#">Sort Mover</a>                        | Esta subclase de la clase cuadro de lista supermover se ocupa de la ordenación de los datos.  | _movers.vcx     |
| <a href="#">Sort Object</a>                       | Realiza una ordenación de un origen de datos.   | _table.vcx      |
| <a href="#">Sort Selector</a>                     | Realiza una ordenación ascendente o descendente, basada en el control actual.   | _table2.vcx     |
| <a href="#">Sound Player</a>                      | Esta clase carga y reproduce un archivo de sonido.  | _multimedia.vcx |
| <a href="#">Splash Screen</a>                     | Proporciona una sencilla pantalla de inicio para aplicaciones personalizadas.   | _dialogs.vcx    |
| <a href="#">SQL Pass Through</a>                  | Proporciona paso a través de SQL y permite ejecutar procedimientos almacenados en la base de datos host.                                | _dataquery.vcx  |

|  |  |                 |
|--|--|-----------------|
| <a href="#">Stop Watch</a>                     | Proporciona un control de detención de inspección para un formulario o contenedor.   | _datetime.vcx   |
| <a href="#">String Library</a>                 | Realiza diversas conversiones de cadenas.  | _utility.vcx    |
| <a href="#">Super Mover</a>                    | Proporciona los botones Mover, Quitar, Mover todos y Quitar todos.   | _movers.vcx     |
| <a href="#">System Toolbars</a>                | Clase administrativa que maneja y hace un seguimiento de las barras de herramientas del sistema.   | _app.vcx        |
| <a href="#">Table Mover</a>                    | Esta subclase de la clase cuadro de lista supermover carga automáticamente tablas y campos desde el origen de datos actual.                    | _movers.vcx     |
| <a href="#">Text Preview</a>                   | Proporciona un visor del texto resultante.   | _reports.vcx    |
| <a href="#">Thermometer</a>                    | Proporciona una clase termómetro estándar.   | _controls.vcx   |
| <a href="#">Trace Aware Timer</a>              | Utilidad de aplicación que determina si la ventana de seguimiento está abierta.  | _app.vcx        |
| <a href="#">Type Library</a>                   | La rutina principal ExportTypeLib crea un archivo de texto con el resultado Typelib.   | _utility.vcx    |
| <a href="#">URL Combo</a>                      | Crea un cuadro combinado para escribir en una dirección URL de Web. Inicia Microsoft Internet Explorer y se sitúa en el sitio correspondiente. | _internet.vcx   |
| <a href="#">URL Open Dialog</a>                | Proporciona un cuadro de diálogo que crea una lista desplegable con el historial de direcciones URL.   | _internet.vcx   |
| <a href="#">VCR Buttons</a>                    | Grupo de botones de exploración Top, Next, Prev y Bottom.  | _table.vcx      |
| <a href="#">VCR Picture Navigation Buttons</a> | Conjunto de botones de exploración de imagen VCR.  | _table2.vcx     |
| <a href="#">Video Player</a>                   | Carga y reproduce un archivo de vídeo mediante comandos MCI.   | _multimedia.vcx |
| <a href="#">Web Browser control</a>            | Subclase de control Browser de Internet Explorer 4.0, que proporciona enganches con el código de Visual FoxPro.                                | _webview.vcx    |
| <a href="#">Window Handler</a>                 | Realiza diversas operaciones comunes de ventanas habituales en las aplicaciones.   | _ui.vcx         |

Si desea más detalles sobre estas bibliotecas de clases, vea el tema [Visual FoxPro Foundation Classes](#)

. Puede obtener información sobre cómo utilizar estas clases en el tema [Directivas para usar las Visual FoxPro Foundation Classes](#).

## Biblioteca de clases de la Galería de componentes (Vpfgallery.vcx)

La biblioteca de clases de la Galería de componentes, Vpfgallery.vcx, proporciona los tipos de elemento como clases.

| Tipo de elemento                 | Descripción  |
|----------------------------------|--|
| <b>Clase</b> (_ClassItem)        | Tipo de elemento genérico de cualquier clase de Visual FoxPro. Puede provenir de archivos .vcx o .prg.   |
| <b>Archivo</b> (_FileItem)       | Cualquier archivo. Visual FoxPro busca en el Registro funciones de shell y las agrega al menú. La galería incluye una rutina de búsqueda que comprueba si hay extensiones específicas y redirige el tipo de elemento.<br><br>La galería admite las convenciones UNC para la programación en grupo (compartir catálogos entre redes). |
| <b>ActiveX</b> (_ActiveXItem)    | Control o servidor ActiveX, como por ejemplo, un .ocx creado por Visual Basic CCE, o un archivo .exe o .dll creado por Visual FoxPro.  |
| <b>Datos</b> (_DataItem)         | Origen de datos de Visual FoxPro (.dbc, .dbf, Vista, etc.).  |
| <b>Imagen</b> (_ImageItem)       | Tipo de elemento de archivo cuya extensión corresponde a una imagen, como por ejemplo .bmp, .jpg, .gif, .ico, .cur, .ani, etcétera.  |
| <b>Sonido</b> (_SoundItem)       | Tipo de elemento de archivo cuya extensión es .wav o .rmi.   |
| <b>Vídeo</b> (_VideoItem)        | Tipo de elemento de archivo cuya extensión es .avi.  |
| <b>Dirección URL</b> (_UrlItem)  | Tipo de elemento de Web que incluye documentos de Web y locales, como por ejemplo archivos HTML o documentos activos de Visual FoxPro.   |
| <b>Ejemplo</b> (_SampleItem)     | Tipo de elemento de archivo para archivos que se ejecutan en Visual FoxPro y que pueden ser archivos ejecutables de Visual FoxPro, como por ejemplo .app, .exe, .prg, .scx o .frx.   |
| <b>Plantilla</b> (_TemplateItem) | Tipo de elemento de archivo de comandos que abre un generador para el elemento de Visual FoxPro representado por el tipo del elemento resaltado, como un formulario o un informe.  |
| <b>Catálogo</b> (_CatalogItem)   | Tipo de Galería de componentes que permite agregar y abrir catálogos de Visual FoxPro.   |
| <b>Formulario</b> (_FormItem)    | Tipo para formularios de Visual FoxPro (.scx).   |

|                                |  |
|--------------------------------|--|
| <b>Informe</b> (_ReportItem)   | Tipo para informes de Visual FoxPro (.frx).  |
| <b>Menú</b> (_MenuItem)        | Tipo para menús de Visual FoxPro (.mnx).     |
| <b>Programa</b> (_ProgramItem) | Tipo para programas de Visual FoxPro (.prg). |
| <b>Proyecto</b> (_ProjectItem) | Tipo para proyectos de Visual FoxPro (.pjx). |

Puede utilizar el Examinador de clases para examinar los detalles de cualquiera de las clases anteriores.

Si desea más información sobre otras clases utilizadas en la Galería de componentes, vea el tema [Visual FoxPro Foundation Classes](#) o bien utilice el [Examinador de clases](#) para examinar las bibliotecas de la carpeta Ffc.

### Tabla de estructura de la Galería de componentes

La estructura de la Galería de componentes se describe en la siguiente tabla.

| <b>Campo</b> | <b>Nombre de campo</b> | <b>Tipo</b> | <b>Ancho</b> | <b>Índice</b> |
|--------------|------------------------|-------------|--------------|---------------|
| 1            | TYPE                   | Character   | 12           | No            |
| 2            | ID                     | Character   | 12           | No            |
| 3            | PARENT                 | Memo        | 4            | No            |
| 4            | LINK                   | Memo        | 4            | No            |
| 5            | TEXT                   | Memo        | 4            | No            |
| 6            | TYPEDESC               | Memo        | 4            | No            |
| 7            | DESC                   | Memo        | 4            | No            |
| 8            | PROPERTIES             | Memo        | 4            | No            |
| 9            | FILENAME               | Memo        | 4            | No            |
| 10           | CLASS                  | Memo        | 4            | No            |
| 11           | PICTURE                | Memo        | 4            | No            |
| 12           | FOLDERPICT             | Memo        | 4            | No            |
| 13           | SCRIPT                 | Memo        | 4            | No            |
| 14           | CLASSLIB               | Memo        | 4            | No            |
| 15           | CLASSNAME              | Memo        | 4            | No            |
| 16           | ITEMCLASS              | Memo        | 4            | No            |
| 17           | ITEMTPDESC             | Memo        | 4            | No            |

|    |          |          |   |    |
|----|----------|----------|---|----|
| 18 | VIEWS    | Memo     | 4 | No |
| 19 | KEYWORDS | Memo     | 4 | No |
| 20 | SRCALIAS | Memo     | 4 | No |
| 21 | SRCRECNO | Numeric  | 6 | No |
| 22 | UPDATED  | DateTime | 8 | No |
| 23 | COMMENT  | Memo     | 4 | No |
| 24 | USER     | Memo     | 4 | No |

## Aplicación Analizador de trayecto

Una aplicación de trayecto escribe información sobre las líneas de código de un archivo que se han ejecutado. Una aplicación analizador de trayecto proporciona información sobre las líneas que se han ejecutado, cuántas veces se ha ejecutado cada una, su duración, etcétera. El [trayecto](#) y el [analizador de trayecto](#) permiten al programador identificar áreas problemáticas de una aplicación, especialmente código que se pasa por alto y cuellos de botella para el rendimiento.

El analizador de trayecto de Visual FoxPro se divide en dos partes: un motor de trayecto que puede utilizar o personalizar y una aplicación multiventana que puede utilizar para analizar los programas y proyectos.

La aplicación Analizador de trayecto ofrece distintas formas de ver los datos que proporciona el motor de trayecto. Coverage.app es una subclase del motor de trayecto. Puede automatizar el trayecto, modificar la interfaz de usuario para adaptarla a sus necesidades, ejecutar el Analizador de trayecto en modo no supervisado y no mostrar la ventana de la aplicación o utilizar las características del motor sin usar la interfaz.

Al iniciarse, la aplicación de trayecto suspende el registro de trayecto activado con un comando SET COVERAGE TO. Al liberar el objeto trayecto, la aplicación permite elegir si se desea restablecer el valor de SET COVERAGE.

### Archivo de registro del Analizador de trayecto

El Analizador de trayecto utiliza un archivo de registro que genera Visual FoxPro al utilizar la opción Trayecto del menú Herramientas del depurador o al utilizar [SET COVERAGE TO](#) como en el comando siguiente:

```
SET COVERAGE TO cTrayecto.log
```

Al utilizar el comando, la cláusula ADDITIVE permite no sobrescribir un registro ya existente. Este comando inicia el flujo de datos y abre *cTrayecto.log*, un archivo de texto que almacena el flujo de datos sobre el archivo o la aplicación examinados.

Un archivo de registro de trayecto consta de registros dispuestos en líneas delimitadas por comas. En la lista siguiente se describe la estructura de cada registro.

| Elemento | Descripción   |
|----------|---|
| 1        | tiempo de ejecución   |
| 2        | clase que ejecuta el código   |
| 3        | objeto, método o procedimiento en el que se encuentra o se invoca el código |
| 4        | número de línea dentro del método o procedimiento                           |
| 5        | archivo definido completamente  |
| 6        | nivel de pila de llamada (sólo Visual FoxPro 6.0)                           |

Una vez especificado el nombre del archivo de registro, ejecute el programa o aplicación que desee examinar. Cuando termine el programa, puede utilizar el comando SET COVERAGE TO para detener el flujo de datos hacia el registro de trayecto.

Para visualizar el registro de trayecto puede iniciar el Analizador de trayecto en el menú Herramientas o utilizar [DO](#) como en el comando siguiente:

```
DO (_COVERAGE) [WITH cCoverage]
```

Si no especifica un archivo de registro, Visual FoxPro le pedirá el nombre de uno. La variable de sistema [\\_COVERAGE](#) tiene como valor predeterminado en Visual FoxPro 6.0 la aplicación Analizador de trayecto, Coverage.app.

## Examinar el trayecto y el perfil de una aplicación

Para utilizar eficazmente el Analizador de trayecto, debe preparar cuidadosamente la aplicación y el entorno. Si sigue las directrices indicadas a continuación, el Analizador de trayecto le proporcionará información útil y precisa sobre su proyecto o aplicación.

### Para utilizar el Analizador de trayecto con el fin de examinar el trayecto de una aplicación

1. Utilice la opción **Registro de trayecto** del menú **Herramientas** del depurador, o bien ejecute el [comando SET COVERAGE](#) para iniciar el flujo de datos de trayecto y abrir el archivo para registrar los datos.
2. Ejecute el programa o aplicación cuyo trayecto desea examinar.
3. Ejecute la aplicación de trayecto en el menú **Herramientas** o bien utilice DO (\_COVERAGE) en la ventana de comandos.

La aplicación Analizador de trayecto se iniciará de forma predeterminada en **modo Trayecto**.

### Para utilizar el Analizador de trayecto con el fin de examinar el perfil de una aplicación

1. Utilice el [comando SET COVERAGE](#) para iniciar el flujo de datos de trayecto y abrir el

archivo en el que va a registrar los datos.

2. Ejecute el programa o aplicación que desea analizar.
3. Ejecute la aplicación de trayecto desde el menú **Herramientas**, o bien utilice DO ([\\_COVERAGE](#)) en la ventana de comandos.
4. Haga clic en el botón **Modo perfil** del cuadro de diálogo **Analizador de trayecto**.

Si le interesa un análisis con más frecuencia, puede cambiar la opción predeterminada por el modo perfil en el [cuadro de diálogo Opciones del Analizador de trayector](#).

### **Para utilizar el Analizador de trayecto con un archivo de trayecto específico**

Ejecute la aplicación de trayecto con la opción WITH y el nombre del archivo de registro como en el ejemplo siguiente:

```
DO (_COVERAGE) WITH "Mireg.LOG"
```

En este ejemplo se utiliza el archivo de registro Mireg.log y se abre la ventana de la aplicación Analizador de trayecto para mostrar el resultado. Si no especifica ningún nombre de archivo, el Analizador de trayecto utilizará el especificado en el comando SET COVERAGE TO actual o mostrará el cuadro de diálogo Abrir archivo si el registro de trayecto está desactivado (OFF).

### **Para utilizar el Analizador de trayecto sin la interfaz de usuario**

Ejecute la aplicación de trayecto con la opción WITH y especifique "verdadero" (.T.) para utilizar el modo no supervisado, como en el ejemplo siguiente.

```
DO (_COVERAGE) WITH "Mireg.LOG",.T.
```

En este ejemplo, la aplicación Analizador de trayecto utiliza el archivo de registro Mireg.log, y se ejecuta sin mostrar su ventana de aplicación.

### **Para utilizar el Analizador de trayecto con un archivo de complemento específico**

Ejecute la aplicación de trayecto con la opción WITH y el nombre del archivo de complemento, como en el ejemplo siguiente:

```
DO (_COVERAGE) WITH "Mireg.LOG",, "add_ui.prg"
```

En este ejemplo se utiliza el archivo de registro Mireg.log y se abre la ventana de la aplicación Analizador de trayecto para mostrar el resultado; y a continuación se ejecuta el programa complemento ADD\_UI.PRG. El segundo parámetro, no especificado, es un valor lógico que indica si el motor de trayecto funciona en modo no supervisado. Con la opción predeterminada, "falso" (.F.), aparece la ventana del Analizador de trayecto.

Además de ver la información de perfil, puede insertar en ella comentarios o marcas, y guardarla en un archivo para utilizarla posteriormente.

## Modificar el Analizador de trayecto

De forma predeterminada, la aplicación Analizador de trayecto se ejecuta en una ventana independiente. Si lo desea, puede cambiar la opción Entorno para que se ejecute dentro de la ventana principal de Visual FoxPro. En el [cuadro de diálogo Opciones del Analizador de trayector](#), cambie la selección de **Entorno** de **Marco de trayecto** a **Marco de FoxPro** y a continuación reinicie el Analizador de trayecto.

