

*Pequeno  
Guia*

**VISUAL BASIC 5**

---

## Propriedade OLERequestPendingMsgText

Retorna ou define o texto da mensagem alternativa "ocupado" exibido quando a entrada de mouse ou teclado é recebida enquanto um pedido de automação está pendente. Não está disponível durante o tempo de criação.

### Sintaxe

*object*.**OLERequestPendingMsgText** [= *string*]

A sintaxe da propriedade **OLERequestPendingMsgText** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>string</i>	Uma <u>expressão de sequência de caracteres</u> que avalia para o texto da mensagem que será exibida na caixa de mensagem alternativa para a condição pendente do pedido ActiveX.

### Comentários

O Visual Basic exibe uma caixa de diálogo padrão **Component Request Pending** quando uma entrada de mouse ou teclado é recebida enquanto um pedido de automação está pendente. A caixa de diálogo inclui texto e um botão **Switch To** que se destinam a serem usados com componentes ActiveX visíveis como o Microsoft Excel. Existem situações em que a caixa de diálogo padrão pode não atender às suas necessidades:

- Seu programa pode chamar um método de um objeto fornecido por um componente ActiveX que não tenha interface de usuário. Os componentes ActiveX criados usando a Professional Edition do Visual Basic, por exemplo, podem ser executados em segundo plano sem quaisquer formulários visíveis.
- O componente ActiveX chamado pode ter sido criado usando os recursos Remote Automation do Visual Basic, Enterprise Edition, e pode estar sendo executado em outro computador localizado a alguma distância do usuário.
- Se o seu programa carregou uma pasta de trabalho do Microsoft Excel usando a função **GetObject**, a pasta de trabalho não estará visível quando o usuário alternar para o Microsoft Excel. Na realidade, o próprio Microsoft Excel pode não estar visível, caso em que o botão **Switch To** nada faz.

Nestas situações, o texto padrão e o botão **Switch To** são inadequados e podem confundir o usuário de seu programa.

A propriedade **OLERequestPendingMsgText** permite substituir a caixa de diálogo **Component Request Pending** padrão por uma caixa de mensagem alternativa. Definir **OLERequestPendingMsgText** para sua própria sequência de caracteres de mensagem faz com que a caixa de diálogo padrão **Component Request Pending** seja substituída por uma caixa de mensagem simples contendo seu texto de mensagem e um botão **O**.

**Applies To** Uma vez que um pedido de automação tenha sido aceito por um componente ActiveX, não há maneira de cancelá-lo.

Se **OLERequestPendingMsgText** for igual a uma sequência de caracteres vazia (""), a caixa de diálogo **Component Request Pending** padrão é exibida.

**Importante** Quando você sabe que um pedido de automação pode levar mais que alguns segundos, e estiver usando um componente ActiveX remoto ou oculto, deve-se definir uma mensagem alternativa. Para componentes ActiveX remotos, a mensagem alternativa é recomendável para todos os pedidos. Tráfego de rede pode fazer ocasionalmente com que mesmo um pedido ActiveX muito curto leve diversos segundos.

## Propriedade OLERequestPendingMsgTitle

Retorna ou define a legenda da mensagem alternativa "ocupado" exibida quando a entrada de mouse ou teclado é recebida enquanto um pedido de automação está pendente. Não está disponível durante o tempo de criação.

### Sintaxe

*object*.**OLERequestPendingMsgTitle** [= *string*]

A sintaxe da propriedade **OLERequestPendingMsgTitle** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto

*string*

na lista Applies To.

Uma expressão de sequência de caracteres que avalia para a legenda da caixa de mensagem alternativa para a condição pendente do pedido ActiveX.

### Comentários

Caso a propriedade **OLERequestPendingMsgText** tenha sido definida, o valor da propriedade **OLERequestPendingMsgTitle** é usado como legenda da caixa de mensagem "ocupado" alternativa que substitui a caixa de diálogo **Component Request Pending** padrão. O valor padrão da propriedade **OLERequestPendingMsgTitle** é o valor atual da propriedade **Title** do objeto **App**. Esta é uma configuração recomendada.

Se a propriedade **OLERequestPendingMsgText** for definida como uma sequência de caracteres vazia (""), a propriedade **OLERequestPendingMsgTitle** é ignorada.

## Propriedade OLERequestPendingTimeout

Retorna ou define o número de milissegundos que devem decorrer antes que a caixa de diálogo **Component Request Pending** (ou mensagem alternativa) possa ser disparada pela entrada de mouse ou teclado recebida, enquanto o pedido de automação está pendente. Não está disponível durante o tempo de criação.

### Sintaxe

*object*.**OLERequestPendingTimeout** [= *milliseconds*]

A sintaxe da propriedade **OLERequestPendingTimeout** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>milliseconds</i>	Um inteiro <b>Long</b> representando o número de milissegundos que deve decorrer antes que uma mensagem de ocupado seja disparada.

### Comentários

O valor padrão para esta propriedade é 5000 milissegundos (cinco segundos).

**Importante** Este valor de tempo limite também afeta documentos que são vinculados ou incorporados usando o controle **OLE Container** ou a Toolbox. Caso esteja usando documentos vinculados ou incorporados e altera esta propriedade antes de um pedido de automação, pode ser interessante reconfigurar o valor posteriormente.

## Propriedade OLEServerBusyMsgText

Retorna ou define o texto da mensagem alternativa "ocupado" que é exibida no lugar da caixa de diálogo **Component Busy** padrão se um componente ActiveX rejeitar um pedido de automação. Não está disponível durante o tempo de criação.

### Sintaxe

*object*.**OLEServerBusyMsgText** [= *String*]

A sintaxe da propriedade **OLEServerBusyMsgText** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>String</i>	Uma <u>expressão de sequência de caracteres</u> que avalia para a mensagem de texto que será exibida na caixa de mensagem alternativa para a condição ocupada do componente ActiveX.