También puede utilizar el cuadro de diálogo **Opciones del Analizador de trayecto** para modificar las siguientes características del Analizador de trayecto:

| Característica       | Descripción   |
|----------------------|---|
| Complementos         | Especifica si se deben registrar los complementos en el Analizador de trayecto cuando se utilizan. Si desea más información, vea la sección "Complementos del Analizador de trayecto".                            |
| Marcas de trayecto   | Especifica si el Analizador de trayecto debe marcar el código que se ejecuta o el que no se ejecuta.<br>Especifica los caracteres empleados para marcar el código.<br>Especifica cuándo se debe marcar el código. |
| Fuentes              | Especifica las fuentes que utiliza el Analizador de trayecto para el código y las presentaciones.   |
| Búsqueda inteligente | Especifica si el Analizador de trayecto debe buscar automáticamente archivos en las ubicaciones especificadas anteriormente.  |
| Modo inicial         | Especifica si el Analizador de trayecto se debe abrir en el modo trayecto o en el modo perfil.  |

## Asegurar la corrección de los datos del Analizador de trayecto

Para ayudar a asegurar que los archivos que procesa el Analizador de trayecto son los correctos:

- Establezca el directorio del proyecto como predeterminado antes de iniciar el registro del trayecto, de forma que los archivos a los que haga referencia sean relativos a ese directorio.
- Evite cambiar dinámicamente el nombre de los objetos. El Analizador de trayecto no encontrará los objetos cuyo nombre haya cambiado en tiempo de ejecución.
- Evite el uso de archivos de origen cuyos nombres tengan exactamente la misma raíz, incluso cuando las extensiones sean distintas. Internamente, el Analizador de trayecto no puede distinguirlos.
- Asegúrese de que el proyecto sólo contiene las versiones correctas de los archivos modificados con frecuencia.
- Asegúrese de que el proyecto no contiene varias copias de un mismo archivo en subdirectorios

distintos.

- Realice una compilación para ejecutar el trayecto:
  - Asegúrese de que la aplicación contiene información de depuración.
  - Desactive el cifrado (OFF).
  - Utilice RECOMPILE o Generar todo para obligar a compilar todo el código fuente.
  - Realice la compilación inmediatamente antes de la ejecución del trayecto, para asegurar que el código fuente concuerde exactamente con el código objeto.

Algunas líneas del código, como los comentarios, las instrucciones DEFINE CLASS y ELSE, así como las líneas comprendidas entre TEXT y ENDTEXT no aparecen en los registros de trayecto, ya que no son ejecutables ni siquiera potencialmente. Además, las líneas interrumpidas por símbolos de continuación (caracteres punto y coma) se consideran como una sola línea de código y sólo se marca la última línea que las forma.

## Complementos del Analizador de trayecto

Los complementos son archivos de código (normalmente .prg o .scx) que proporcionan un método sencillo para reajustar el Analizador de trayecto. La subclase cov\_standard del motor de trayecto que engloba la interfaz de usuario de Coverage.app sólo muestra una pequeña parte de lo que puede hacer con el motor. El motor analiza el registro de trayecto; mientras que cov\_standard simplemente muestra el resultado en una de las numerosas formas en que puede obtenerse.

Puede crear una subclase de cov\_engine distinta con una presentación muy diferente. Por ejemplo, esa subclase podría mostrar un cuadro de diálogo que ejecutase consultas sobre las estadísticas de trayecto reunidas por el motor. Las opciones de presentación podrían ofrecer una vista del código marcado para un conjunto filtrado de entradas del registro o únicamente un gráfico con los resultados del perfil.

Es posible que no encuentre conveniente usar otra subclase de cov\_engine para crear una nueva interfaz desde cero, ya que la clase cov\_engine permite un proceso más sencillo: puede agregar funcionalidad a cov\_standard o a cualquier otra subclase de cov\_engine mediante complementos. Cov\_standard expone esta característica a través de un botón del cuadro de diálogo principal del Analizador de trayecto. Al ejecutar un complemento en una instancia de cov\_standard, como por ejemplo el Analizador de trayecto, el complemento puede manipular la funcionalidad de cov\_engine, las tablas de trayecto y también cov\_standard. Los complementos podrían además agregar nuevos cuadros de diálogo y otras características a la interfaz visual de cov\_standard.

## Escribir complementos

Puede escribir complementos para mejorar la interfaz estándar o bien puede usar una subclase de cov\_standard para crear su propia interfaz, completamente nueva.

## Mejorar la aplicación estándar

En la lista siguiente se señalan mejoras que podrían conseguirse mediante complementos:

- Agregar una característica visible al cuadro de diálogo principal.
- Agregar un cuadro de diálogo al conjunto de formularios del motor de trayecto (observe más abajo la limitación sobre la forma de asegurarse de que el cuadro de diálogo aparezca en el lugar correcto).
- Mostrar un cuadro de diálogo aparte para tener acceso a una característica del motor de trayecto (observe más abajo la limitación sobre la forma de asegurarse de que el cuadro de diálogo aparezca en el lugar correcto).
- Proporcionar una interfaz de consultas que utilice la tabla de origen y presente una lista con todas las líneas que satisfagan los criterios, y que filtre y ordene el resultado.

**Nota** Puede utilizar los métodos `Adjust...` del motor (`AdjustCoverageFilenameCursor()`, `AdjustSourceCursor()` y `AdjustTargetCursor()`) para agregar campos a las tablas de origen y destino cuando el motor las crea, y utilizar esos campos en los complementos.

- Agregar nombres de archivo al cursor `IgnoredFiles` para eliminarlos del análisis, lo que puede ahorrar tiempo.
- Utilizar en enlace especial `Init` para los complementos.
- Registrar los complementos para recuperarlos y tener acceso a los mismos fácilmente desde una lista.

La clase de diálogo modal `cov_AddInDialog` de la subclase del motor de trayecto estándar presenta en una lista desplegable los cuadros de diálogo registrados previamente. Al establecer `ON` en la opción `IRegisterAdd-In` del motor de trayecto, se agrega al Registro de Windows la ruta completa de los complementos ejecutados con éxito, lo que permite volver a ejecutarlos fácilmente. La clase `Standard UI` también permite establecer esta propiedad en el [cuadro de diálogo Opciones del Analizador de trayecto](#).

El objeto Motor de trayecto mantiene en la propiedad `aAddIns` una lista con todos los complementos registrados.

- Utilizar la información de los campos del archivo final `coverage.log`, la pila de llamadas, para diseñar su propia interfaz o su propia vista del registro de trayecto.

Al escribir complementos, considere lo siguiente:

- Puede utilizar cualquiera de los tipos de archivo admitidos como complementos. Los tipos de archivo admitidos son `.qpr`, `.qpx`, `.mpr`, `.mpx`, `.app`, `.exe`, `.scx`, `.fxp`, `.prg` y los procedimientos (si ya están disponibles en una biblioteca de procedimientos abierta).
- El conjunto de formularios del Motor de trayecto tiene una barra de herramientas "invisible". Si un complemento no es visual, puede utilizar esta barra de herramientas para que lo contenga. Si el complemento es un control visual, probablemente el cuadro de diálogo principal de la subclase estándar es el lugar más adecuado para ubicarlo. De este modo su posición y tamaño

se sincronizará automáticamente con el resto del cuadro de diálogo al cambiar el tamaño de éste.

- Todos los métodos del motor que utilizan las tablas de origen y destino tienen argumentos opcionales que permiten apuntarlos desde los alias correspondientes al trabajar con ellos. También puede cambiar el contenido actual de las propiedades `cSourceAlias` y `cTargetAlias` para hacerlo coincidir con el par de cursores que le interesen. Así podrá comparar diversas ejecuciones del registro de trayecto entre sí desde la misma interfaz.
- Limitaciones:
  - Los complementos deben aceptar un parámetro (el Motor de trayecto pasa una referencia a sí mismo).
  - Los complementos deben ser de uno de los tipos de archivo permitidos indicados anteriormente.
  - Los procedimientos que utilice como complementos deben estar disponibles en una biblioteca de procedimientos que se encuentre cargada (vea [SET PROCEDURE](#)). El Motor no utiliza la sintaxis `IN NombreArchivo` y no llama a los procedimientos ni a los archivos `.prg` como funciones, ni obtiene sus valores con `RETURN`. Tampoco utiliza las palabras clave `NAME` o `LINK` en el comando `DO FORM`, por lo que puede administrar usted mismo la referencia o hacer que el formulario sea miembro del conjunto de formularios del Motor para que éste lo alcance automáticamente.
  - Si ejecuta un complemento al inicio deberá utilizar una referencia, ya que la variable pública `_oCoverage` aún no estará disponible. En las demás ocasiones, puede utilizar la referencia a la variable pública en el código, si lo prefiere.
  - Al escribir un complemento como un formulario, si crea el formulario con `ShowWindow = 1` y ejecuta el trayecto en su propio marco, los complementos formulario deberán aparecer en el marco del trayecto.
  - Si utiliza `.RunAddIn` desde la ventana Comandos, asegúrese de que el marco del trayecto sea el marco MDI activo antes de instanciar los formularios.

### Crear una subclase de la clase `Cov_Standard`

Puede crear una subclase del motor de trayecto, su subclase estándar. En la lista siguiente se describe la estructura del conjunto de archivos de origen del proyecto `COVERAGE`.

| Archivo                        | Descripción   |
|--------------------------------|---|
| Coverage.prg                   | "Envoltura" del objeto trayecto, que instancia el objeto. |
| Coverage.vcx<br>Coverage.vct   | Todas las clases del motor y su subclase estándar.        |
| Cov_short.mnx<br>Cov_short.mnt | Menú contextual.  |

|                            |  |
|----------------------------|--|
| Cov_pjx.frx<br>Cov_pjx.frt | Mecanismo predeterminado para obtener resultados a nivel de proyecto.  |
| Coverage.h                 | <p>Archivo de encabezado para todo el código de trayecto, que incorpora los elementos siguientes:</p> <p>*— Constantes de caracteres de trayecto para registro y análisis:</p> <pre>#INCLUDE COV_CHAR.H</pre> <p>*— Cadenas de trayecto localizadas (puede utilizar algunas constantes de registro y análisis):</p> <pre>#INCLUDE COV_LOCS.H</pre> <p>*— Constantes de componentes de cuadros de diálogo comunes del trayecto:</p> <pre>#INCLUDE COV_DLGS.H</pre> <p>*— Especificaciones y requisitos del trayecto:</p> <pre>#INCLUDE COV_SPEC.H</pre> <p>*— Constantes de objetos del registro de trayecto:</p> <pre>#INCLUDE COV_REGS.H</pre> <p>*— Opciones de ajuste de trayecto:</p> <pre>#INCLUDE COV_TUNE.H</pre> |

El conjunto de archivos de origen del proyecto COVERAGE también incluye otros archivos .ico .bmp y .msk.

Puede utilizar el archivo COV\_TUNE.H (que contiene los comentarios y explicaciones correspondientes) para familiarizarse con las opciones disponibles sin tener que rescribir el código.

Como el uso de complementos está gobernado por la superclase del motor de trayecto, cualquier otra subclase de trayecto que cree podrá utilizar complementos de la misma forma en que lo hace la subclase estándar.

La subclase del motor de trayecto instanciada por la aplicación predeterminada Coverage.app no aumenta el método RunAddIn() del motor de trayecto de ningún modo. El cuadro de diálogo modal recibe una referencia al objeto Trayecto y establece la propiedad cAddIn del motor de trayecto.

Si escribe su propia subclase del motor de trayecto, asegúrese de que pueda utilizar la misma clase de cuadro de diálogo modal (cov\_AddInDialog) para tratar los complementos como lo hace la aplicación trayecto estándar. El cuadro de diálogo no se apoya en ninguna característica de la subclase estándar.

Puede llamar a un cuadro de diálogo modal distinto, establecer el nombre de archivo directamente en la propiedad `cAddIn` o anular el contenido de esta propiedad al pasar el nombre del archivo de complemento que desea ejecutar al método `RunAddIn()`.

Independientemente del modo en que logre el acceso a un complemento para ejecutarlo en una subclase, puede observar la lista de complementos registrados en `Coverage.app` si busca los nombres de los archivos en la propiedad `aAddIns` del motor de trayecto.

Si desea información sobre las propiedades, eventos y métodos del motor de trayecto, vea [Objeto Motor de trayecto](#).

## Enganches del Administrador de proyectos

En versiones anteriores de Visual FoxPro, el único acceso a un proyecto se lograba mediante la manipulación directa de las tablas del archivo `.pjx` del proyecto. En Visual FoxPro 6.0, el acceso a un proyecto puede realizarse mediante programa, lo que permite manipular el proyecto como un objeto. Un proyecto puede manipularse en tiempo de diseño, mientras está abierto en el Administrador de proyectos, o en tiempo de diseño y en tiempo de depuración sin que el Administrador de proyectos esté visible.

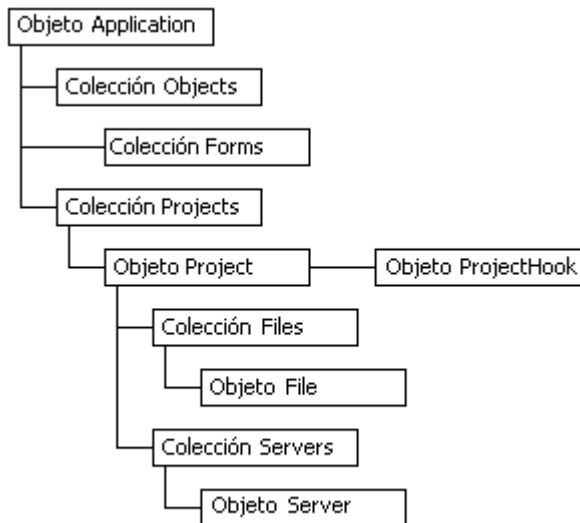
Las siguientes son algunas de las acciones que pueden realizarse sobre un proyecto mediante programación:

- Agregar o eliminar archivos de un proyecto.
- Agregar archivos del proyecto a aplicaciones de control de código fuente (como por ejemplo Microsoft Visual SourceSafe) y proteger o desproteger archivos con control de código fuente.
- Determinar el número de archivos de un proyecto y sus tipos.
- Abrir y modificar los archivos de un proyecto.
- Cambiar las propiedades de un proyecto.
- Cambiar las propiedades de los archivos de un proyecto.
- Cambiar las propiedades de los servidores Automation (bibliotecas de vínculo dinámico, `.dll` o archivos ejecutables `.exe`) generados desde el proyecto.
- Ejecutar código cuando se producen eventos en el proyecto.
- Volver a generar el proyecto o generar archivos `.app`, `.exe` o `.dll` a partir del mismo.

Con los nuevos enganches del Administrador de proyectos, los programadores avanzados pueden crear sus propios administradores de proyectos con interfaces de usuario únicas y personalizadas.

### Jerarquía de objetos de un proyecto

La jerarquía de objetos de un proyecto consta del proyecto, un objeto proyecto y su objeto ProjectHook asociado. El objeto Project contiene una colección de archivos, formada por los archivos del proyecto y una colección de servidores, formada por los servidores de Automatización creados desde el proyecto. En los diagramas siguientes se ilustra la jerarquía de objetos de un proyecto dentro del modelo de objetos de Visual FoxPro:



## Colección Proyectos

La colección Proyectos proporciona acceso directo al objeto proyecto y así permite manipular el proyecto, y también los archivos y servidores que contiene. A la colección Proyectos se le agrega un objeto Project siempre que se crea o se abre un proyecto, o cuando se genera un .app, .dll o .exe desde él.

Al igual que ocurre con otras colecciones OLE, es posible obtener información sobre un proyecto a partir de la colección Proyectos. Por ejemplo, el código siguiente utiliza las propiedades Count e Item de la colección Proyectos para mostrar los nombres de todos los proyectos de la colección y a continuación utiliza el comando FOR EACH para mostrar la misma información:

```

nProjectCount = Application.Projects.Count

FOR nCount = 1 TO nProjectCount
    ? Application.Projectos.Item(nCount).Name
NEXT

FOR EACH oProj IN Application.Projectos
    ? oProj.Name
ENDFOR
  
```

Esta línea de código utiliza la propiedad ActiveProject para agregar un programa, Main.prg, al proyecto activo actualmente:

```
Application.ActiveProject.Archivos.Add('Main.prg')
```

Esta línea de código agrega Main.prg al primer proyecto agregado a la colección Proyectos:

```
Application.Proyectos[1].Archivos.Add('Main.prg')
```

La colección Proyectos tiene la propiedad y el método siguientes:

### Propiedades

---



---

[Count](#)

---



---

### Métodos

---



---

[Item](#)

---



---

### Objeto Project

El objeto Project se instancia siempre que se abre un proyecto desde el menú Archivo o con los comandos CREATE PROJECT, MODIFY PROJECT, BUILD APP, BUILD DLL, BUILD EXE o BUILD PROJECT. El objeto Project permite manipular el proyecto mediante programa y es posible el acceso al mismo a través del [objeto Application](#) de Visual FoxPro. Observe que el objeto Application cuenta con la nueva propiedad [ActiveProject](#), que proporciona una referencia de objeto al proyecto abierto en el Administrador de proyectos activo actualmente.