### Comentários

O Visual Basic continua a repetir um pedido de automação para o número de milissegundos especificados pela propriedade **OLEServerBusyTimeout**. Caso o componente ActiveX não tenha aceito o pedido dentro daquele intervalo, o Visual Basic exibe uma caixa de diálogo **Component Busy** padrão. Esta caixa de diálogo inclui texto e um botão **Switch To** que se destinam a serem usados com componentes ActiveX visíveis como o Microsoft Excel. Existem situações em que a caixa de diálogo padrão pode não atender às suas necessidades:

- O programa pode chamar um método de um objeto fornecido por um componente ActiveX que não tenha uma interface de usuário. Os componentes ActiveX criados usando a Professional

---

Edition do Visual Basic, por exemplo, podem ser executados em segundo plano sem quaisquer formulários visíveis.

- O componente ActiveX chamado pode ter sido criado usando os recursos de Remote Automation do Visual Basic, Enterprise Edition, e pode estar sendo executado em outro computador a alguma distância do usuário.
- Caso seu programa tenha carregado uma pasta de trabalho do Microsoft Excel usando a função **GetObject**, a pasta de trabalho não estará visível quando o usuário alterna para o Microsoft Excel. Na realidade, o próprio Microsoft Excel pode não estar visível, caso em que o botão **Switch To** não faz nada.

Nestas situações, o texto padrão e o botão **Switch To** são inadequados, e podem confundir o usuário de seu programa.

A propriedade **OLEServerBusyMsgText** permite substituir a caixa de diálogo **Component Busy** padrão por uma caixa de mensagem alternativa. Configurando **OLEServerBusyMsgText** para sua própria seqüência de caracteres de mensagem faz com que a caixa de diálogo **Component Busy** seja substituída por uma caixa de mensagem simples contendo seu texto de mensagem, um botão **O** e um botão **Cancel**.

Se **OLERequestPendingMsgText** for igual a uma seqüência de caracteres vazia (""), a caixa de diálogo **Component Busy** padrão é exibida.

Caso o usuário pressione o botão **Cancel** na caixa de diálogo **Component Busy** padrão, ou na caixa de mensagem alternativa, o erro -2147418111 (&H80010001) do ActiveX é produzido no procedimento que efetuou o pedido de automação.

**Importante** Quando se sabe que um pedido de automação pode levar mais que alguns segundos, e está-se usando um componente ActiveX oculto ou remoto deve-se definir uma mensagem alternativa. Para componentes ActiveX remotos, a mensagem alternativa é recomendada para todos os pedidos. Tráfego de rede ocasionalmente pode fazer com que até mesmo um pedido ActiveX muito curto leve vários segundos.

## Propriedade OLEServerBusyMsgTitle

Retorna ou define a legenda da mensagem alternativa "ocupado" que é exibida quando um componente ActiveX rejeita um pedido de automação. Não disponível durante o tempo de criação.

### Sintaxe

*object*.**OLEServerBusyMsgTitle** [= *String*]

A sintaxe da propriedade **OLEServerBusyMsgTitle** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>String</i>	Uma <u>expressão de seqüência de caracteres</u> que avalia para a legenda da caixa de mensagem alternativa para a condição ocupada do componente ActiveX.

### Comentários

Caso a propriedade **OLEServerBusyMsgText** tenha sido definida, o valor da propriedade **OLEServerBusyMsgTitle** é usado como a legenda da mensagem alternativa ocupado que substitui a caixa de diálogo **Component Busy** padrão. O valor padrão da propriedade **OLEServerBusyMsgTitle** é o valor atual da propriedade **Title** do objeto **App**. Esta é a configuração recomendada.

Caso a propriedade **OLEServerBusyMsgText** esteja definida como uma seqüência de caracteres vazia (""), a propriedade **OLEServerBusyMsgTitle** é ignorada.

## Propriedade OLEServerBusyRaiseError

Determina se um pedido de automação rejeitado produz um erro, ao invés de exibir a caixa de diálogo **Component Busy** padrão ou uma mensagem alternativa. Não está disponível durante o tempo de criação.

### Sintaxe

*object*.**OLEServerBusyRaiseError** [= *boolean*]

A sintaxe da propriedade **OLEServerBusyRaiseError** tem estas três partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> avalia para um objeto na

<i>boolean</i>	lista Applies To. Uma <u>expressão booleana</u> que especifica se um erro deve ser produzido, conforme descrito em Configurações.
----------------	--

### Configurações

As configurações de *boolean* são:

Configuração	Descrição
<b>True</b>	(Padrão) Um erro é produzido quando o número de milissegundos especificado pela propriedade <b>OLEServerBusyTimeout</b> foi usado.
<b>False</b>	Dependendo da configuração da propriedade <b>OLEServerBusyMsgText</b> , ou a caixa de diálogo <b>Server Busy</b> padrão ou uma mensagem alternativa de ocupado será exibida.

### Comentários

Produzir um erro quando um componente ActiveX rejeita um pedido de automação provoca o retorno do controle a seu programa, o que lhe permite oferecer sua própria caixa de diálogo personalizada no lugar da caixa de diálogo **Component Busy** padrão ou da mensagem alternativa de ocupado.

O erro de automação que será produzido é -2147418111 (&H80010001).

## Propriedade OLEServerBusyTimeout

Retorna ou define o número de milissegundos durante os quais um pedido de automação continuará a ser repetido, antes que a caixa de diálogo **Component Busy** padrão (ou uma mensagem alternativa) seja exibida. Não disponível durante o tempo de criação.

### Sintaxe

*object*.**OLEServerBusyTimeout** [= *milliseconds*]

A sintaxe da propriedade **OLEServerBusyTimeout** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>milliseconds</i>	Um inteiro <b>Long</b> representando o número de milissegundos durante os quais será repetido um pedido de automação.

### Comentários

O valor padrão desta propriedade é 10.000 milissegundos (dez segundos).

**Importante** Este valor de tempo limite também afeta os documentos vinculados ou incorporados usando o controle **OLE Container** ou a Toolbox. Caso esteja usando documentos vinculados ou incorporados e alterar esta propriedade antes de um pedido de automação, pode ser interessante redefinir os valores posteriormente.

## Propriedade EXENAME

Retorna a parte principal do nome do arquivo executável (sem a extensão) que está sendo executado no momento. Se estiver sendo executada em um ambiente de desenvolvimento, retorna o nome do projeto.

### Sintaxe

*object*.**EXENAME**

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

## Propriedade TasVisible

Retorna ou define um valor que determina se o aplicativo aparece na lista de tarefas do Windows.

---

### Sintaxe

*object.TasVisible* [= *boolean*]

A sintaxe da propriedade **TasVisible** tem estas três partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão booleana</u> que determina se o aplicativo aparece na lista de tarefas, conforme descrito em Configurações.

### Configurações

As configurações para *boolean* são:

Configuração	Descrição
<b>True</b>	(Padrão) O aplicativo aparece na lista de tarefas do Windows.
<b>False</b>	O aplicativo não aparece na lista de tarefas do Windows.

### Comentários

A propriedade **TasVisible** somente pode ser configurada como **False** em aplicativos que não exibem uma interface como, por exemplo componentes ActiveX que não contenham ou exibam objetos **Form**. Enquanto o aplicativo exibe uma interface, a propriedade **TasVisible** é automaticamente definida como **True**.

## Propriedade hInstance

Retorna um identificador à ocorrência do aplicativo.

### Sintaxe

*object.hInstance*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

A propriedade **hInstance** retorna um tipo de dados Long.

Ao trabalhar com um projeto no ambiente de desenvolvimento Visual Basic, a propriedade **hInstance** retorna o identificador de ocorrência da ocorrência de Visual Basic.

## Propriedade Comments

Retorna ou define uma seqüência de caracteres contendo comentários sobre o aplicativo em execução. Somente leitura durante o tempo de execução.

### Sintaxe

*object.Comments*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Pode-se definir esta propriedade durante o projeto na caixa **Type** na guia **Mae** da caixa de diálogo **Project Properties**.

## Propriedade CompanyName

Retorna ou define o valor de uma seqüência de caracteres contendo o nome da empresa ou do criador de um aplicativo em execução. Somente leitura durante o tempo de execução.

### Sintaxe

*object.CompanyName*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Pode-se definir esta propriedade durante o projeto na caixa **Type** na guia **Mae** da caixa de diálogo **Project Properties**.

---

## Propriedade FileDescription

Retorna ou define o valor de uma sequência de caracteres contendo informações sobre descrição de arquivo relativas ao aplicativo em execução. Somente leitura durante o tempo de execução.

### Sintaxe

*object*.FileDescription

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Durante o projeto, pode-se definir esta propriedade na caixa **Type** na guia **Mae** da caixa de diálogo **Project Properties**.

## Propriedade LegalCopyright

Retorna ou define o valor da sequência de caracteres contendo informações legais de copyright sobre o aplicativo em execução. Somente leitura durante o tempo de execução.

### Sintaxe

*object*.LegalCopyright

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Durante o projeto, pode-se definir esta propriedade na caixa **Type** da guia **Mae** na caixa de diálogo **Project Properties**.

## Propriedade LegalTrademarks

Retorna ou define o valor da sequência de caracteres contendo informações legais de marca registrada relativas ao aplicativo em execução. Somente leitura durante o tempo de execução.

### Sintaxe

*object*.LegalTrademarks

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Durante o projeto, pode-se definir esta propriedade na caixa **Type** da guia **Mae** na caixa de diálogo **Project Properties**.

## Propriedade Major

Retorna ou define o número de versão mais importante do projeto. Somente leitura durante o tempo de execução.

### Sintaxe

*object*.Major

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

O valor da propriedade **Major** situa-se no intervalo de 0 a 9.999.

Esta propriedade oferece informações de versão sobre o aplicativo em execução.

Durante o projeto, pode-se definir esta propriedade na caixa **Major** da guia **Mae** da caixa de diálogo **Project Properties**.

## Propriedade Minor

Retorna ou define o número de versão secundário do projeto. Somente leitura durante o tempo de execução.

### Sintaxe

*object*.Minor

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

---

## Comentários

O valor da propriedade **Minor** situa-se no intervalo 0 a 9.999.

Esta propriedade oferece informações de versão sobre o aplicativo em execução.

Durante o projeto, pode-se definir esta propriedade na caixa **Minor** da guia **Mae** na caixa de diálogo **Project Properties**.

## Propriedade ProductName

Retorna ou define o valor de uma seqüência de caracteres contendo o nome do produto do aplicativo em execução. Somente leitura durante o tempo de execução.

### Sintaxe

*object*.**ProductName**

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Durante o projeto, pode-se definir esta propriedade na caixa **Type** da guia **Mae** na caixa de diálogo **Project Properties**.

## Propriedade Revision

Retorna ou define o número de versão da revisão do projeto. Somente leitura durante o tempo de execução.

### Sintaxe

*object*.**Revision**

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

O valor da propriedade **Revision** situa-se no intervalo 0 a 9.999.

Esta propriedade fornece informações de versão sobre o aplicativo em execução.

Durante o projeto, pode-se definir esta propriedade na caixa **Revision** da guia **Mae** na caixa de diálogo **Project Properties**.