El objeto Project tiene las propiedades y métodos siguientes:

### Propiedades

---



---

[Application](#)

[AutoIncrement](#)

---



---

[BaseClass](#)

[BuildDateTime](#)

---



---

[Debug](#)

[Encrypted](#)

---



---

[HomeDir](#)

[Icon](#)

---



---

[MainClass](#)

[MainFile](#)

---



---

[Name](#)

[Parent](#)

---



---

[ProjectHook](#)

[ProjectHookClass](#)

---



---

[ProjectHookLibrary](#)

[SCCProvider](#)

---



---

[ServerHelpFile](#)

[ServerProject](#)

---



---

[TypeLibCLSID](#)

[TypeLibDesc](#)

---



---

[TypeLibName](#)

[VersionComments](#)

---



---

[VersionCompany](#)

[VersionCopyright](#)

---



---

[VersionDescription](#)

[VersionLanguage](#)

---



---

[VersionNumber](#)

[VersionProduct](#)

---

---

[VersionTrademarks](#)

---

---

[Visible](#)

---

---

---

**Métodos**

---

[Build](#)

---

---

[CleanUp](#)

---

---

[Refresh](#)

---

---

[SetMain](#)

---

---

**Objeto ProjectHook**

Un objeto ProjectHook es una clase de base de Visual FoxPro que se instancia de forma predeterminada siempre que se abre un proyecto asignado al objeto ProjectHook. (Puede incluir la cláusula NOPROJECTHOOK en CREATE PROJECT y MODIFY PROJECT para evitar que se instancie un objeto ProjectHook para el proyecto.)

El objeto ProjectHook permite el acceso mediante programación a los eventos que ocurren en un proyecto. Por ejemplo, puede ejecutar código siempre que agregue un archivo al proyecto.

Puede especificar una clase ProjectHook predeterminada para los nuevos proyectos en la [ficha Proyectos](#) del cuadro de diálogo Opciones. Si no se especifica una clase ProjectHook predeterminada en la ficha Proyectos, no se asignará ninguna a los proyectos nuevos. Puede especificar una clase ProjectHook para un proyecto individual (anulando la predeterminada) en el [cuadro de diálogo Información del proyecto](#). En tiempo de ejecución, puede utilizar la propiedad ProjectHook para especificar una clase ProjectHook para un proyecto. Si cambia la clase ProjectHook de un proyecto, la nueva no tendrá efecto hasta que cierre el proyecto y lo abra de nuevo.

El objeto ProjectHook tiene las propiedades, eventos y métodos siguientes:

**Propiedades**

---

[BaseClass](#)

---

---

[Class](#)

---

---

[ClassLibrary](#)

---

---

[Comment](#)

---

---

[Name](#)

---

---

[OLEDropEffects](#)

---

---

[OLEDropHasData](#)

---

---

[OLEDropMode](#)

---

---

[Parent](#)

---

---

[ParentClass](#)

---

---

[Tag](#)

---

---

**Eventos**

---

[AfterBuild](#)

---

---

[BeforeBuild](#)

---

---

[Destroy](#)

---

---

[Error](#)

---

---

[Init](#)

---

---

[OLEDragDrop](#)

---

---

---

[OLEDragOver](#)[OLEGiveFeedBack](#)

---

[QueryAddFile](#)[QueryModifyFile](#)

---

[QueryRemoveFile](#)[QueryRunFile](#)

---

---

### Métodos

---

[AddProperty](#)[ReadExpression](#)

---

[ReadMethod](#)[ResetToDefault](#)

---

[SaveAsClass](#)[WriteExpression](#)

---

### El objeto Project y su interacción con el objeto ProjectHook

Al abrir el Administrador de proyectos desde el menú Archivo o con los comandos [CREATE PROJECT](#) o [MODIFY PROJECT](#), aparecerá la ventana del Administrador de proyectos y se instanciará un objeto Project con su objeto ProjectHook asociado. Los comandos de generación de proyectos ([BUILD PROJECT](#), [BUILD APP](#), [BUILD DLL](#) y [BUILD EXE](#)) también instancian los objetos Project y ProjectHook.

Cuando se produce un evento en un proyecto, el objeto Project lo envía al objeto ProjectHook. Entonces se ejecuta el código de usuario del evento en el objeto ProjectHook y el control se envuelve al objeto Project. El valor devuelto al objeto Project desde el objeto ProjectHook determina si el objeto Project termina la operación. Si incluye NODEFAULT en el código del evento, evitará que se realice la acción predeterminada. Por ejemplo, al agregar NODEFAULT al evento QueryAddFile se impide que se agregue un archivo con éxito al proyecto.

### Colección Archivos

La colección Archivos proporciona acceso directo al objeto archivo, lo que permite manipular los objetos archivo de un proyecto mientras éste está abierto. Al igual que ocurre con otras colecciones OLE, es posible obtener información sobre un archivo de un proyecto a partir de la colección Archivos. Por ejemplo, en el código siguiente se utilizan las propiedades Count e Item de la colección Archivos para mostrar los nombres de todos los archivos de la colección y, a continuación, se utiliza el comando FOR EACH para mostrar la misma información:

```
nFileCount = Application.ActiveProject.Archivos.Count

FOR nCount = 1 TO nFileCount
    ? Application.ActiveProject.Archivos.Item(nCount).Name
NEXT

FOR EACH oProj IN Application.ActiveProject.Archivos
    ? oProj.Name
ENDFOR
```

Esta línea de código utiliza la propiedad ActiveProject para agregar un archivo, Main.prg, al proyecto activo actualmente:

```
Application.ActiveProject.Archivos.Add('Main.prg')
```

Esta línea de código agrega Main.prg al primer proyecto agregado a la colección Proyectos:

```
Application.Proyectos[1].Archivos.Add('Main.prg')
```

La colección Archivos tiene las propiedades y métodos siguientes:

### Propiedades

---



---

[Count](#)

---



---

### Métodos

---



---

[Add](#)

---



---

[Item](#)

---



---

### Objeto archivo

El objeto archivo permite manipular los archivos individuales de un proyecto.

El objeto archivo tiene las propiedades y métodos siguientes:

### Propiedades

---



---

[CodePage](#)

[Description](#)

---



---

[Exclude](#)

[FileClass](#)

---



---

[FileClassLibrary](#)

[LastModified](#)

---



---

[Name](#)

[ReadOnly](#)

---



---

[SCCStatus](#)

[Type](#)

---



---

### Métodos

---



---

[AddToSCC](#)

[CheckIn](#)

---



---

[CheckOut](#)

[GetLatestVersion](#)

---



---

[Modify](#)

[Remove](#)

---



---

[RemoveFromSCC](#)

[Run](#)

---



---

[UndoCheckOut](#)

---



---

### Colección Servidores

La colección Servidores ofrece acceso directo al objeto servidor, lo que permite manipular los servidores que contiene un proyecto. A la colección Servidores se le agrega un objeto servidor siempre que se genera una biblioteca de vínculos dinámicos (.dll) o un archivo ejecutable (.exe) desde el proyecto. Si desea más información sobre la creación de servidores de Automatización, vea [Crear servidores de Automatización](#) en el Capítulo 16, “Agregar OLE”, del *Manual del programador*.

La colección Servidores tiene la propiedad y el método siguientes:

### Propiedades

---



---

[Count](#)

---



---

### Métodos

---



---

[Item](#)

---



---

### Objeto servidor

El objeto servidor permite determinar información (incluida información de biblioteca de tipos) sobre los servidores de Automatización contenidos en un proyecto. Esta información también está disponible en la [ficha Servidores](#) del cuadro de diálogo Información del proyecto. Observe que el objeto servidor no se crea hasta que se genera el proyecto que contiene la clase OLEPUBLIC (especificada en el comando [DEFINE CLASS](#)).

El objeto servidor tiene las propiedades y métodos siguientes:

### Propiedades

---



---

| <a href="#">CLSID</a>              | <a href="#">Description</a> |
|------------------------------------|-----------------------------|
| <a href="#">HelpContextID</a>      | <a href="#">Instancing</a>  |
| <a href="#">ProgID</a>             | <a href="#">ServerClass</a> |
| <a href="#">ServerClassLibrary</a> |                             |

---



---

### Arquitectura del objeto Project

El objeto Project de Visual FoxPro expone una interfaz IDispatch para que los clientes de Automatización, los controles ActiveX y otros objetos COM tengan acceso a él a través de interfaces OLE estándar. Al exponer el objeto Project una interfaz IDispatch, los errores que puedan generarse al manipular proyectos serán errores OLE.

### Mejoras del lenguaje

Existen dos nuevas cláusulas para los comandos CREATE PROJECT y MODIFY PROJECT. La primera, NOPROJECTHOOK, evita que se instancie el objeto ProjectHook de un proyecto. La

segunda, NOSHOW, abre el proyecto sin mostrarlo en el Administrador de proyectos, lo que permite manipularlo mediante programación sin mostrarlo. Puede utilizar la propiedad Visible para mostrar posteriormente el Administrador de proyectos. Si desea más información sobre estas nuevas cláusulas, vea [CREATE PROJECT](#) y [MODIFY PROJECT](#).

## Eventos de proyecto

En las secciones siguientes se describen los eventos que ocurren cuando se crean proyectos, se modifican, cierran, generan, etcétera; y también el orden en que se producen.

### Crear un nuevo proyecto

Los eventos siguientes se producen al ejecutar CREATE PROJECT, al crear un nuevo proyecto desde el menú **Archivo** o al hacer clic en el botón **Nuevo** de la barra de herramientas y especificar la creación de un proyecto nuevo:

1. Se crea el objeto Project.
2. Se instancia el objeto ProjectHook.
3. Se produce el evento Init del objeto ProjectHook. Si el evento Init devuelve "verdadero" (.T.), que es el valor predeterminado, el proyecto se crea y se muestra en el Administrador de proyectos.

Si el evento Init devuelve "falso" (.F.), el proyecto no se crea, se liberan los objetos Project y ProjectHook y no se muestra el Administrador de proyectos.

### Modificar un proyecto existente

Los eventos siguientes se producen al ejecutar MODIFY PROJECT, al modificar un proyecto existente desde el menú **Archivo** o al hacer clic en el botón **Abrir** de la barra de herramientas y especificar un proyecto nuevo o ya existente:

1. Se crea el objeto Project. El objeto Project obtiene sus valores del archivo .pjx del proyecto.
2. Se instancia el objeto ProjectHook.
3. Se produce el evento Init del objeto ProjectHook. Si el evento Init devuelve "verdadero" (.T.) (el valor predeterminado), el proyecto se abre para su modificación en el Administrador de proyectos.

Si el evento Init devuelve "falso" (.F.), el proyecto no se abre para su modificación, se liberan los objetos Project y ProjectHook, y no se muestra el Administrador de proyectos.

### Cerrar un proyecto

Los eventos siguientes se producen al cerrar un proyecto abierto:

1. Se produce el evento Destroy del objeto ProjectHook y se libera el objeto ProjectHook.

2. Se libera el objeto Project.

### **Ejecutar BUILD APP, BUILD DLL o BUILD EXE**

Los eventos siguientes se producen cuando se ejecuta BUILD APP, BUILD DLL o BUILD EXE:

1. Se crea el objeto Project. El objeto Project obtiene sus valores del archivo .pjx del proyecto.
2. Se instancia el objeto ProjectHook.
3. Se produce el evento Init para el objeto ProjectHook. Si el evento Init devuelve "verdadero" (.T.), que es el valor predeterminado, se produce el evento BeforeBuild del objeto ProjectHook. Si se ha especificado NODEFAULT en el evento BeforeBuild, el archivo .app, .dll o .exe no se genera. En caso contrario, el proceso de generación continúa.

Si se agrega algún archivo al proyecto durante el proceso de generación, se producirá el evento QueryAddFile del objeto ProjectHook antes de agregar efectivamente el archivo. Si se ha especificado NODEFAULT en el evento QueryAddFile, no se agregará el archivo al proyecto. En caso contrario, el archivo se agrega. Cuando se ha generado con éxito el archivo .app, .dll o .exe, se produce el evento AfterBuild del objeto ProjectHook y, a continuación, su evento Destroy.

Si el evento Init devuelve "falso" (.F.), el archivo .app, .dll o .exe se genera, y se liberan los objetos Project y ProjectHook.

### **Ejecutar BUILD PROJECT**

Los eventos siguientes se producen al ejecutar BUILD PROJECT con la cláusula FROM. Si se omite la cláusula FROM, los eventos se producen en el orden descrito anteriormente para la ejecución de BUILD APP, BUILD DLL o BUILD EXE.

1. Se crea el objeto Project. El objeto Project obtiene sus valores del archivo .pjx del proyecto.
2. Se instancia el objeto ProjectHook.
3. Se produce el evento Init del objeto ProjectHook. Si el evento Init devuelve "verdadero" (.T.), que es el valor predeterminado, los archivos especificados en la cláusula FROM se agregan individualmente al proyecto. El evento QueryAddFile del objeto ProjectHook se produce antes de agregar cada uno de los archivos. Si se ha especificado NODEFAULT en el evento QueryAddFile, los archivos no se agregarán al proyecto. En caso contrario, sí se agregan.

A continuación se produce el evento BeforeBuild del objeto ProjectHook. Si se ha especificado NODEFAULT en el evento BeforeBuild, el proyecto no se generará. En caso contrario, el proyecto se genera. Cuando termina la generación, se produce el evento AfterBuild del objeto ProjectHook y, a continuación, tiene lugar su evento Destroy.

Si el evento Init del objeto ProjectHook devuelve "falso" (.F.), el proyecto no se genera. En tal caso se liberan los objetos Project y ProjectHook, y no se crea un nuevo archivo .pjx.

## Realizar una operación arrastrar y colocar

Los eventos siguientes se producen al arrastrar un archivo o un conjunto de archivos sobre la sección de esquema (árbol) del Administrador de proyectos:

1. Cuando el puntero del *mouse* se sitúa sobre la sección de esquema del Administrador de proyectos, se produce el evento `OLEDragOver` del objeto `ProjectHook`, con el valor 0 (`DRAG_ENTER` en `Foxpro.h`) en el parámetro *nEstado*. A continuación se produce el evento `OLEDragOver` repetidamente con el valor 2 (`DRAG_OVER` en `Foxpro.h`) en el parámetro *nEstado*. Si el puntero del *mouse* sale de la sección de esquema del Administrador de proyectos, se produce el evento `OLEDragOver` con el valor 1 (`DRAG_LEAVE` en `Foxpro.h`) en el parámetro *nEstado*.
2. El evento `OLEDragDrop` del objeto `ProjectHook` se produce al soltar el botón del *mouse* cuando el puntero se encuentra sobre la sección de esquema del Administrador de proyectos. De forma predeterminada, Visual FoxPro agrega al proyecto todos los archivos colocados en el Administrador de proyectos. Antes de agregar cada archivo, se produce el evento `QueryAddFile` del objeto `ProjectHook`.

## Agregar un archivo con el botón Agregar

Los eventos siguientes se producen al agregar un archivo a un proyecto con el botón **Agregar** del Administrador de proyectos:

1. Aparece el cuadro de diálogo **Abrir**.
2. Si selecciona un archivo y elige **Aceptar**, se creará un objeto archivo para el archivo seleccionado.
3. Se produce el evento `QueryAddFile` del objeto `ProjectHook` y se pasa al evento el nombre del objeto archivo. Si se ha especificado `NODEFAULT` en el evento `QueryAddFile`, el archivo no se agrega. En caso contrario, el archivo se agrega al proyecto.

## Agregar un archivo con el botón Nuevo

Los eventos siguientes se producen al agregar un archivo nuevo a un proyecto con el botón **Nuevo** del Administrador de proyectos:

1. Aparece el diseñador o editor correspondiente al archivo.
2. Cuando se guarda el nuevo archivo, aparece el cuadro de diálogo **Guardar como**. Al hacer clic en **Guardar** se crea un objeto archivo para el nuevo archivo.
3. Se produce el evento `QueryAddFile` del objeto `ProjectHook` y se le pasa el nombre del objeto archivo. Si se ha incluido `NODEFAULT` en el evento `QueryAddFile`, el archivo no se agrega. En caso contrario, el archivo se agrega al proyecto.