## Evento AccesseyPress

Ocorre quando o usuário do controle pressiona uma das teclas de acesso do controle, ou quando a tecla ENTER é pressionada tendo o desenvolvedor definido a propriedade **Default** como **True**, ou quando a tecla ESCAPE é pressionada tendo o desenvolvedor definido a propriedade **Cancel** como **True**. A propriedade **Default** e a propriedade **Cancel** são ativadas pelo autor do controle definindo a propriedade **DefaultCancel** como **True**.

### Sintaxe

**Sub** *object*.**AccesseyPress**(*eyAscii* As Integer)

A sintaxe do evento AccesseyPress tem três partes:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.
<i>eyAscii</i>	Um inteiro que contém o valor Ascii da tecla (sem o ALT) que causou o disparo do evento AccesseyPress, da mesma maneira que o evento padrão eyPress.

## Propriedade Accesseys

Retorna ou define uma seqüência de caracteres contendo as teclas que funcionarão como teclas de acesso para o controle.

### Sintaxe

*object*.**Accesseys**[= *AccesseyString*]

A sintaxe da propriedade **Accesseys** tem estas partes:

Parte	Descrição
-------	-----------

---



---

<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.
<i>AccessesString</i>	Uma seqüência de caracteres contendo as teclas que funcionarão como teclas de acesso.

### Comentários

A propriedade **Accesses** é uma seqüência de caracteres que contém todas as teclas de acesso do controle. Como exemplo, para configurar as letras S e Y como teclas de acesso, a propriedade **Accesses** seria definida como "sy".

Quando um usuário pressiona uma das teclas de acesso juntamente com a tecla ALT, o controle recebe o foco (dependendo da configuração da propriedade **ForwardFocus**).

As teclas de acesso a controles constituintes são implicitamente incluídas como Accesses, embora não apareçam na propriedade **Accesses**.

## Propriedade Alignable

Retorna ou define um valor que determina se um controle é alinhável, ou pode utilizar a propriedade extensora **Align**. A propriedade **Alignable** é de leitura/gravação no momento da criação do controle, e não está disponível durante o tempo de execução do controle.

### Configurações

As configurações de **Alignable** são:

Configuração	Descrição
<b>True</b>	O controle é alinhável; o recipiente adicionará a propriedade <b>Align</b> ao objeto extensor.
<b>False</b>	O controle não é alinhável. Este é o valor padrão.

### Comentários

O alinhamento do próprio controle será manipulado pelo recipiente; o autor do controle pode usar a propriedade extensora **Align** para decidir como redesenhar o controle e dispor os controles constituintes em resposta a um alinhamento.

**Observação:** Nem todos os recipientes suportam controles alinháveis. A interceptação de erros deve ser usada se você acessa a propriedade **Align** extensora para determinar como seu controle foi alinhado.

## Objeto AmbientProperties

Um objeto **AmbientProperties** contém informações do ambiente de um recipiente para sugerir o comportamento a controles contidos no recipiente.

### Comentários

Os recipientes oferecem propriedades de ambiente para sugerir comportamentos aos controles. Como exemplo, **BacColor** é uma das propriedades de ambiente padrão; o recipiente está sugerindo que o controle bem comportado deva definir suas propriedades de cor de fundo.

As propriedades do objeto **AmbientProperties** são as propriedades de ambiente do recipiente. Estas propriedades são somente leitura.

Algumas propriedades de ambiente são padrão, enquanto que outras são específicas de determinados recipientes. Um controle pode acessar propriedades de ambiente não-padrão, mas isto torna o controle específico de recipiente. O controle deve tratar o caso onde uma propriedade de ambiente não esteja presente no recipiente atual.

Quando o controle é compilado, o Visual Basic não tem como saber quais propriedades de ambiente específicas de recipiente podem estar disponíveis quando o controle é executado; portanto, as referências a propriedades de ambiente específicas de recipiente serão sempre acopladas posteriormente.

O objeto **AmbientProperties** não está disponível quando o evento Initialize é produzido; mas está disponível quando o evento InitProperties ou ReadProperties é produzido.

O objeto **AmbientProperties** tem diversas propriedades padrão:

- A propriedade **BacColor**, uma propriedade Color que contém a cor sugerida para o interior do controle contido. O padrão fornecido pelo Visual Basic caso o recipiente não suporte esta propriedade é 0x80000005: a cor do sistema para o segundo plano de uma janela.
- A propriedade **DisplayAsDefault**, uma propriedade Boolean que especifica se o controle é o controle padrão. O padrão do Visual Basic fornecido se o recipiente não suporta esta propriedade é **False**.
- A propriedade **DisplayName**, uma propriedade String contendo o nome de controle que deve