## Modificar un archivo con el botón Modificar

Los eventos siguientes se producen al modificar un archivo de un proyecto con el botón **Modificar** del Administrador de proyectos:

1. Se produce el evento QueryModifyFile del objeto ProjectHook antes de que se muestre el diseñador o editor correspondiente al archivo.
2. Se pasa como parámetro al evento QueryModifyFile el objeto archivo del archivo a modificar. Si se ha incluido NODEFAULT en el evento QueryModifyFile, no se muestra el diseñador o editor correspondiente al archivo y éste no se modifica. En caso contrario, el archivo se abre con el diseñador o editor correspondiente para su modificación.

### Quitar un archivo con el botón Quitar

Los eventos siguientes se producen al quitar un archivo de un proyecto con el botón **Quitar** del Administrador de proyectos:

1. Se produce el evento QueryRemoveFile del objeto ProjectHook.
2. El objeto archivo correspondiente al archivo que se va a quitar se pasa como parámetro al evento QueryRemoveFile. Si se ha especificado NODEFAULT en el evento QueryRemoveFile, el archivo no se quita. En caso contrario, el archivo se quita del proyecto.

### Ejecutar un archivo con el botón Ejecutar

Los eventos siguientes se producen al ejecutar un archivo de un proyecto con el botón **Ejecutar** del Administrador de proyectos:

1. Se produce el evento QueryRunFile del objeto ProjectHook.
2. El objeto archivo correspondiente al archivo que se va a ejecutar se pasa como parámetro al evento QueryRunFile. Si se ha especificado NODEFAULT en el evento QueryRunFile, el archivo no se ejecuta. En caso contrario, se ejecuta.

### Volver a generar un proyecto o generar un archivo con el botón Generar

Los eventos siguientes se producen al volver a generar un proyecto, o al generar un archivo .app, .dll o .exe desde un proyecto con el botón **Generar** del Administrador de proyectos:

1. Aparece el cuadro de diálogo **Opciones de generación**.
2. Puede elegir **Volver a generar el proyecto**, **Generar aplicación**, **Generar ejecutable** o **Generar DLL COM** y luego especificar opciones de generación adicionales. Si hace clic en **Cancelar**, la generación no se produce.
3. El evento BeforeBuild del objeto ProjectHook se produce al elegir **Aceptar**, y entonces comienza el proceso de generación.
4. Al completarse la generación, se produce el evento AfterBuild del objeto ProjectHook.

## Ejemplo de enlaces del Administrador de proyectos

La aplicación de ejemplo Solutions de Visual FoxPro incluye un ejemplo llamado “Seguir actividades de un proyecto” que ilustra muchos de los nuevos enlaces del Administrador de proyectos.

### Para ejecutar la aplicación de ejemplo Solutions

- Escriba lo siguiente en la ventana **Comandos**:

```
DO (HOME(2) + 'solution\solution')
```

– O bien –

1. En el menú **Programa**, elija **Ejecutar**.
2. Elija la carpeta ...\**Samples\Vfp98\Solution**.
3. Haga doble clic en **Solution.app**.

### Para ejecutar el ejemplo “Seguir actividades de un proyecto”

1. Una vez iniciado Solution.app, haga doble clic en **Nuevas características de Visual FoxPro 6.0**.
2. Haga clic en **Seguir actividades de un proyecto** y luego en el botón **Ejecutar ejemplo**.

El ejemplo “Seguir actividades de un proyecto” permite abrir un proyecto para después manipularlo de diversas formas. Los cambios realizados en el proyecto se almacenan en una tabla. Al cerrar el proyecto, podrá ver los cambios realizados en él en una ventana Examinar.

Si desea más información sobre el funcionamiento del ejemplo “Seguir actividades de un proyecto” y desea observar más de cerca el código que tiene detrás, puede abrir el formulario utilizado para crearlo.

### Para abrir el formulario “Seguir actividades de un proyecto”

1. Una vez iniciado Solution.app, haga doble clic en **Nuevas características de Visual FoxPro 6.0**.
2. Haga clic en **Seguir actividades de un proyecto** y luego en el botón **Ver código**.

Acttrack.scx, el formulario empleado para crear el ejemplo “Seguir actividades de un proyecto”, se abrirá en el Diseñador de formularios.

También puede estimar conveniente observar más de cerca la biblioteca de clases de ProjectHook, Project\_hook.vcx, que está asignada al proyecto abierto en el ejemplo “Seguir actividades de un proyecto”. La mayoría del código que se ejecuta cuando se producen eventos en el proyecto se encuentra en los procedimientos de evento de esta biblioteca de clases. Project\_hook.vcx se encuentra

en el directorio ...\\Samples\Vfp98\Solution\Tahoe.

## Asistentes y generadores nuevos y mejorados

Los asistentes y generadores siguientes son nuevos o se han mejorados.

### [Asistente para aplicaciones](#) *Nuevo*

El Asistente para aplicaciones de Visual FoxPro 6.0 admite el [Marco de trabajo de aplicación](#) mejorado y el nuevo [Generador de aplicaciones](#). Puede ejecutar el Asistente para aplicaciones desde la [Galería de componentes](#) o desde el menú **Herramientas** de Visual FoxPro, al hacer clic en **Asistentes** y luego en **Aplicación**.

**Nota** El [Asistente para aplicaciones \(5.0\)](#) de Visual FoxPro 5.0 está disponible en el [cuadro de diálogo Selección de asistente](#) por motivos de compatibilidad con versiones anteriores.

### [Asistentes para conexión](#) *Nuevos*

Los Asistentes para conexión incluyen el **Asistente para generación de código** y el **Asistente para ingeniería inversa**. Estos asistentes permiten administrar fácilmente las transferencias entre las bibliotecas de clases de Visual FoxPro y los modelos de Microsoft Visual Modeler.

### [Asistente para bases de datos](#) *Nuevo*

El Asistente para bases de datos de Visual FoxPro utiliza plantillas para crear bases de datos y tablas. También puede usar este asistente para crear índices y relaciones entre las tablas de una nueva base de datos.

### [Asistente para documentación](#) *Mejorado*

El Asistente para documentación de Visual FoxPro ofrece ahora una versión para utilizar los [Analizadores de código](#) mientras se crea la documentación.

### [Asistente para formularios](#) *Mejorado*

El Asistente mejorado para formularios de Visual FoxPro proporciona máscaras de entrada, formatos y una clase de asignación de campos para campos específicos almacenados en una base de datos. Este asistente incluye también más opciones de estilo de formularios, como por ejemplo, si son desplazables.

### [Asistente para gráficos](#) *Mejorado*

El Asistente para gráficos de Visual FoxPro crea un gráfico a partir de una tabla de Visual FoxPro mediante Microsoft Graph. Este asistente actualizado admite el componente Graph 8.0 de Microsoft Office 97, que incluye la automatización de la hoja de datos y la serie por la opción fila/columna.

### [Asistente para importar](#) *Mejorado*

El Asistente mejorado para importar de Visual FoxPro admite Office 97 y el tratamiento de múltiples hojas de Microsoft Excel y proporciona la opción de importar una tabla en una base de datos.

### [Asistente para etiquetas](#) *Mejorado*

El Asistente para etiquetas de Visual FoxPro permite ahora un mayor control de las fuentes de las etiquetas y proporciona acceso directo al Asistente para agregar etiquetas.

### [Asistente para combinar correspondencia](#) *Mejorado*

El Asistente para Combinar correspondencia de Visual FoxPro crea un origen de datos con un documento combinado de Microsoft Word o con un archivo de texto que puede abrir cualquier procesador de texto. Este asistente admite el componente Microsoft Word 8.0 de Office 97, permite una auténtica automatización VBA con el [objeto Application](#) y utilizar colecciones.

### [Asistente para tablas dinámicas](#) *Mejorado*

El Asistente para tablas dinámicas de Visual FoxPro ayuda a crear tablas de hoja de cálculo interactivas que resumen y analizan los datos de dos o más campos. Este asistente actualizado admite el componente Microsoft Excel 8.0 de Office 97. Puede elegir guardar una tabla dinámica directamente en Excel o agregar una tabla dinámica como un objeto a un formulario.

### [Asistente para informes](#) *Mejorado*

El Asistente para informes de Visual FoxPro incluye ahora funcionalidad avanzada de agrupamiento y resumen que permite personalizar más fácilmente los informes dentro del propio asistente. También incorpora más estilos de informes entre los que elegir.

### [Asistente para vistas remotas](#) *Mejorado*

El Asistente para vistas remotas de Visual FoxPro proporciona acceso a las tablas de sistema, lo que le permite utilizar la funcionalidad de los controladores ODBC que las admitan.

### [Asistente para ejemplos](#) *Nuevo*

El Asistente para ejemplos de Visual FoxPro muestra en pasos sencillos cómo crear su propio asistente. El resultado es un archivo HTML creado a partir de los registros del origen de datos especificado.

### [Asistente para instalación](#) *Mejorado*

El Asistente para instalación de Visual FoxPro permite un mejor uso de los controles ActiveX y también anular el límite en el número de archivos que puede utilizar Windows para las instalaciones en NT. También permite agregar .DLL externas a la aplicación a través de la instalación y crear instalaciones basadas en Web.

### [Asistente para tablas](#) *Mejorado*

El Asistente para tablas de Visual FoxPro ofrece ahora nuevas plantillas de tablas, configuraciones de estilo opcionales, uso de los tipos de datos binarios Character y Memo y acceso a las bases de datos. Puede agregar una tabla a una base de datos, y también utilizar la configuración de la base de datos para determinar los formatos de los campos que se van a agregar a la tabla. También puede establecer relaciones entre las tablas de la base de datos.

### [Asistente para publicación en Web](#) *Nuevo*

El Asistente para publicación en Web de Visual FoxPro genera un archivo HTML creado a partir de los registros del origen de datos que especifique.

## Marco de trabajo de aplicación mejorado

El Marco de trabajo de aplicación de Visual FoxPro 6.0 está diseñado para hacer más fácil la programación de aplicaciones de Visual FoxPro. Para tener acceso al mismo puede utilizar el [Asistente para aplicaciones](#) o el elemento **Nueva aplicación** de la [Galería de componentes](#). Este marco de trabajo mejorado admite el marco disponible en Visual FoxPro 5.0, que incluye:

- Un archivo de proyecto (.pjx).
- Un archivo de programa principal (Main.prg) para la configuración global y de entorno, que inicia la pantalla de inicio u otras llamadas específicas y que inicia los formularios Tutorial.
- Un menú principal.
- El [objeto Marco de trabajo de Visual FoxPro](#) para ejecutar el menú principal, las barras de herramientas de los formularios y la administración de informes, el control de errores y la administración de las sesiones de datos.

El marco de trabajo de Visual FoxPro 6 utiliza un objeto Application mejorado, y proporciona los elementos adicionales siguientes:

- Archivo de inclusión principal que contiene el valor APP\_GLOBAL para facilitar su localización y su uso por parte de componentes con configuraciones y cadenas.
- Archivo de configuración opcional (Config.fpw) para determinados tipos de aplicaciones.
- Uso de la [clase ProjectHook](#) para el control de los eventos relacionados con el proyecto.
- Metatabla de aplicación para mantener la información que utilizan la clase ProjectHook y los generadores de aplicaciones para crear formularios en el nuevo proyecto.
- Uso del [Generador de aplicaciones](#) para facilitar la adición de componentes al proyecto.

### Iniciar el Generador de aplicaciones

Puede iniciar el [Generador de aplicaciones](#) desde el menú **Herramientas** de Visual FoxPro o desde la

## Galería de componentes.

### Para iniciar el Generador de aplicaciones en el menú Herramientas

1. Haga clic en **Asistentes** y, a continuación, haga clic en **Todos**.
2. Haga clic en **Generador de aplicaciones** en el cuadro de diálogo **Selección de Asistente**.

### Para iniciar el Generador de aplicaciones desde la Galería de componentes

- Haga doble clic en el elemento **Nueva aplicación**.

Al elegir **Aceptar** se cerrará el generador y se aplicará la configuración de las propiedades de todas las fichas.

También puede iniciar el Generador de aplicaciones con el botón secundario del *mouse* en la ventana Administrador de proyectos, pero al hacerlo de este modo el Generador de aplicaciones sólo crea metatablas para la aplicación y sólo verá tres fichas en él. La única forma de obtener el marco de trabajo de aplicación mejorado completo es a través del [Asistente para aplicaciones](#) o con el elemento Nueva aplicación de la [Galería de componentes](#).

Para obtener detalles sobre el contenido y el uso del marco de trabajo mejorado y el Generador de aplicaciones, vea [Programar aplicaciones con el marco de trabajo de aplicaciones](#) en la Ayuda.

## Archivos

### Archivo de inclusión principal

Este archivo #INCLUDE común lo utilizan los componentes con configuraciones y cadenas. También incluye el valor APP\_GLOBAL, el nombre que utilizan los componentes como referencia.

### Archivo de configuración

Archivo Config.fpw opcional utilizado en aplicaciones tales como formularios de alto nivel, para implementar configuraciones tales como SCREEN=OFF.

### Clase ProjectHook

Controla los eventos relacionados con el proyecto, tales como agregar nuevos archivos. También tiene acceso al Generador de aplicaciones para establecer acciones y propiedades de la interacción de los archivos con la aplicación.

### Metatabla de aplicación

Contiene información tal como la configuración de proyecto, establecida o utilizada por el Generador de aplicaciones y los objetos ProjectHook.

### Generador de aplicaciones

Facilita la adición de componentes al proyecto y el establecimiento de propiedades tales como las opciones desplazamiento.

Un marco de trabajo de aplicación incluye el archivo de proyecto como biblioteca de clases inicial, una subclase de las clases de base de Visual FoxPro lista para rellenarse con tablas y documentos nuevos o ya existentes.

El marco de trabajo le permite especificar si desea crear una aplicación completa o sólo un marco de trabajo de aplicación. Si elige crear una aplicación completa, puede incluir en ella una base de datos y formularios o informes ya creados. O bien puede crear una nueva aplicación desde cero con una plantilla de base de datos. Si elige crear un marco de trabajo, podrá volver atrás posteriormente y agregarle componentes.

## Crear un marco de trabajo

Puede crear un marco de trabajo de aplicación con el [Asistente para aplicaciones](#) o con el elemento Nueva aplicación de la [Galería de componentes](#). Al utilizar la Galería de componentes, se agrega un nuevo elemento carpeta de proyecto a la carpeta Favoritos.

Independientemente del método que utilice, Visual FoxPro mostrará un [Generador de aplicaciones](#) en el que podrá agregar la información a almacenar en una metatabla.

### Para crear una aplicación

1. En el menú **Herramientas**, haga clic en **Asistentes**, y luego haga clic en **Aplicaciones**.  
- o bien -
1. En la carpeta Catálogos de la **Galería de componentes**, haga doble clic en el elemento **Nueva aplicación**.
2. En el cuadro de diálogo **Escriba el nombre del proyecto**,
  - Especifique el nombre del proyecto.
  - Acepte o busque el archivo de proyecto.
  - Elija las opciones para **Crear estructura de directorios del proyecto** (predeterminado) y **Agregar al catálogo Favoritos** (predeterminado)

Para obtener detalles sobre el contenido y el uso del marco de trabajo mejorado, y el Generador de aplicaciones, vea [Programar aplicaciones con el marco de trabajo de aplicaciones](#) en la Ayuda.

Puede utilizar también la [Galería de componentes](#) para agregar formularios, informes, datos y objetos de servicio al marco de trabajo de la nueva aplicación, y para agregar controles a los formularios.

Al utilizar la Galería de componentes para agregar un formulario a una aplicación, puede crear un nuevo formulario o crear una subclase a partir de una clase existente.

## Capítulo 33: Mejoras para la programación

Microsoft Visual FoxPro incluye ahora nuevas características de programación para aumentar la productividad del programador. Entre dichas características se incluyen los métodos Access y Assign (que permiten ejecutar código cuando se consulta o se modifica el valor de una propiedad), la compatibilidad con más formatos de archivos gráficos y nuevos elementos del lenguaje para simplificar la programación. Además, se han incluido en Visual FoxPro muchas de las funciones de manipulación de nombres de archivo disponibles en Foxtools.fll, una biblioteca API de Visual FoxPro.

En este capítulo se trata:

- [Métodos Access y Assign](#)
- [Compatibilidad con gráficos GIF y JPEG](#)
- [Elementos del lenguaje nuevos y mejorados](#)
- [Compatibilidad con el milenio](#)

### Métodos Access y Assign

Se ha mejorado Visual FoxPro para que admita los métodos Access y Assign. Estos métodos definidos por el usuario permiten ejecutar código cuando se consulta el valor de una propiedad o cuando se intenta modificar el valor de una propiedad.

El código del método Access se ejecuta cuando se consulta el valor de una propiedad, normalmente con la propiedad en una referencia de objeto, al almacenar el valor de la propiedad en una variable o al mostrar el valor de la propiedad con un signo de interrogación (?).

El código del método Assign se ejecuta cuando se intenta modificar el valor de una propiedad, normalmente mediante los comandos STORE o = para asignar un nuevo valor a la propiedad.

Los métodos Access y Assign sólo se ejecutan cuando se consultan o modifican los valores de las propiedades en tiempo de ejecución. La consulta o modificación de los valores de las propiedades en tiempo de diseño no hace que se ejecuten los métodos Access y Assign.

**Nota** Como el valor que intenta asignar a la propiedad se pasa al método Assign, debe incluir una instrucción PARAMETERS o LPARAMETERS en el método Assign para aceptar el valor.

Puede crear independientemente los métodos Access y Assign (puede crear un método Access sin un método Assign o un método Assign sin un método Access).

Puede crear métodos Access y Assign para propiedades creadas mediante programación en una instrucción DEFINE CLASS o de forma interactiva para un formulario o una clase con el Diseñador de formularios y el Diseñador de clases.

**Nota** También se pueden crear métodos Access y Assign para todas las propiedades nativas de Visual FoxPro. Por ejemplo, puede crear un método Access para la propiedad Left de un formulario,

lo que le permitirá ejecutar código siempre que se consulte la propiedad Left del formulario. Puede crear un método Assign para una propiedad nativa de sólo lectura de Visual FoxPro (por ejemplo, la propiedad ParentClass), pero el método nunca se ejecutará.

## Ventajas de los métodos Access y Assign

Los métodos Access y Assign proporcionan las ventajas siguientes:

- Puede crear una interfaz pública para una clase o un objeto que separe la interfaz de la implementación.
- Puede implementar fácilmente la validación de las propiedades.
- Puede proteger fácilmente las propiedades en controles ActiveX que derivan de clases.

## Crear métodos Access y Assign

Las mejoras del comando DEFINE CLASS y de los Diseñadores de formularios y de clases le permiten crear métodos Access y Assign mediante programación y de forma interactiva.

### Nuevos sufijos para DEFINE CLASS

Se han agregado dos sufijos, `_ACCESS` y `_ASSIGN`, al comando DEFINE CLASS para crear métodos Access y Assign. Si anexa una de estas palabras clave al nombre de una función o un procedimiento, se creará un método Access o Assign para una propiedad que tenga el mismo nombre que la función o el procedimiento.

Por ejemplo, el siguiente ejemplo de código utiliza DEFINE CLASS para crear una clase personalizada llamada MiClase. Se crea una propiedad definida por el usuario, MiPropiedad, para la clase. A continuación se crea un método Access para MiPropiedad con la instrucción PROCEDURE.

Cuando se consulta el valor de la propiedad, se ejecuta el código del procedimiento (WAIT WINDOW 'Éste es el método Access'). También se crea un método Assign para MiPropiedad, de nuevo con una instrucción PROCEDURE. Cuando se intente modificar el valor de la propiedad, se ejecutará el código del procedimiento (WAIT WINDOW 'Éste es el método Assign').

Observe el uso de la instrucción LPARAMETERS para aceptar el valor pasado al método Assign. Este ejemplo también muestra cómo puede crear propiedades de sólo lectura.

```
DEFINE CLASS MiClase AS Custom
    MiPropiedad = 100 && Propiedad definida por el usuario

    PROCEDURE MiPropiedad_ACCESS && Método Access
        WAIT WINDOW 'Éste es el método Access';
        + ' ' + PROGRAM( )
        RETURN THIS.MiPropiedad
    ENDPROC

    PROCEDURE MiPropiedad_ASSIGN && Método Assign
        LPARAMETERS tAssign && Necesario para aceptar el valor
        WAIT WINDOW 'Éste es el método Assign';
        + ' ' + PROGRAM( )
    ENDPROC
ENDDDEFINE
```

El ejemplo siguiente muestra cómo puede agregar un método Assign a una propiedad nativa de Visual FoxPro y realizar una sencilla validación del valor de la propiedad que intenta establecer. Observe que en este ejemplo se crea un método Assign sin un método Access correspondiente.

Se usa DEFINE CLASS para crear una clase Form llamada frmMiForm. Se crea un método Assign llamado Left\_ASSIGN con una instrucción PROCEDURE. El código del método Assign se ejecuta siempre que se intente asignar un valor a la propiedad Left del formulario.

Si intenta asignar un valor negativo a la propiedad Left, se muestra un mensaje y no se modifica el valor de la propiedad Left. Si intenta asignar un valor no negativo a la propiedad Left, la propiedad Left del formulario queda establecida a dicho valor.

```
DEFINE CLASS frmMiForm AS Form

    PROCEDURE Left_ASSIGN && Método Assign
        LPARAMETERS tAssign && Necesario para aceptar el valor

        DO CASE
            CASE tAssign < 0 && valor de Left negativo
                WAIT WINDOW 'El valor tiene que ser mayor que 0'
            OTHERWISE && valor de Left no negativo
                THIS.Left = tAssign
        ENDCASE
    ENDPROC
ENDEDEFINE
```

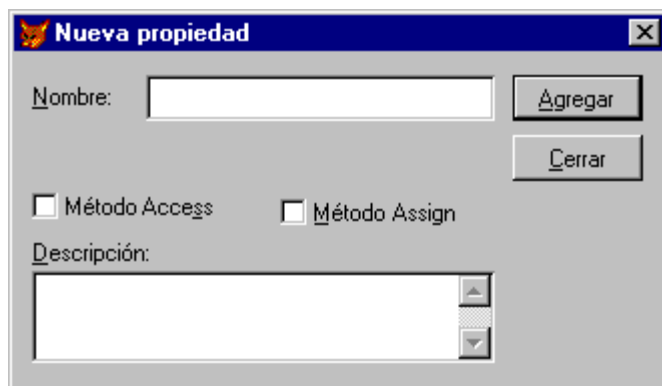
Para obtener más información sobre la sintaxis utilizada para crear métodos Access y Assign, vea [DEFINE CLASS](#).

## Los Diseñadores de clases y de formularios

### Para crear un método Access o Assign en el Diseñador de formularios

1. Elija **Nueva propiedad** en el menú **Formulario**.

Se muestra el cuadro de diálogo **Nueva propiedad**.



2. Escriba el nombre de la propiedad que va a crear en el cuadro de texto **Nombre** y, a continuación, seleccione la casilla de verificación **Método Access** o la casilla de verificación

**Método Assign** (o ambas).

3. Elija **Agregar** para crear la propiedad del formulario y para crear los métodos Access y Assign de la propiedad.

### **Para crear un método Access o Assign para una propiedad intrínseca de Visual FoxPro en el Diseñador de formularios**

1. Elija **Nuevo método** en el menú **Formulario**.

Se muestra el cuadro de diálogo **Nuevo método**.

2. Escriba el nombre de la propiedad intrínseca seguido de `_Access` o `_Assign` en el cuadro de texto **Nombre**. Por ejemplo, para crear un método Access para la propiedad `Left`, escriba `Left_Access` en el cuadro de texto **Nombre**.
3. Elija **Agregar** para crear métodos Access o Assign para la propiedad intrínseca.

**Nota** En el Diseñador de formularios sólo puede crear propiedades con métodos Access y Assign para un formulario o un conjunto de formularios. Para crear propiedades con métodos Access y Assign para un control o un objeto, utilice el Diseñador de clases para crear la clase de control o de objeto. En el Diseñador de clases, agregue propiedades con métodos Access y Assign al control o al objeto y después agregue la clase de control u objeto al formulario en el Diseñador de formularios.

### **Para crear un método Access o Assign para una clase en el Diseñador de clases**

1. Elija **Nueva propiedad** en el menú **Clase**.

Se muestra el cuadro de diálogo **Nueva propiedad**.

2. Escriba el nombre de la propiedad que va a crear en el cuadro de texto **Nombre** y, a continuación, seleccione la casilla de verificación **Método Access** o la casilla de verificación **Método Assign** (o ambas).
3. Elija **Agregar** para crear una propiedad para la clase y crear los métodos Access o Assign de la propiedad.

Para obtener más información acerca de la creación de métodos Access y Assign, vea el tema [Nueva propiedad \(Nueva propiedad\)](#).

### **Método THIS\_ACCESS**

Se ha agregado a Visual FoxPro 6.0 un nuevo método global de clase, `THIS_ACCESS`. El código de un método `THIS_ACCESS` se ejecuta siempre que se intente modificar el valor de un miembro de un objeto o siempre que se intente consultar un miembro de un objeto.

El método `THIS_ACCESS` se crea en el código en un comando `DEFINE CLASS` o en los cuadros de diálogo **Nuevo método** o **Modificar propiedades** de las bibliotecas de clases visuales `.vcx`. Un método `THIS_ACCESS` siempre debe devolver una referencia de objeto; si no es así, se generará un error.

Normalmente se devuelve la referencia de objeto THIS. El método THIS\_ACCESS también debe incluir un parámetro para aceptar el nombre del miembro del objeto que se modifica o consulta.

El siguiente ejemplo muestra cómo crear un método THIS\_ACCESS en el código de un comando DEFINE CLASS. Cuando este ejemplo se ejecuta como programa, 'Caption' se muestra dos veces, la primera cuando se le asigna un valor a la propiedad Caption y la segunda cuando se consulta el valor de la propiedad Caption. Después se muestra el valor de la propiedad Caption ('abc').

```
CLEAR
oTempObj = CREATEOBJECT('MiForm')  && Crea una instancia del formulario
oTempObj.Caption = 'abc'  && Asigna un valor y desencadena THIS_ACCESS
? oTempObj.Caption  && Consulta un valor y desencadena THIS_ACCESS

DEFINE CLASS MiForm AS Form
  PROCEDURE THIS_ACCESS
    LPARAMETER cMemberName  && Nombre del miembro del objeto

    IF cMemberName = 'caption'
      ? cMemberName  && Muestra el nombre del miembro del objeto
    ENDIF
    RETURN THIS
  ENDPROC
ENDDDEFINE
```

Observe que THIS\_ACCESS no pretende ser un sustituto global de los métodos Access y Assign (sólo proporciona información acerca del miembro del objeto al que se tiene acceso o se consulta). A diferencia de los métodos Access y Assign, THIS\_ACCESS no proporciona control sobre los valores devueltos a miembros de objeto específicos.

## Notas de programación de Access y Assign

Las secciones siguientes describen la información de programación para métodos Access y Assign.

### Alcance

Los métodos Access y Assign están protegidos de forma predeterminada (no puede tener acceso a un método Access o Assign ni modificarlo desde fuera de la clase en la que se cree el método Access o Assign).

Incluya la palabra clave HIDDEN cuando cree un método Access o Assign para impedir el acceso y las modificaciones a las propiedades desde fuera de la definición de clase. Sólo los métodos y los eventos de la definición de la clase pueden tener acceso a las propiedades ocultas. Mientras que las subclases de la definición de clase pueden tener acceso a las propiedades protegidas, sólo la definición de la clase puede tener acceso a las propiedades ocultas.

**Nota** Si no incluye la palabra clave HIDDEN, puede crear subclases con los métodos Access y Assign.

### Depuración

Puede ver el código de los métodos Access y Assign en la ventana Seguimiento de la ventana Depurador. Sin embargo, los métodos Access y Assign no se pueden ejecutar en las ventanas

Inspección y Locales de la ventana Depurador.

### **Pasar matrices a los métodos Assign**

Se pasan matrices a los métodos Access y Assign de la misma forma en que se pasan a procedimientos estándar de Visual FoxPro.

Si ejecuta SET UDFPARMS TO REFERENCE o se antepone @ al nombre de la matriz, se pasa la matriz completa al método Access o Assign. Si ejecuta SET UDFPARMS TO VALUE o escribe el nombre de la matriz entre paréntesis, se pasa por valor el primer elemento de la matriz. Los elementos de matriz siempre se pasan por valor. Para obtener más información acerca de cómo pasar valores y matrices, vea [SET UDFPARMS](#).

### **Controles ActiveX**

Las propiedades, los eventos o los métodos nativos de los controles ActiveX no admiten las propiedades Access y Assign. Sin embargo, las propiedades, los eventos y los métodos del Contenedor OLE que contiene al control ActiveX sí admite los métodos Access y Assign.

### **Método ResetToDefault**

Si ejecuta el método ResetToDefault para un método Access o Assign se modifica el código del método Access o Assign al miniprograma predeterminado. El resultado es que el código heredado del método, si lo hubiera, no se ejecuta. La técnica utilizada para asegurar que el código heredado de la clase primaria se ejecuta varía según el tipo de método.

Coloque el código siguiente en la subclase de un método Access para ejecutar el código en la clase primaria:

```
RETURN DODEFAULT( )
```

Coloque el código siguiente en la subclase de un método Access para ejecutar el código de la clase primaria:

```
LPARAMETERS vnewval  
DODEFAULT(vnewval)  
THIS.<propiedad> = vnewval
```

Coloque el código siguiente en la subclase de un método THIS\_ACCESS para ejecutar el código de la clase primaria:

```
LPARAMETERS cmember  
RETURN DODEFAULT(cmember)
```

## **Compatibilidad con gráficos GIF y JPEG**

Se ha mejorado Visual FoxPro para que admita los formatos de archivos gráficos GIF (Graphics Interchange Format) y JPEG (Joint Photographic Electronic Group), ampliamente utilizados en Internet.

En general, las áreas que aceptaban el formato .bmp (mapa de bits) en versiones anteriores de Visual FoxPro aceptan ahora los formatos de archivos gráficos.

| Formato gráfico                     | Extensión de archivo |
|-------------------------------------|----------------------|
| Mapa de bits                        | .bmp                 |
| Device Independent Bitmap           | .dib                 |
| Graphics Interchange Format         | .gif                 |
| Joint Photographic Electronic Group | .jpg                 |
| Cursor                              | .cur                 |
| Cursor animado                      | .ani                 |
| Icono                               | .ico                 |

**Nota** En Visual FoxPro se pueden usar los archivos de cursores, cursores animados e iconos como archivos gráficos. Por ejemplo, puede especificar un cursor animado para la propiedad Picture del control Imagen (sin embargo, el control Imagen muestra la representación estática del cursor).

Visual FoxPro proporciona la compatibilidad con gráficos en tres áreas: lenguaje, controles y objetos, y la interfaz.

## El lenguaje de Visual FoxPro

Se han mejorado los siguientes comandos y funciones para admitir los nuevos formatos de archivos gráficos.

### GETPICT()

Se ha mejorado el cuadro de diálogo **Abrir imagen** que se muestra al ejecutar la [función GETPICT\(\)](#) en la ventana **Comandos**, lo que le permitirá encontrar rápidamente todos los archivos gráficos admitidos por Visual FoxPro. Ahora el cuadro de lista desplegable **Archivos de tipo** incluye los siguientes elementos.

| Elemento                    | Especificaciones de archivo                     |
|-----------------------------|---|
| Todos los archivos          | *.*   |
| Todos los archivos gráficos | *.bmp, *.dib, *.jpg, *.gif, *.ani, *.cur, *.ico |
| Mapa de bits                | *.bmp, *.dib                                    |
| Cursor                      | *.cur   |
| Cursor animado              | *.ani   |

|       |       |
|-------|-------|
| Icono | *.ico |
| JPEG  | *.jpg |
| GIF   | *.gif |

Active la casilla de verificación **Vista previa** para mostrar el archivo gráfico seleccionado actualmente. En versiones anteriores de Visual FoxPro era necesario elegir el botón **Vista previa** cada vez que se seleccionaba un nuevo archivo gráfico. También se ha aumentado el tamaño del área **Imagen**.

## CLEAR RESOURCES

Ahora el [comando CLEAR RESOURCES](#) de Visual FoxPro borra todos los archivos gráficos almacenados localmente, como los archivos .gif y .jpg.