- 
- ser automaticamente exibido. O padrão fornecido pelo Visual Basic se o recipiente não suporta esta propriedade é uma sequência vazia: "".
- A propriedade **Font**, um objeto **Font** que contém as informações de fonte sugeridas do controle contido. O padrão fornecido pelo Visual Basic caso o recipiente não suporte esta propriedade é MS Sans Serif 8.
  - A propriedade **ForeColor**, uma propriedade Color que contém a cor sugerida para o primeiro plano do controle contido. O padrão fornecido pelo Visual Basic caso o recipiente não suporte esta propriedade é 0x80000008: a cor do sistema para texto de janela.
  - A propriedade **LocaleID**, uma propriedade Long que especifica o idioma e país do usuário. O padrão fornecido pelo Visual Basic, caso o recipiente não suporte esta propriedade, é a ID do local atual do sistema.
  - A propriedade **MessageReflect**, uma propriedade Boolean que especifica se o recipiente suporta reflexão de mensagem. O padrão fornecido pelo Visual Basic, caso o recipiente não suporte esta propriedade, é **False**.
  - A propriedade **Palette**, um objeto **Picture** cuja paleta especifica a paleta sugerida para o controle contido.
  - A propriedade **RightToLeft**, uma propriedade Boolean que indica a direção de exibição do texto, e a aparência visual do controle em um sistema bidirecional. O padrão fornecido pelo Visual Basic, caso o recipiente não suporte esta propriedade, é **False**.
  - A propriedade **ScaleUnits**, uma propriedade String contendo o nome da unidade de coordenadas usada pelo recipiente. O padrão fornecido pelo Visual Basic, caso o recipiente não suporte esta propriedade, é uma sequência vazia: "".
  - A propriedade **ShowGrabHandles**, uma propriedade Boolean que especifica se o recipiente trata a exibição de alças de manipulação. O padrão fornecido pelo Visual Basic, caso o recipiente não suporte esta propriedade é **True**.
  - A propriedade **ShowHatching**, uma propriedade Boolean que especifica se o recipiente trata da exibição de hachurado. O padrão fornecido pelo Visual Basic, caso o recipiente não suporte esta propriedade é **True**.
  - A propriedade **SupportsMnemonics**, uma propriedade Boolean que especifica se o recipiente trata de teclas de acesso para o controle. O padrão fornecido pelo Visual Basic, caso o recipiente não suporte esta propriedade é **False**.
  - A propriedade **TextAlign**, uma enumeração que especifica como o texto deve ser alinhado. O padrão fornecido pelo Visual Basic, caso o recipiente não suporte esta propriedade, é **0 - General Align**.
  - A propriedade **UserMode**, uma propriedade Boolean que especifica se o ambiente está em modo de criação ou modo de usuário final. O padrão fornecido pelo Visual Basic, caso o recipiente não suporte esta propriedade. é **True**.
  - A propriedade **UIDead**, uma propriedade Boolean que especifica se a interface de usuário não está respondendo. O padrão fornecido pelo Visual Basic, caso o recipiente não suporte esta propriedade, é **False**.

## Propriedade Ambient

Retorna um objeto **AmbientProperties** contendo as propriedades de ambiente do recipiente. A propriedade **Ambient** não está disponível no momento da criação do controle e será somente leitura durante o tempo de execução do controle.

### Sintaxe

*object.Ambient*

A sintaxe da propriedade **Ambient** tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

## Evento AmbientChanged

Ocorre quando se altera o valor de um propriedade de ambiente.

### Sintaxe

**Sub** *object\_AmbientChanged*(*PropertyName As String*)

A sintaxe do evento **AmbientChanged** tem estas partes:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

<i>PropertyName</i>	Uma seqüência que identifica a propriedade de ambiente que foi modificada.
---------------------	--

### Comentários

Usando *PropertyName*, o controle pode acessar o objeto **AmbientProperties** na propriedade **Ambient** para verificar o novo valor da propriedade de ambiente alterado.

Se for colocada uma ocorrência do controle em um formulário do Visual Basic, e a propriedade **FontTransparent** do formulário for alterada, o evento **AmbientChanged** não se produzirá.

## Evento ApplyChanges

Ocorre quando o usuário pressiona o botão **O** ou o botão **Apply** na página de propriedades, ou quando as páginas de propriedade são alternadas selecionando-se guias.

### Sintaxe

**Sub** *object* **ApplyChanges()**

A sintaxe do evento ApplyChanges tem estas partes:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Quando o evento ApplyChanges é produzido, o autor da página de propriedades precisa tratar da configuração de todo os novos valores de propriedade dos controles; espera-se que o autor tenha mantido o controle de quais propriedades foram alteradas, caso contrário todas as propriedades precisarão ser definidas. Para saber quais controles devem ser alterados, utilize a propriedade **SelectedControls**.

O evento ApplyChanges somente será produzido se a propriedade **Changed** for definida como **True**.

## Método AsyncRead

Inicia a leitura dos dados para o recipiente, a partir de um arquivo ou URL de maneira assíncrona.

### Sintaxe

*object*.**AsyncRead** *Target*, *AsyncType* [, *PropertyName*]

A sintaxe do método **AsyncRead** tem estas três partes:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.
<i>Target</i>	Uma expressão de seqüência de caracteres especificando a localização dos dados. Este pode ser uma caminho ou um URL.
<i>AsyncType</i>	Uma expressão de inteiro especificando como os dados serão apresentados, conforme descrito em Configurações.
<i>PropertyName</i>	Uma expressão de seqüência de caracteres opcional especificando o nome da propriedade a ser carregada.

### Configurações

As configurações para *AsyncType* são:

Configuração	Descrição
<b>vbAsyncTypeFile</b>	Os dados são fornecidos em um arquivo que é criado pelo Visual Basic.
<b>VbAsyncTypeByteArray</b>	Os dados são fornecidos como uma matriz de bytes que contém os dados recuperados. Pressupõe-se que o autor do controle saiba como manipular os dados.
<b>VbAsyncTypePicture</b>	Os dados são fornecidos em um objeto <b>Picture</b> .

### Comentários

Uma vez que os dados pedidos pelo método **AsyncRead** estejam disponíveis, o evento Asyn-

---

cReadComplete será criado no objeto. A leitura assíncrona pode ser cancelada antes de seu término chamando o método **CancelAsyncRead**.

O parâmetro *PropertyName* pode ser qualquer nome arbitrário, uma vez que sua única função é atuar como um identificador para este pedido de dados em particular. O valor em *PropertyName* é usado para identificar a leitura assíncrona em particular para cancelar o método **CancelAsyncRead**, e o valor em *PropertyName* também é usado para identificar a leitura assíncrona em particular que foi completada no evento AsyncReadComplete.

## Evento AsyncReadComplete

Ocorre quando o recipiente completou um pedido de leitura assíncrona.