## Controles y objetos de Visual FoxPro

La tabla siguiente muestra los controles y objetos de Visual FoxPro que tienen propiedades para las que puede especificar archivos gráficos. Ahora puede especificar archivos gráficos de tipo .gif, .jpg, cursores, cursores animados e iconos para estas propiedades, además de los archivos gráficos de tipo .bmp y .dib admitidos en las versiones anteriores de Visual FoxPro.

| Control u objeto                    | Propiedades                               |
|-------------------------------------|---|
| Control CheckBox                    | DisabledPicture<br>DownPicture<br>Picture |
| Contenedor de objetos CommandButton | DisabledPicture<br>DownPicture<br>Picture |
| Objeto Container                    | Picture                                   |
| Objeto Control                      | Picture                                   |
| Objeto Custom                       | Picture                                   |
| Objeto Form                         | Picture                                   |
| Control Image                       | Picture                                   |
| Control OptionButton                | DisabledPicture<br>DownPicture<br>Picture |
| Objeto Page                         | Picture                                   |
| Objeto _Screen                      | Picture                                   |

## La interfaz de Visual FoxPro

Varios de los diseñadores de Visual FoxPro le permiten especificar archivos gráficos con el cuadro de diálogo **Abrir**. Los cuadros de diálogo **Abrir** de los siguientes diseñadores se han mejorado para incluir los nuevos formatos de archivos gráficos.

### Diseñador de formularios y Diseñador de clases

En la ventana **Propiedades**, si hace doble clic en la propiedad o elige el botón del cuadro de diálogo de la propiedad, puede mostrar el cuadro de diálogo **Abrir** para las propiedades que admiten archivos gráficos.

### Administrador de proyectos

Puede agregar archivos gráficos a un proyecto desde las fichas **Todos** y **Otros** del Administrador de proyectos. Cuando está seleccionada la ficha **Todos** o la ficha **Otros**, seleccione el elemento **Otros archivos** y, a continuación, elija **Agregar**. Se muestra el cuadro de diálogo **Abrir**, que le permitirá agregar un archivo de gráficos al proyecto.

### Diseñador de informes

La barra de herramientas **Controles de informes** contiene el botón **Imagen/Control ActiveX dependiente**. Haga clic en dicho botón y arrastre el cursor sobre una banda del Diseñador de informes para mostrar el cuadro de diálogo **Imagen para informe**. Para mostrar el cuadro de diálogo **Abrir**, elija el botón **Archivo**.

## Elementos del lenguaje nuevos y mejorados

Se han agregado muchos elementos nuevos al lenguaje de Visual FoxPro y se han mejorado otros. Los nuevos elementos del lenguaje mostrados en esta sección incluyen [documentos activos](#), [enganches del Administrador de proyectos](#), [arrastrar y colocar de OLE](#), [mejoras en el servidor](#) y [otros elementos nuevos](#).

También se muestran los elementos [mejorados](#).

Además, se han agregado a Visual FoxPro muchas de las funciones de manipulación de nombres de archivo disponibles en Foxttools.fll, una biblioteca API de Visual FoxPro. Ya no es necesario utilizar SET LIBRARY TO FOXTOOLS.FLL para llamar a estas funciones de [Foxttools](#); puede llamarlas directamente desde sus programas de Visual FoxPro.

En esta sección también se describen las mejoras de rendimiento, solidez y facilidad de uso de Visual FoxPro.

| <b>Nuevos elementos del lenguaje para documentos activos</b> | <b>Descripción</b>  |
|--|---|
| <a href="#">Objeto ActiveDoc</a>                             | Crea un documento activo que se puede alojar en un contenedor de documentos activos como Microsoft Internet Explorer.   |
| <a href="#">Propiedad AlwaysOnBottom</a>                     | Evita que una ventana de formulario cubra otras ventanas.   |
| <a href="#">Evento CommandTargetExec</a>                     | Se produce cuando el usuario hace clic en un elemento de menú o en un elemento de una barra de herramientas que pertenece al contenedor de documento activo.  |
| <a href="#">Evento CommandTargetQuery</a>                    | Se produce cuando el contenedor de documentos activos tiene que saber si el documento activo admite varios comandos de menú o de barra de herramientas del contenedor de forma que pueda habilitar o deshabilitar los correspondientes elementos de menú o botones de la barra de herramientas. |
| <a href="#">Evento ContainerRelease</a>                      | Se produce cuando un contenedor libera un documento activo.   |
| <a href="#">Propiedad ContainerReleaseType</a>               | Especifica si se abre un documento activo en el entorno de tiempo de ejecución de Visual FoxPro cuando lo libera su contenedor.   |
| <a href="#">Propiedad ContinuousScroll</a>                   | Especifica si el desplazamiento en un formulario es continuo o si el desplazamiento sólo tiene lugar cuando se libera un cuadro de desplazamiento.  |
| <a href="#">Comando DEFINE PAD</a>                           | Admite nuevas opciones NEGOTIATE para especificar la ubicación del título del menú en documentos activos.   |
| <a href="#">Función GETHOST()</a>                            | Devuelve una referencia de objeto al contenedor de un documento activo.   |
| <a href="#">Método GoBack</a>                                | Explora hacia atrás en la lista del historial de un contenedor de documentos activos.   |
| <a href="#">Método GoFoward</a>                              | Explora hacia adelante en la lista del historial de un contenedor de documentos activos.  |
| <a href="#">Evento HideDoc</a>                               | Se produce cuando se explora desde un documento activo.   |
| <a href="#">Propiedad HscrollSmallChange</a>                 | Especifica la cantidad que se desplaza un formulario en la dirección horizontal cuando hace clic en una flecha de desplazamiento horizontal.  |
| <a href="#">Objeto Hyperlink</a>                             | Crea un objeto Hyperlink.   |
| <a href="#">Función ISHOSTED()</a>                           | Devuelve un valor de tipo Logical que indica si un  |

|  |   |
|--|---|
|  | documento activo está alojado en un contenedor de documentos activos.   |
| <a href="#">Método NavigateTo</a>  | Explora en un contenedor de documentos activos a una ubicación especificada.  |
| <a href="#">Evento Run</a>   | Se produce cuando un documento activo termina de coordinarse con su contenedor y con COM y está listo para ejecutar el código de usuario.   |
| <a href="#">Variable del sistema RUNACTIVEDOC</a>                                  | Especifica la aplicación que inicia un documento activo.  |
| <a href="#">Propiedad ScrollBars</a>   | Disponible ahora para formularios. Si un formulario está en un documento activo, las barras de desplazamiento se muestran automáticamente cuando el tamaño del contenedor de documentos activos sea menor que el tamaño del formulario. |
| <a href="#">Evento Scrolled</a>  | Disponible ahora para formularios. Le permite determinar si se hace clic en las barras de desplazamiento horizontal o vertical o si se mueve una barra de desplazamiento.   |
| <a href="#">Método SetViewport</a>   | Establece los valores de las propiedades ViewPortLeft y ViewPortTop de un formulario.   |
| <a href="#">Evento ShowDoc</a>   | Se produce al explorar hasta un documento activo.   |
| <a href="#">SYS(4204) – Depuración de documentos activos</a>                       | Activa o desactiva el soporte para la depuración de documentos activos en el depurador de Visual FoxPro.  |
| <a href="#">Propiedad ViewPortHeight</a>   | Contiene el alto de la vista de un formulario.  |
| <a href="#">Propiedad ViewPortLeft</a>   | Contiene la coordenada izquierda del formulario visible en la vista.  |
| <a href="#">Propiedad ViewPortTop</a>  | Contiene la coordenada superior del formulario visible en la vista.   |
| <a href="#">Propiedad ViewPortWidth</a>  | Contiene el ancho de la vista de un formulario.   |
| <a href="#">Propiedad VscrollSmallChange</a>                                       | Especifica la cantidad de desplazamiento vertical de un formulario cuando hace clic en una flecha de desplazamiento.  |
| <b>Nuevos elementos del lenguaje para enganches del Administrador de proyectos</b> | <b>Descripción</b>  |
| <a href="#">Propiedad ActiveProject</a>  | Contiene una referencia de objeto al objeto Project de la ventana del Administrador de proyectos.   |
| <a href="#">Método Add</a>   | Agrega un archivo a un proyecto.  |

|   |  |
|---|--|
| <a href="#">Método AddToSCC</a>         | Agrega un archivo de un proyecto al control de código fuente.  |
| <a href="#">Evento AfterBuild</a>       | Se produce después de que se vuelva a generar un proyecto o de que se cree en un proyecto un archivo de aplicación (.app), una biblioteca de vínculos dinámicos (.dll) o un archivo ejecutable (.exe). |
| <a href="#">Propiedad AutoIncrement</a> | Especifica si la versión generada de un proyecto se incrementa automáticamente cada vez que se genere un archivo .exe o .dll en proceso distribuible.  |
| <a href="#">Evento BeforeBuild</a>      | Se produce antes de que se vuelva a generar un proyecto o de que se cree en un proyecto un archivo de aplicación (.app), una biblioteca de vínculos dinámicos (.dll) o un archivo ejecutable (.exe).   |
| <a href="#">Método Build</a>            | Vuelve a generar un proyecto o crea un archivo de aplicación (.app), una biblioteca de vínculos dinámicos (.dll) o un archivo ejecutable (.exe) a partir de un proyecto.                               |
| <a href="#">Propiedad BuildDateTime</a> | Contiene la fecha y la hora en que se generó por última vez un proyecto.   |
| <a href="#">Método CheckIn</a>          | Protege las modificaciones realizadas a un archivo de proyecto bajo control de código fuente.  |
| <a href="#">Método CheckOut</a>         | Desprotege un archivo que está bajo control de código fuente y le permite hacer modificaciones en el archivo.  |
| <a href="#">Método CleanUp</a>          | Limpia una tabla del proyecto; para ello, quita los registros marcados para eliminación y empaqueta los campos memo.   |
| <a href="#">Método Close</a>            | Cierra un proyecto y libera el objeto ProjectHook del proyecto y los demás objetos del proyecto.   |
| <a href="#">Propiedad CLSID</a>         | Contiene el CLSID registrado (Identificador de clase) para un servidor de un proyecto.   |
| <a href="#">Propiedad CodePage</a>      | Contiene la página de códigos de un archivo de un proyecto.  |
| <a href="#">Propiedad Count</a>         | La cuenta del número de objetos proyecto, archivo o servidor de una colección de proyectos, archivos o servidores.   |
| <a href="#">Comando CREATE PROJECT</a>  | Mejorado en Visual FoxPro 6.0. Admite dos nuevas opciones, NOSHOW y NOPROJECTHOOK, para utilizarlas con los nuevos enganches del Administrador de proyectos.   |
| <a href="#">Propiedad Debug</a>         | Especifica si la información de depuración se incluye en el  |

|  |   |
|--|---|
|  | código fuente compilado de un proyecto.   |
| <a href="#">Propiedad Description</a>      | En un objeto archivo es la descripción del archivo. En un objeto servidor es la descripción de la clase del servidor.   |
| <a href="#">Propiedad Encrypted</a>        | Especifica si se va a cifrar el código fuente compilado de un proyecto.   |
| <a href="#">Propiedad Exclude</a>          | Especifica si se va a excluir un archivo de una aplicación (.app), una biblioteca de vínculos dinámicos (.dll) o un archivo ejecutable (.exe) cuando se genere a partir de un proyecto. |
| <a href="#">Propiedad FileClass</a>        | Contiene el nombre de la clase de formulario en la que se basa un formulario de un proyecto.  |
| <a href="#">Propiedad FileClassLibrary</a> | Contiene el nombre de la biblioteca de clases que contiene la clase en la que se basa un formulario de un proyecto.   |
| <a href="#">Objeto File</a>                | Proporciona referencias a archivos específicos de un proyecto.  |
| <a href="#">Colección Files</a>            | Una colección de objetos archivo.   |
| <a href="#">Método GetLatestVersion</a>    | Obtiene la versión más reciente de un archivo de un proyecto que está bajo control de código fuente y copia una versión de sólo lectura en su unidad local.                             |
| <a href="#">Propiedad HomeDir</a>          | Especifica el directorio de inicio de un proyecto.  |
| <a href="#">Propiedad Instancing</a>       | Especifica cómo se puede crear una instancia de un servidor de un proyecto.   |
| <a href="#">Método Item</a>                | Devuelve una referencia de objeto al elemento especificado en una colección de proyectos.   |
| <a href="#">Propiedad LastModified</a>     | Contiene la fecha y la hora de la última modificación realizada a un archivo de un proyecto.  |
| <a href="#">Propiedad MainClass</a>        | Contiene el nombre de una clase ActiveDoc establecida como programa principal de un proyecto.   |
| <a href="#">Propiedad MainFile</a>         | Contiene el nombre y la ruta del archivo establecido como programa principal de un proyecto.  |
| <a href="#">Método Modify</a>              | Abre un archivo de un proyecto para su modificación en el editor o diseñador apropiado.   |
| <a href="#">Comando MODIFY PROJECT</a>     | Mejorado en Visual FoxPro 6.0. Admite dos nuevas opciones, NOSHOW y NOPROJECTHOOK, para su utilización con los nuevos enganches del Administrador de proyectos.                         |
| <a href="#">Propiedad ProgID</a>           | Contiene el PROGID registrado (Identificador programático) de un servidor de un proyecto.   |

|  |   |
|--|---|
| <a href="#">Objeto Project</a>               | Se crea una instancia cuando se crea o se abre un proyecto.   |
| <a href="#">Objeto ProjectHook</a>           | Se crea una instancia cuando se abre un proyecto y proporciona acceso mediante programación a eventos de proyecto.                    |
| <a href="#">Propiedad ProjectHook</a>        | Una referencia de objeto al objeto ProjectHook que se crea para un proyecto.  |
| <a href="#">Propiedad ProjectHookClass</a>   | La clase ProjectHook predeterminada de un proyecto.   |
| <a href="#">Propiedad ProjectHookLibrary</a> | La biblioteca de clases visuales .vcx que contiene la clase ProjectHook predeterminada de un proyecto.                                |
| <a href="#">Colección Projects</a>           | Una colección de objetos proyecto.  |
| <a href="#">Evento QueryAddFile</a>          | Se produce justo antes de que se agregue un archivo a un proyecto.  |
| <a href="#">Evento QueryModifyFile</a>       | Se produce justo antes de que se modifique un archivo en un proyecto.   |
| <a href="#">Evento QueryRemoveFile</a>       | Se produce justo antes de que se elimine un archivo de un proyecto.   |
| <a href="#">Evento QueryRunFile</a>          | Se produce justo antes de que se ejecute un archivo o de que se realice una vista previa de un informe o una etiqueta en un proyecto. |
| <a href="#">Método Remove</a>                | Quita un archivo de su colección de archivos y del proyecto.  |
| <a href="#">Método RemoveFromSCC</a>         | Elimina un archivo del proyecto del control de código fuente.   |
| <a href="#">Método Run</a>                   | Ejecuta o muestra una vista previa de un archivo de un proyecto.  |
| <a href="#">Propiedad SCCProvider</a>        | El nombre del proveedor de control de código fuente para un proyecto.   |
| <a href="#">Propiedad SCCStatus</a>          | Contiene un valor numérico que indica el estado de control de código fuente de un archivo de un proyecto.                             |
| <a href="#">Objeto Server</a>                | Una referencia de objeto a un servidor del proyecto.  |
| <a href="#">Colección Servers</a>            | Una colección de objetos servidor.  |
| <a href="#">Propiedad ServerClass</a>        | Contiene el nombre de una clase servidor de un proyecto.  |
| <a href="#">Propiedad ServerClassLibrary</a> | Contiene el nombre de la biblioteca de clases o de programa que contiene una clase de servidor.                                       |
| <a href="#">Propiedad ServerHelpFile</a>     | El archivo de Ayuda de la biblioteca de tipos creada para las clases de servidor de un proyecto.                                      |

|  |   |
|--|---|
| <a href="#">Propiedad ServerProject</a>      | El nombre del proyecto que contiene las clases de servidor.   |
| <a href="#">Método SetMain</a>               | Establece el archivo principal de un proyecto.  |
| <a href="#">Propiedad Type</a>               | El tipo de archivo de un archivo del proyecto.  |
| <a href="#">Propiedad TypeLibCLSID</a>       | El CLSID de registro (Identificador de clase) para una biblioteca de tipos creada para las clases de servidor de un proyecto. |
| <a href="#">Propiedad TypeLibDesc</a>        | La descripción de una biblioteca de tipos creada para las clases de servidor de un proyecto.                                  |
| <a href="#">Propiedad TypeLibName</a>        | El nombre de una biblioteca de tipos creada para las clases de servidor de un proyecto.                                       |
| <a href="#">Método UndoCheckOut</a>          | Descarta las modificaciones realizadas en un archivo y vuelve a protegerlo.   |
| <a href="#">Propiedad VersionComments</a>    | Los comentarios del proyecto.   |
| <a href="#">Propiedad VersionCompany</a>     | El nombre de la compañía del proyecto.  |
| <a href="#">Propiedad VersionCopyright</a>   | La información de copyright del proyecto.   |
| <a href="#">Propiedad VersionDescription</a> | La descripción del proyecto.  |
| <a href="#">Propiedad VersionLanguage</a>    | La información de idioma del proyecto.  |
| <a href="#">Propiedad VersionNumber</a>      | El número de versión del proyecto.  |
| <a href="#">Propiedad VersionProduct</a>     | El nombre del producto del proyecto.  |
| <a href="#">Propiedad VersionTrademarks</a>  | La información de marcas registradas del proyecto.  |