### Sintaxe

**Sub** *object*.**AsyncReadComplete**(*PropertyValue* As **AsyncProperty**)

A sintaxe de evento AsyncReadComplete tem estas partes:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.
<i>PropertyValue</i>	Um objeto <b>AsyncProperty</b> que contém as propriedades abaixo:
<i>Value</i>	Uma Variant contendo os resultados da leitura assíncrona. Esta é a propriedade padrão.
<i>PropertyName</i>	Uma seqüência de caracteres contendo o nome da propriedade que foi passado no método <b>AsyncRead</b> .
<i>AsyncType</i>	Um número inteiro especificando o tipo de dados contido na propriedade <i>Value</i> , conforme descrito em Configurações.

### Configurações

As configurações para *AsyncType* são:

Configuração	Descrição
<b>vbAsyncTypeFile</b>	A propriedade <i>Value</i> contém uma seqüência de caracteres que é um caminho até um arquivo temporário que contém os dados.
<b>vbAsyncTypeByteArray</b>	A propriedade <i>Value</i> contém uma matriz de bytes que contém os dados.
<b>vbAsyncTypePicture</b>	A propriedade <i>Value</i> contém um objeto de figura do formato correto.

### Comentários

O valor contido em *PropertyName* especifica o pedido de leitura de dados assíncronos em particular que foi completado, e corresponde ao valor dado em um acionamento anterior do método **AsyncRead**.

O código de tratamento de erro devia ter sido colocado no procedimento de evento AsyncReadComplete porque uma condição de erro pode ter interrompido a descarga. Se foi este o caso, este erro ocorre quando a propriedade **Value** do objeto **AsyncProperty** é acessada.

## Propriedade BacStyle (Objeto UserControl)

Retorna ou define um valor indicando o tipo de segundo plano do controle. A propriedade **BacStyle** é para leitura/gravação durante o tempo de criação, e somente leitura durante o tempo de execução do controle.

### Sintaxe

*object*.**BacStyle** [= *enum*]

A sintaxe da propriedade **BacStyle** tem estas partes:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

<i>enum</i>	Um valor enumerado que determina como o segundo plano do controle será exibido, conforme descrito em Configurações.
-------------	---

### Configurações

As configurações para *enum* são:

Configuração	Descrição
<b>0-Transparent</b>	Segundo plano transparente. Os controles atrás deste controle e o segundo plano do formulário que o contém serão visíveis nas áreas vazias deste controle. A área da exibição do controle pode, então, ser dividida em duas áreas, a porção que é parte de um controle constituinte, e o restante. Esta última área da exibição do controle não pode receber desenhos, e a parte dos controles que estão colocados no controle pelo desenvolvedor ou usuário final que caírem nesta última área também serão invisíveis. Os eventos de mouse que caírem nesta última área não pertencerão ao controle, mas ao recipiente subjacente.
<b>1-Opaque</b>	Segundo plano opaco. Este é o padrão.
<b>2-TransparentPaint</b>	Segundo plano transparente. A diferença entre esta opção e <b>0-Transparent</b> é que os controles atrás deste controle e o segundo plano do formulário estarão visíveis através das áreas vazias do controle, mas a área inteira do controle pode receber desenhos, os controles colocados sobre este controle não estarão invisíveis e todos os eventos de mouse que caírem dentro deste controle pertencerão a ele.

## Propriedade BorderStyle

Retorna ou define um valor indicando qual é o estilo de borda do controle. A propriedade **BorderStyle** é de leitura/gravação durante o tempo de criação do controle, e somente leitura durante o tempo de execução do controle.

### Sintaxe

*object*.**BorderStyle** [= *enum*]

A sintaxe da propriedade **BorderStyle** tem estas partes:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.
<i>enum</i>	Um valor enumerado que determina qual é o estilo de borda do controle, conforme descrito em Configurações.

### Configurações

As configurações para *enum* são:

Configuração	Descrição
<b>0-None</b>	Nenhuma borda. Este é o valor padrão.
<b>1-Fixed Single</b>	Uma única linha é desenhada ao redor do controle.

## Método CancelAsyncRead

Cancela um pedido de dados assíncronos.

### Sintaxe

*object*.**CancelAsyncRead** [*PropertyName*]

A sintaxe do método **CancelAsyncRead** tem estas partes:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.
<i>PropertyName</i>	Uma expressão de sequência de caracteres opcional

---

pecificando o nome do pedido de dados assíncronos a serem cancelado.

### Comentários

Somente a leitura de dados assíncronos especificada por *PropertyName* é cancelada; todas as outras continuam normalmente.

O valor contido em *PropertyName* especifica um determinado pedido de leitura de dados assíncronos, e deve corresponder ao valor dado em um acionamento anterior do método **AsyncRead**. Caso *PropertyName* não seja fornecido, o último acionamento do método **AsyncRead** que não forneceu um *PropertyName* será cancelado.

## Propriedade CanGetFocus

Retorna ou define um valor que determina se um controle pode receber o foco. A propriedade **CanGetFocus** é de leitura/gravação durante o tempo de criação do controle, e não está disponível durante o tempo de execução.

### Configurações

As configurações para **CanGetFocus** são:

Configuração	Descrição
<b>True</b>	O controle pode receber foco. Caso o controle contenha controles constituintes, o próprio controle poderá receber o foco, a menos que nenhum de seus controles constituintes possa receber o foco. Cabe ao autor do controle escrever o código que desenha o retângulo do foco no controle quando ele recebe o foco. Este é o valor padrão.
<b>False</b>	O controle não pode receber foco.

### Comentários

Desde que o controle contenha pelo menos um controle constituinte, que tenha sido configurado para receber o foco, **CanGetFocus** não pode ser definido como **False**. Caso **CanGetFocus** seja **False**, nenhum controle constituinte pode ser configurado para receber o foco.

## Método CanPropertyChange

Pergunta ao recipiente se uma propriedade acoplada a uma origem de dados pode ter seu valor alterado. O método **CanPropertyChange** é mais útil se a propriedade especificada em *PropertyName* estiver acoplada a uma origem de dados.

### Sintaxe

*object*.**CanPropertyChange** *PropertyName*

A sintaxe do método **CanPropertyChange** tem estas partes:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.
<i>PropertyName</i>	Uma expressão de seqüência de caracteres que representa um nome da propriedade que o controle está pedindo para alterar.

### Valores de retorno

Os valores de retorno possíveis para **CanPropertyChange** são:

Configuração	Descrição
<b>True</b>	A propriedade especificada em <i>PropertyName</i> pode ser alterada neste momento.
<b>False</b>	A propriedade especifica em <i>PropertyName</i> não pode ser alterada desta vez; o recipiente tem a tabela acoplada a dados aberta somente leitura. Não configure o valor da propriedade; fazer isto pode causar erros em alguns recipientes do controle.

### Comentários

O controle deve sempre chamar **CanPropertyChange** antes de alterar o valor de uma propriedade que pode estar acoplada a dados.

**Observação:** Atualmente, **CanPropertyChange** sempre retorna **True** em Visual Basic, mesmo que o campo acoplado seja somente leitura na origem dos dados. O Visual Basic não produz um erro quando o controle tenta alterar um campo somente leitura; ele apenas não atualiza a origem dos dados.

Como exemplo, o código abaixo mostra como o método **CanPropertyChange** é utilizado:

```
Public Property Let Address(ByVal cValue As String)
    If CanPropertyChange("Address") Then
        m_Address = cValue
        PropertyChanged "Address"
    End If
End Property
```

## Propriedade Changed

Retorna ou define um valor indicando que um valor de uma propriedade ou uma página de propriedades foi alterado. A propriedade **Changed** não está disponível durante o tempo de autoria da página de propriedades, e para leitura/gravação durante o tempo de execução da página de propriedades.

### Sintaxe

*object.Changed* [= *boolean*]

A sintaxe da propriedade **Changed** tem estas partes:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.
<i>boolean</i>	Um valor booleano que determina se uma propriedade na página de propriedades foi alterada, tornando suja a página de propriedades.

### Configurações

As configurações de *boolean* são:

Configuração	Descrição
<b>True</b>	A página de propriedades está suja agora, uma vez que o valor de uma propriedade na página foi alterado.
<b>False</b>	A página de propriedades não está suja, e nenhuma propriedade da página teve seu valor alterado.

### Comentários

Quando o usuário altera o valor de propriedades em uma página de propriedades, estas alterações não devem ser efetivas imediatamente; ao invés, as alterações somente serão aplicadas caso o usuário pressione o botão **Aplicar**, o botão **O** ou alterar páginas de propriedade selecionando guias. Isto permite ao usuário anular com facilidade qualquer alteração que tenha sido feita em uma página de propriedades.

A propriedade **Changed** deveria ser configurada como **True**, por exemplo, quando um usuário altera um valor de propriedade em uma página de propriedades. A configuração da propriedade **Changed** como **True** modificaria a página de propriedades para tornar disponível o botão **Aplicar**.

## Propriedade ContainedControls

Retorna uma coleção dos controles que foram adicionados ao controle pelo desenvolvedor ou usuário final durante o tempo de criação do controle. A propriedade **ContainedControls** não está disponível durante o tempo da autoria do controle, e somente leitura durante o tempo de execução.

### Sintaxe

*object.ContainedControls*

A sintaxe da propriedade **ContainedControls** tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

A coleção **ContainedControls** é preenchida com todos os controles que foram adicionados ao

controle pelo desenvolvedor ou pelo usuário final. O controle pode usar a coleção **ContainedControls** para executar operações sobre qualquer um destes controles contidos. Esta coleção funciona de maneira parecida com a coleção **Controles** em um formulário. Para permitir que controles contidos sejam colocados no controle, a propriedade **ControlContainer** deve ser **True**.

Controles contidos não podem ser adicionados ou removidos através desta coleção **ContainedControls**; os controles contidos devem ser alterados de toda maneira permitida pelo recipiente. A propriedade **ContainedControls** pode não ser suportada por todos os recipientes, embora o recipiente possa suportar o controle que tem controles contidos; os formulários Visual Basic suportam esta propriedade. Se esta propriedade não for suportada, as chamadas à coleção **ContainedControls** provocarão erros; utilize tratamento de erros ao acessar a coleção. Observe, contudo, que se o tratamento de erros for executado durante um procedimento de evento como o procedimento de evento **InitProperties** ou o procedimento de evento **ReadProperties**, o tratamento de erros não provocará um evento de erro; fazer isto pode ser fatal para o recipiente.

A coleção **ContainedControls** não está disponível quando o evento **Initialize** é produzido, mas estará disponível quando o evento **InitProperties** ou o evento **ReadProperties** for criado.

Uma vez que a coleção **ContainedControls** esteja presente, ela pode não conter imediatamente referências a controles que um desenvolvedor tenha colocado no controle. Por exemplo, caso o controle seja um formulário do Visual Basic, a propriedade **Count** da coleção **ContainedControls** será zero até depois que o evento **ReadProperties** tenha sido executado.

## Propriedade **ControlContainer**

Retorna ou define um valor que determina se um controle pode conter controles colocados nele pelo desenvolvedor ou pelo usuário final durante o tempo de execução do controle; da mesma forma que o controle **PictureBox** pode conter outros controles. A propriedade **ControlContainer** é para leitura/gravação durante o tempo de autoria, e não está disponível durante o tempo de execução do controle.

### Configurações

As configurações der **ControlContainer** são:

Configuração	Descrição
<b>True</b>	O controle pode conter controles nele colocados. Se uma ocorrência deste controle for colocada em um recipiente que não tenha conhecimento de <b>ISimpleFrame</b> , o suporte de controles contidos será desativado. O controle continuará a funcionar corretamente quanto a todo o resto, mas os desenvolvedores ou usuários finais não poderão colocar controles em uma ocorrência deste controle.
<b>False</b>	O controle não pode conter qualquer controle nele colocado. Este é o valor padrão.