### **Nuevos elementos del lenguaje para arrastrar y colocar de OLE Descripción**

|  |   |
|--|---|
| <a href="#">Método ClearData</a>       | Borra todos los datos y los formatos de datos del objeto DataObject con arrastrar y colocar de OLE.                         |
| <a href="#">Objeto DataObject</a>      | Contenedor para los datos que se transfieren desde un origen de arrastre OLE a un destino para colocar OLE.                 |
| <a href="#">Método GetData</a>         | Obtiene datos desde el objeto DataObject con arrastrar y colocar de OLE.  |
| <a href="#">Método GetFormat</a>       | Determina si los datos de un formato especificado están disponibles en el objeto DataObject con arrastrar y colocar de OLE. |
| <a href="#">Evento OLECompleteDrag</a> | Se produce cuando se colocan datos en el destino o cuando   |

|   |   |
|---|---|
|   | se cancela la operación arrastrar y colocar de OLE.   |
| <a href="#">Método OLEDrag</a>                                | Inicia una operación arrastrar y colocar de OLE.  |
| <a href="#">Evento OLEDragDrop</a>                            | Se produce cuando se colocan datos en el destino y la propiedad OLEDropMode del destino es 1 - Activado.  |
| <a href="#">Propiedad OLEDragMode</a>                         | Especifica cómo se inicia una operación de arrastre.  |
| <a href="#">Evento OLEDragOver</a>                            | Se produce cuando se arrastran datos a su destino y la propiedad OLEDropMode del destino es 1 – Activado.   |
| <a href="#">Propiedad OLEDragPicture</a>                      | Especifica la imagen que se muestra bajo el puntero del <i>mouse</i> durante una operación arrastrar y colocar de OLE. Puede especificar un archivo de imagen del tipo .bmp, .dib, .jpg, .gif, .ani, .cur e .ico. |
| <a href="#">Propiedad OLEDropEffects</a>                      | Especifica el tipo de operaciones arrastrar y colocar de OLE que admite un destino.   |
| <a href="#">Propiedad OLEDropHasData</a>                      | Especifica cómo se controla una operación colocar.  |
| <a href="#">Propiedad OLEDropMode</a>                         | Especifica cómo controla un destino de colocar las operaciones colocar de OLE.  |
| <a href="#">Propiedad OLEDropTextInsertion</a>                | Especifica si puede colocar texto en medio de una palabra en la parte de cuadro de texto de un control.   |
| <a href="#">Evento OLEGiveFeedBack</a>                        | Se produce después de cada evento OLEDragOver. Permite que el origen de arrastre especifique el tipo de operación arrastrar y colocar de OLE y la información visual.   |
| <a href="#">Evento OLESetData</a>                             | Se produce en un origen de arrastre cuando un destino para colocar llama al método GetData y no hay datos con el formato especificado en el objeto DataObject con arrastrar y colocar de OLE.                     |
| <a href="#">Evento OLEStartDrag</a>                           | Se produce cuando se llama al método OLEDrag.   |
| <a href="#">Método SetData</a>                                | Coloca datos en el objeto DataObject con arrastrar y colocar de OLE.  |
| <a href="#">Método SetFormat</a>                              | Coloca un formato de datos en el objeto DataObject con arrastrar y colocar de OLE.  |
| <b>Nuevos elementos del lenguaje para mejoras de servidor</b> | <b>Descripción</b>  |
| <a href="#">Función COMARRAY()</a>                            | Especifica cómo se pasan matrices a objetos COM.  |
| <a href="#">Función COMCLASSINFO()</a>                        | Devuelve información de registro acerca de un objeto  |

|  |   |
|--|---|
|  | COM como un servidor de automatización de Visual FoxPro.  |
| <a href="#">Función COMRETURNERROR()</a>                                     | Llena la estructura de excepciones de COM con información que los clientes de COM pueden utilizar para determinar el origen de los errores de Automatización.   |
| <a href="#">Función CREATEOBJECTEX()</a>                                     | Crea una instancia de un objeto COM registrado (como un servidor de automatización de Visual FoxPro) en un equipo remoto.   |
| <a href="#">Propiedad ServerName</a>   | Contiene la ruta completa y el nombre de archivo de un servidor de automatización.  |
| <a href="#">Propiedad StartMode</a>  | Contiene un valor numérico que indica cómo se inició la instancia de Visual FoxPro.   |
| <a href="#">SYS(2335) – Modo de servidor desatendido</a>                     | Activa o desactiva los estados modales en los servidores de automatización .exe distribuibles de Visual FoxPro.   |
| <a href="#">SYS(2334) – Modo de invocación de servidor de automatización</a> | Devuelve un valor que indica cómo se invocó el servidor de automatización de Visual FoxPro o si se está ejecutando una aplicación independiente (.exe).   |
| <b>Otros elementos nuevos del lenguaje</b>                                   | <b>Descripción</b>  |
| <a href="#">Método AddProperty</a>   | Agrega una nueva propiedad a un objeto.   |
| <a href="#">Función AGETFILEVERSION()</a>                                    | Crea una matriz que contiene información acerca de archivos con recursos de versión de Microsoft Windows como los archivos .exe, .dll y .fl, o de servidores de automatización creados en Visual FoxPro.<br>Corresponde a la función GetFileVersion( ) de Foxtools. |
| <a href="#">Función AGETCLASS()</a>  | Muestra bibliotecas de clases en el cuadro de diálogo Abrir y crea una matriz que contiene el nombre de la biblioteca de clases y la clase elegidas.  |
| <a href="#">Función ALINES()</a>   | Copia cada línea de una expresión de cadena de caracteres o un campo memo en una fila correspondiente de una matriz.  |
| <a href="#">Función AMOUSEOBJ()</a>  | Devuelve los datos de posición del puntero del <i>mouse</i> y referencias de objeto para el objeto y el contenedor del objeto sobre los que se encuentra el puntero del <i>mouse</i> .  |
| <a href="#">Función ANETRESOURCES()</a>                                      | Coloca los nombres de los recursos compartidos de red e impresoras en una matriz y después devuelve el número de recursos.  |
| <a href="#">Función AVCXCLASSES()</a>  | Coloca la información acerca de las clases de una biblioteca de clases en una matriz.   |

|  |   |
|--|---|
| <a href="#">Propiedad DisplayCount</a>                               | Especifica el número de elementos mostrados en la parte de lista de un control ComboBox.  |
| <a href="#">Función FILETOSTR()</a>                                  | Devuelve el contenido de un archivo como una cadena de caracteres.  |
| <a href="#">Variable del sistema _GALLERY</a>                        | Especifica el programa que se ejecuta cuando elige Galería de componentes en el menú Herramientas.  |
| <a href="#">Variable del sistema _GENHTML</a>                        | Especifica un programa de generación de HTML (Lenguaje de marcado de hipertexto) que crea un archivo de texto que contiene la versión HTML de un formulario, menú, informe o tabla. |
| <a href="#">Variable del sistema _GETEXPR</a>                        | Especifica el programa que se ejecuta cuando se ejecute el comando GETEXPR o cuando se muestre el cuadro de diálogo Generador de expresiones.                                       |
| <a href="#">Método GridHitTest</a>                                   | Devuelve, como parámetros de resultados, los componentes de un control cuadrícula correspondientes a las coordenadas horizontal (X) y vertical (Y) especificadas.                   |
| <a href="#">Variable del sistema _INCLUDE</a>                        | Especifica un archivo de encabezado predeterminado incluido en las clases, formularios o conjuntos de formularios definidos por el usuario.   |
| <a href="#">Función INDEXSEEK()</a>                                  | Sin mover el puntero de los registros, busca en una tabla indexada la primera ocurrencia de un registro cuya clave de índice coincida con una expresión especificada.               |
| <a href="#">Función NEWOBJECT()</a>                                  | Crea una nueva clase u objeto directamente desde una biblioteca de clases visuales .vcx o desde un programa.  |
| <a href="#">Método NewObject</a>                                     | Agrega una nueva clase u objeto a un objeto directamente desde una biblioteca de clases visuales .vcx o desde un programa.  |
| <a href="#">Variable del sistema _SAMPLES</a>                        | Contiene la ruta del directorio en el que se instalan los ejemplos de Visual FoxPro.  |
| <a href="#">Comando SET BROWSEIME</a>                                | Especifica si se abre el Editor de métodos de entrada cuando se llega a un cuadro de texto en una ventana Examinar.   |
| <a href="#">Comando SET STRICTDATE</a>                               | Especifica si constantes ambiguas de tipo Date y DateTime generan errores.  |
| <a href="#">Función STRTOFILE()</a>                                  | Escribe el contenido de una cadena de caracteres en un archivo.   |
| <a href="#">SYS(3055) – Complejidad de las cláusulas FOR y WHERE</a> | Establece el nivel de complejidad de las cláusulas FOR y WHERE en los comandos y las funciones que las admiten.   |

|  |  |
|--|--|
| <a href="#">SYS(3056) – Leer la configuración del Registro</a> | Hace que Visual FoxPro vuelva a leer su configuración y la actualice con la configuración del Registro del sistema.  |
| <a href="#">Propiedad TitleBar</a>                             | Especifica si se muestra una barra de título en la parte superior de un formulario.  |
| <a href="#">Función VARTYPE()</a>                              | Devuelve el tipo de datos de una expresión.  |
| <b>Elementos del lenguaje mejorados</b>                        | <b>Descripción</b>   |
| <a href="#">Operador =</a>                                     | Se puede utilizar en Visual FoxPro 6.0 para determinar si dos referencias de objeto hacen referencia al mismo objeto.  |
| <a href="#">Comando ALTER TABLE – SQL</a>                      | Acepta una nueva cláusula FOR para las cláusulas ADD PRIMARY KEY y ADD FOREIGN KEY. FOR le permite crear índices principales y externos filtrados.   |
| <a href="#">Comando APPEND FROM</a>                            | Admite una nueva opción XL8 para importar datos desde hojas de cálculo de Microsoft Excel 97, y una nueva opción CSV para importar datos desde archivos con valores separados por comas.                               |
| <a href="#">Propiedad Century</a>                              | Ahora el valor predeterminado es 1 – Activado. La parte de siglo de la fecha se muestra en un cuadro de texto para proporcionar compatibilidad con el milenio.   |
| <a href="#">Control CheckBox</a>                               | Ahora admite la propiedad <a href="#">ReadOnly</a> .   |
| <a href="#">Objeto Column</a>                                  | Ahora admite las propiedades <a href="#">Comment</a> y <a href="#">Tag</a> y el método <a href="#">SaveAsClass</a> .   |
| <a href="#">Comando COMPILE DATABASE</a>                       | Ahora COMPILE DATABASE empaqueta los campos memo en el archivo memo .dct de la base de datos para eliminar el espacio no utilizado del archivo de memos.   |
| <a href="#">Objeto Container</a>                               | Ahora admite la propiedad <a href="#">Tag</a> .  |
| <a href="#">Objeto Control</a>                                 | Ahora admite la propiedad <a href="#">Tag</a> .  |
| <a href="#">Comando COPY TO</a>                                | Admite una nueva opción CSV para exportar datos como archivo de valores separados por comas.   |
| <a href="#">Comando CREATE FORM</a>                            | Admite una nueva cláusula AS que le permite crear un nuevo formulario o conjunto de formularios a partir de un formulario o de un conjunto de formularios de una biblioteca de clases visuales .vcx.                   |
| <a href="#">Objeto Cursor</a>                                  | Ahora admite las propiedades <a href="#">Comment</a> y <a href="#">Tag</a> y los métodos <a href="#">ReadExpression</a> , <a href="#">ReadMethod</a> , <a href="#">SaveAsClass</a> y <a href="#">WriteExpression</a> . |
| <a href="#">Objeto Custom</a>                                  | Ahora admite la propiedad <a href="#">Tag</a> .  |

---

|  |  |
|--|--|
| <a href="#">Objeto DataEnvironment</a> | Ahora admite las propiedades <a href="#">Comment</a> y <a href="#">Tag</a> y los métodos <a href="#">ReadExpression</a> , <a href="#">ReadMethod</a> , <a href="#">SaveAsClass</a> y <a href="#">WriteExpression</a> . |
| <a href="#">Función DATE()</a>         | Ahora admite argumentos numéricos opcionales que le permiten crear valores de fechas compatibles con el milenio.   |
| <a href="#">Función DATETIME()</a>     | Ahora admite argumentos numéricos opcionales que le permiten crear valores de fechas compatibles con el milenio.   |
| <a href="#">Comando DEFINE CLASS</a>   | Admite nuevos métodos Access y Assign, que le permiten ejecutar código siempre que consulte una propiedad o que intente cambiar el valor de una propiedad.   |
| <a href="#">Función FDATE()</a>        | Ahora admite un argumento opcional que le permite determinar la hora de la última modificación de un archivo sin tener que utilizar funciones de manipulación de caracteres.   |
| <a href="#">Objeto Form</a>            | Ahora admite la propiedad <a href="#">Scrollbars</a> y el evento <a href="#">Scrolled</a> .  |
| <a href="#">Objeto FormSet</a>         | Ahora admite las propiedades <a href="#">Parent</a> y <a href="#">Tag</a> .  |
| <a href="#">Función GETDIR()</a>       | Se ha mejorado el cuadro de diálogo Seleccionar directorio para que pueda mostrar más información.   |
| <a href="#">Función GETFILE()</a>      | Admite una nueva opción <i>cTítuloBarraTítulo</i> que le permite especificar el título del cuadro de diálogo Abrir.  |
| <a href="#">Función GETFONT()</a>      | Le permite especificar la fuente, el tamaño y el estilo de la fuente seleccionados inicialmente cuando se muestra el cuadro de diálogo Fuente.   |
| <a href="#">Objeto Header</a>          | Ahora admite las propiedades <a href="#">Comment</a> y <a href="#">Tag</a> y el método <a href="#">SaveAsClass</a> .   |
| <a href="#">Función HOME()</a>         | Ahora le permite determinar los directorios de los ejemplos, las herramientas, los gráficos y los directorios comunes de Visual FoxPro y Visual Studio.  |
| <a href="#">Control Image</a>          | Ahora admite la propiedad <a href="#">ToolTipText</a> .  |
| <a href="#">Comando IMPORT</a>         | Admite una nueva opción XL8 para importar datos desde una hoja de cálculo de Microsoft Excel 97.   |
| <a href="#">Control Label</a>          | Ahora admite la propiedad <a href="#">ToolTipText</a> .  |
| <a href="#">Comando MODIFY MEMO</a>    | Ahora la sintaxis con colores en las ventanas de edición de campos memo está desactivada en las aplicaciones de tiempo de ejecución distribuidas.  |
| <a href="#">Función OS()</a>           | Ahora admite una opción que le permite determinar si el  |

---

|  |   |
|--|---|
|  | sistema operativo acepta DBCS (juegos de caracteres de dos bytes).  |
| <a href="#">Objeto Page</a>  | Ahora acepta la propiedad <a href="#">Tag</a> y el método <a href="#">SaveAsClass</a> .   |
| <a href="#">Control PageFrame</a>                                    | Ahora admite la propiedad <a href="#">Tag</a> .   |
| <a href="#">Función PEMSTATUS()</a>                                  | PEMSTATUS() admite una nueva opción 6 para <i>nAtributo</i> que le permite determinar si se ha heredado de otro objeto o clase una propiedad, evento o método.  |
| <a href="#">Función PROGRAM()</a>                                    | Ahora admite -1 como argumento y le permite determinar el nivel del programa actual.  |
| <a href="#">Método Refresh</a>                                       | Ahora le permite actualizar la presentación visual del Administrador de proyectos y admite un nuevo parámetro para actualizar el estado de control de código fuente de los archivos de un proyecto.   |
| <a href="#">Objeto Relation</a>                                      | Ahora admite las propiedades <a href="#">Comment</a> y <a href="#">Tag</a> , los eventos <a href="#">Destroy</a> , <a href="#">Error</a> e <a href="#">Init</a> y los métodos <a href="#">ReadExpression</a> , <a href="#">ReadMethod</a> y <a href="#">WriteExpression</a> . |
| <a href="#">Comando REPORT</a>                                       | Ahora admite una cláusula PREVIEW IN SCREEN, que le permite colocar una ventana de vista previa en la ventana principal de Visual FoxPro.   |
| <a href="#">Objeto Separator</a>                                     | Ahora admite las propiedades <a href="#">Comment</a> y <a href="#">Tag</a> y los métodos <a href="#">ReadExpression</a> , <a href="#">ReadMethod</a> , <a href="#">SaveAsClass</a> y <a href="#">WriteExpression</a> .  |
| <a href="#">SET BELL</a>   | Ya no es necesaria la duración del sonido.  |
| <a href="#">SET(PRINTER)</a>   | Acepta una nueva opción 3 que le permite determinar la impresora predeterminada actual de Visual FoxPro establecida en los cuadros de diálogo Impresora o Configurar impresión de Visual FoxPro.  |
| <a href="#">SET(BELL)</a>  | Ahora se puede utilizar para determinar el tipo de sonido que se va a reproducir.   |
| <a href="#">Función STRCONV()</a>                                    | Admite un nuevo argumento <i>nIdLocale</i> que le permite especificar el Id. de configuración regional que se va a utilizar en la conversión.   |
| <a href="#">SYS(2333) – Compatibilidad con interfaz ActiveX dual</a> | Ahora le permite determinar el valor actual y el valor de inicio predeterminado de la compatibilidad para interfaz ActiveX dual se ha cambiado de habilitada en Visual FoxPro 5.0 a deshabilitada en Visual FoxPro 6.0.   |
| <a href="#">Función TABLEUPDATE()</a>                                | Si al actualizar registros se produce un error distinto a un  |

simple error de confirmación, el primer elemento de la matriz de errores contendrá -1 y después podrá utilizar AERROR( ) para determinar por qué no se han podido confirmar las modificaciones.

---



---

[Objeto ToolBar](#)

Ahora admite la propiedad [Tag](#) y el método [Release](#).

---



---

[Función TRANSFORM\(\)](#)

El código de formato *cCódigosFormato* ahora es opcional. Se utiliza una transformación predeterminada si se omite el código de formato *cCódigosFormato*.

---



---

[Función VERSION\(\)](#)

Admite dos nuevas opciones *de nExpresión*, 4 y 5, para devolver el número de versión de Visual FoxPro en formatos que se puedan analizar con facilidad.

---



---

**Funciones de Foxtools**

**Descripción**

Se han agregado las siguientes funciones a Visual FoxPro 6.0 desde Foxtools; ahora se pueden utilizar sin ejecutar SET LIBRARY TO FOXTOOLS.

Tenga en cuenta que tiene que volver a compilar los programas, bibliotecas de clases, etiquetas o informes creados en versiones anteriores de Visual FoxPro si contenían alguna de estas funciones.

---



---

[Función ADDBS\(\)](#)

Agrega una barra invertida (si fuera necesario) a una expresión de ruta.

---



---

[Función AGETFILEVERSION\(\)](#)

Crea una matriz que contiene información acerca de archivos con recursos de versión de Windows como los archivos .exe, .dll y .fll, o de servidores de automatización creados en Visual FoxPro. Corresponde a la función GetFileVersion( ) de Foxtools.

---



---

[Función DEFAULTTEXT\(\)](#)

Devuelve un nombre de archivo con una nueva extensión si no la tuviera.

---



---

[Función DRIVETYPE\(\)](#)

Devuelve el tipo de la unidad especificada.

---



---

[Función FORCEEXT\(\)](#)

Devuelve una cadena con la extensión de archivo anterior reemplazada por una nueva extensión.

---



---

[Función FORCEPATH\(\)](#)

Devuelve un nombre de archivo con una nueva ruta en lugar de la ruta anterior.

---



---

[Función JUSTDRIVE\(\)](#)

Devuelve la letra de unidad de una ruta completa.

---



---

[Función JUSTEXT\(\)](#)

Devuelve la extensión de tres letras de una ruta completa.

---



---

[Función JUSTFNAME\(\)](#)

Devuelve la parte del nombre de archivo de una ruta completa.

---



---

[Función JUSTPATH\(\)](#)

Devuelve la parte de ruta de una ruta completa.

---



---

[Función JUSTSTEM\(\)](#)

Devuelve el nombre (el nombre del archivo sin la

---

extensión) de una ruta completa y un nombre de archivo.

---

## Mejoras en el rendimiento de Visual FoxPro

Se ha mejorado significativamente el rendimiento de la concatenación de cadenas en Visual FoxPro 6.0. La concatenación de cadenas suele utilizarse para crear páginas Web con código como el siguiente:

```
cMiCadena = cMiCadena + <etiquetas html>
cMiCadena = cMiCadena + <más etiquetas html>
cMiCadena = cMiCadena + <aún más etiquetas html>
```

Además, también se ha mejorado el rendimiento de la creación de objetos y la creación de instancias; normalmente es 10 o más veces más rápida que en versiones anteriores.

## Mejoras en la robustez de Visual FoxPro

Ahora Visual FoxPro 6.0 detecta errores de protección general (General Protection Faults, GPF) en los controles ActiveX colocados en un formulario o en instancias de objetos COM creadas en Visual FoxPro. Ahora se trata un GPF de control ActiveX o de objeto COM como errores detectables de Visual FoxPro ([Error 1440](#) - El objeto OLE puede estar dañado).

## Mejoras en la facilidad de uso de Visual FoxPro

Puede especificar la cadena de comentario del editor de Visual FoxPro en el Registro de Windows. Abra la carpeta Options de Visual FoxPro 6.0 con el Editor del Registro de Windows (RegEdit) y haga clic con el botón secundario del *mouse* en la carpeta. Elija **Nuevo** y, a continuación, **Valor de la cadena**. Escriba el nombre "EditorCommandString" para el nuevo valor de la cadena. Haga clic con el botón secundario del *mouse* en la cadena y elija **Modificar**. Escriba la cadena de comentario del editor (\*!\* es el valor predeterminado que se utiliza cuando esta entrada no existe en el Registro).

Ahora puede tener acceso al menú Formulario desde la ventana de código del formulario. Además, puede ejecutar un formulario con el método abreviado de teclado CTRL+E, incluso desde una ventana de código de un formulario.

## Compatibilidad con el milenio

Se ha mejorado Visual FoxPro 6.0 para proporcionar mejor compatibilidad con el milenio. Esta sección describe las mejoras de Visual FoxPro que facilitan la creación de aplicaciones compatibles con el milenio.

### SET CENTURY TO

La documentación de Visual FoxPro 5.0 indica que si ejecuta el comando SET CENTURY TO sin argumentos adicionales establece el siglo al siglo actual. Esto sólo es cierto en el siglo 20, porque el siglo se establece a 19 independientemente de cuál sea el siglo actual. En Visual FoxPro 6.0, SET CENTURY TO establece el siglo al siglo actual. Además, el valor de SET CENTURY TO en nuevas sesiones de datos se inicializa al siglo actual.

Además, en Visual FoxPro 6.0, se ha modificado el valor predeterminado de ROLLOVER para SET CENTURY con los dos dígitos del año actual más 50 años (si el año actual es 1998, *nAño* es 48, los dos últimos dígitos de 2048 (1998 + 50). En Visual FoxPro 5.0, el valor predeterminado es 0.

Vea [SET CENTURY](#) para obtener más información.

## Formatos estrictos de fecha

Normalmente, las constantes o expresiones de tipo Date y DateTime se interpretan en función de los valores actuales de [SET DATE](#) y [SET CENTURY](#) en el momento en el que las constantes o las expresiones se compilan o evalúan. Esto significa que muchas constantes de fecha son ambiguas puesto que se pueden evaluar como valores diferentes en función de cuándo se compilaron y el valor de fecha del momento de la compilación.

Por ejemplo, ¿es la constante de fecha {10/11/12} 11 de octubre de 1912, 11 de octubre de 2012, 10 de noviembre de 1912, 12 de noviembre de 1910 o 12 de noviembre de 2010?

Todo depende de los valores actuales de SET DATE y SET CENTURY TO. Esto puede introducir errores en el código existente de Visual FoxPro siempre que se compilen o evalúen en tiempo de ejecución constantes o expresiones de tipo Date o DateTime, como expresiones de informe u objeto. Esto puede producir incompatibilidad con el milenio cuando el valor de SET CENTURY pase al año 2000 y no se especifiquen fechas de cuatro dígitos.

Para evitar esta incompatibilidad, ahora se dispone de un formato de fecha estricto en Visual FoxPro 6.0 (y en Visual FoxPro 5.0). Las fechas estrictas siempre se evalúan con el mismo valor Date o DateTime independientemente de los valores de las fechas. El formato de fecha estricto es:

```
^aaaa-mm-dd[,][hh[:mm[:ss]]][a|p]]
```

El carácter (^) denota siempre el formato de fecha estricto y hace que los valores de tipo Date y DateTime se interpreten en formato AMD. Los separadores válidos son los guiones, las barras, los puntos y los espacios.

Los valores de tipo Date y DateTime no son ambiguos y siempre son válidos. Los formatos vacíos de Date y DateTime incluyen {}, {--} y {--:}.

Con los formatos de fecha estrictos tiene a su disposición un rango más grande de valores de tipo Date y DateTime. En Visual FoxPro 5.0, el menor valor de fecha que se puede expresar es {^0100/1/1}, 1 de Enero de 100 A.C. Esto es así porque los valores de año inferiores a 100 siempre se redondeaban hasta el siglo siguiente basándose en el valor de SET CENTURY.

La menor fecha válida en Visual FoxPro 6.0 es {^0001-01-01}, 1 de enero del 1 A.C. La mayor fecha válida en Visual FoxPro 6.0 es {^9999-12-31}, 31 de diciembre de 9999 D.C.

Observe que el formato estricto de fecha ignora el valor TAIWAN en SET DATE, de forma que el año en el formato Date o DateTime estricto siempre hace referencia al calendario occidental. (Recuerde que esto no es así en Visual FoxPro 5.0).

## SET STRICTDATE

Puede utilizar un nuevo comando, [SET STRICTDATE](#), para forzar la compatibilidad de las constantes y cadenas de fechas con el milenio.

### SET STRICTDATE TO 0

Establecer STRICTDATE a 0 significa que la comprobación del formato estricto de fecha está desactivada. Esto es compatible con Visual FoxPro 5.0. 0 es el valor predeterminado para el entorno de tiempo de ejecución de Visual FoxPro y el controlador ODBC. Cuando STRICTDATE está establecido a 0, los valores Date y DateTime no válidos se evalúan como cadenas vacías.

### SET STRICTDATE TO 1

Establecer STRICTDATE a 1 requiere que todas las constantes de tipo Date y DateTime estén en el formato estricto. Cualquier constante de tipo Date o DateTime que no esté en el formato estricto o que se evalúe como valor no válido generará un error, durante la compilación, en tiempo de ejecución o durante una sesión interactiva en Visual FoxPro. 1 es el valor predeterminado para las sesiones interactivas en Visual FoxPro.

### SET STRICTDATE TO 2

Es idéntico a establecer STRICTDATE a 1, pero además genera un error de compilación (2033 – CTOD y CTOT pueden producir resultados incorrectos) siempre que las funciones [CTOD\(\)](#) y [CTOT\(\)](#) aparezcan en el código.

Como los valores devueltos por CTOD() y CTOT() se basan en SET DATE y SET CENTURY para interpretar la fecha que contienen pueden producir errores de incompatibilidad con el milenio. Utilice DATE() y DATETIME() con los argumentos numéricos opcionales para crear constantes y expresiones de tipo Date y DateTime.

Este valor es útil en las sesiones de depuración para detectar el código que pueda contener errores de incompatibilidad con el milenio.

## Errores de formato estricto de fecha

Se han agregado los siguientes errores nuevos a Visual FoxPro 6.0 y se pueden generar cuando SET STRICTDATE está establecido a 1 ó 2.

### Error 2032: constante Date/DateTime ambigua.

Este error se produce cuando un valor de tipo Date o DateTime no cumple el formato estricto. Las siguientes condiciones producirán este error:

- Falta el signo ^.
- Los separadores de fecha no son guiones, ni barras, ni puntos, ni espacios.

- El campo año contiene menos de cuatro caracteres ({^98-02-16}).
- Los campos mes o día están vacíos ({^1998-02}).

**Error 2033: CTOD y CTOT pueden producir resultados incorrectos.**

Este error se produce por las mismas razones que el error 2032, pero CTOD() y CTOT() pueden ser no compatibles o ambiguas. Utilice en su lugar las funciones [DATE\(\)](#) o [DATETIME\(\)](#).

**Error 2034: valor Date/DateTime evaluado a un valor no válido.**

Un valor de tipo Date o DateTime no tiene el formato válido o está fuera del intervalo válido para Date y DateTime.

Cuando SET STRICTDATE está establecido a 0, las constantes de tipo Date o DateTime no válidas se evalúan como constantes Date y DateTime vacías. Cuando se establece SET STRICTDATE a 1 ó 2, las constantes de fecha no válidas como {^2000-02-31}, 31 de febrero o {^2000-01-01,25:00}, 25 en punto, generarán este error.

Algunos ejemplos de valores de tipo Date y DateTime no válidos son:

- {^2000-02-31}, 31 de febrero, 2000.
- {^2000-01-01,25:00} 25 en punto.
- {^2000-01-01, 14a}, 14 A.M.

**Error 2035: un valor Date/DateTime contiene caracteres no válidos.**

La constante Date o DateTime contiene caracteres no admitidos por las constantes de tipo Date y DateTime.

Cuando se establece SET STRICTDATE a 0, las constantes de tipo Date o DateTime que contengan caracteres ilegales se evalúan como valores Date o DateTime vacíos. Cuando se establece SET STRICTDATE a 1 ó 2, la constante de tipo Date o DateTime que contenga los caracteres ilegales generará este error.

Tenga en cuenta que la propiedad [StrictDateEntry](#) no se ve afectada por el valor de SET STRICTDATE. Esta propiedad no cambia en Visual FoxPro 6.0.

**Cuadro de diálogo Opciones**

La [ficha General](#) del cuadro de diálogo **Opciones** incluye ahora un cuadro de lista desplegable **Compatibilidad con el milenio**, que especifica el valor de SET STRICTDATE. Como los demás elementos del cuadro de diálogo Opciones, el valor se establece para la sesión actual de Visual FoxPro y la elección de **Establecer como predeterminado** guarda el valor en el Registro de Windows para la siguiente sesión de Visual FoxPro.

**Funciones DATE() y DATETIME()**

Ahora las funciones [DATE\(\)](#) y [DATETIME\(\)](#) admiten argumentos numéricos opcionales que le permiten crear valores de tipo Date o DateTime compatibles con el milenio. Las mejoras de estas funciones proporcionan ahora un método preferible para crear valores de tipo Date y DateTime; ya no es necesario utilizar funciones de manipulación de caracteres para crearlos.

## Función **FDATE()**

Ahora la función [FDATE\(\)](#) acepta un argumento opcional que le permite determinar la hora de la última modificación de un archivo sin utilizar funciones de manipulación de caracteres. Por ejemplo, en versiones anteriores de Visual FoxPro era necesario escribir código como el siguiente para determinar cuándo se modificó por última vez el archivo de recursos de Visual FoxPro:

```
tLastModified = CTOT(DTOC(FDATE('Foxuser.dbf')) + ' ' ;  
    + FTIME('Foxuser.dbf'))
```

Ahora puede reemplazar este código con el siguiente:

```
tLastModified = FDATE('Foxuser.dbf', P)
```

## Propiedad **Century**

El valor predeterminado de la propiedad [Century](#) en Visual FoxPro 6.0 es 1 – Activado. La parte de siglo de la fecha se muestra en un cuadro de texto. En las versiones anteriores de Visual FoxPro, el valor predeterminado es 2 (el valor de SET CENTURY determina si se muestra la parte de siglo de las fechas).