### Comentários

O suporte a controle contido não funciona em um formulário do Visual Basic.

Controles contidos colocados em um controle com um segundo plano transparente somente são visíveis onde sua localização se sobrepõe a qualquer controle constituinte. Eventos de mouse serão passados ao controle contido somente se eles ocorrerem onde o controle contido estiver visível.

## Coleção **DataBindings**

A coleção **DataBindings** é uma propriedade extensora que coleciona as propriedades acopláveis disponíveis para o desenvolvedor e o usuário final.

### Comentários

Todas as propriedades acopláveis aparecem na coleção **DataBindings** durante o tempo de execução de usuário final. Durante o tempo de criação do desenvolvedor (tempo de execução do controle), somente as propriedades marcadas "Exibir na coleção **DataBindings** durante o tempo de criação" aparecerão quando a propriedade **DataBindings** for acessada na janela **Properties**.

## Propriedade **DefaultCancel**

Retorna ou define um valor que determina se um controle pode funcionar como um botão de comando padrão. A propriedade **DefaultCancel** é de leitura/gravação durante o tempo de autoria do controle, e não está disponível durante o tempo de execução do controle.

### Configurações

As configurações para **DefaultCancel** são:



Configuração	Descrição
<b>True</b>	O controle pode atuar como um botão de comando de cancelamento ou padrão. O recipiente adicionará as propriedades <b>Default</b> e <b>Cancel</b> ao objeto extensor. A presença das propriedades <b>Default</b> e <b>Cancel</b> permitem ao controle funcionar como um botão de comando padrão. O controle pode, então definir estas propriedades do extensor adicionadas.
<b>False</b>	O controle não pode agir como um botão de comando padrão ou cancelar. Nenhum controle constituinte tem sua propriedade <b>Default</b> ou <b>Cancel</b> configurada como <b>True</b> . Este é o valor padrão.

### Comentários

Configura a propriedade **Default** como **True** e também ter um controle constituinte com sua propriedade **Default** configurada como **True** fará com que o controle constituinte seja pressionado quando a tecla ENTER for pressionada, caso contrário, o evento **AccesseyPress** do controle será produzido quando a tecla ENTER for pressionada.

Configurar a propriedade **Cancel** como **True** e também ter um controle constituinte com sua propriedade **Cancel** definida como **True** fará com que o controle constituinte seja pressionado quando a tecla ESCAPE for pressionada, caso contrário, o evento **AccesseyPress** será produzido quando a tecla ESCAPE for pressionada.

**Importante** O status de um botão padrão ou cancelar pode mudar a qualquer momento. O código deve ser colocado no procedimento de evento **AmbientChanged** do controle, para detectar alterações na propriedade **DisplayAsDefault** e a aparência do controle deve ser ajustada de acordo.

## Propriedade DisplayAsDefault

Retorna um valor booleano para determinar se o controle é o botão padrão para o recipiente e, portanto, deve ser exibido como controle padrão.

### Sintaxe

*object*.**DisplayAsDefault**

A sintaxe da propriedade **DisplayAsDefault** tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Configurações

Os valores booleanos de retorno possíveis da propriedade **DisplayAsDefault** são:

Configuração	Descrição
<b>True</b>	O controle é o botão padrão.
<b>False</b>	O controle não é o botão padrão. Se o recipiente não implementa esta propriedade de ambiente, este será o valor padrão.

### Comentários

Apenas um controle em um recipiente pode ser o padrão; o recipiente do controle determinará qual deles é atualmente o controle padrão e o notificará através da propriedade de ambiente **DisplayAsDefault**. O controle notificado deve desenhar a si mesmo para mostrar que ele é o padrão. Todos os outros controles terão seu valor de propriedade ambiente **DisplayAsDefault** definidos como **False**.

Somente controles do tipo botão podem ser padrão.

## Propriedade DisplayName

Retorna um valor de sequência de caracteres contendo o nome que o controle deve exibir para identificar-se em mensagens de erro.

### Sintaxe

*object*.**DisplayName**

A sintaxe da propriedade **DisplayName** tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista

### Comentários

Esta propriedade de ambiente é a maneira encontrada pelo controle para descobrir qual recipiente (por exemplo, o Visual Basic) está chamando esta ocorrência do controle. Esta sequência de caracteres deve ser usada em mensagens de erro como o nome da ocorrência do controle. Caso o recipiente não implemente esta propriedade de ambiente, o valor padrão será uma sequência de caracteres vazia.

## Propriedade EditAtDesignTime

Retorna ou define um valor que determina se um controle pode tornar-se ativo durante o tempo de criação do desenvolvedor. A propriedade **EditAtDesignTime** é de leitura/gravação durante o tempo de autoria de controle, e não está disponível durante o tempo de execução do controle.

### Configurações

As configurações para **EditAtDesignTime** são:

Configuração	Descrição
<b>True</b>	Permite que o controle torne-se ativo durante o tempo de criação. Um <b>Edit</b> aparecerá no menu de contexto do controle. Quando o desenvolvedor que utiliza o controle escolhe <b>Edit</b> , o controle se torna ativo e comporta-se como faria durante o tempo de execução pelo usuário final.
<b>False</b>	O controle não pode tornar-se ativo durante o tempo de criação. Este é o valor padrão.

### Comentários

O controle permanecerá ativo enquanto estiver selecionado. Quando o desenvolvedor selecionar outro controle, este controle não estará mais ativo, mesmo que o desenvolvedor clique de volta nele. O desenvolvedor deverá selecionar novamente **Edit** no menu de contexto para tornar ativo o controle.

**Observação:** Quando o controle é ativado desta maneira, os eventos do objeto **UserControl** ocorrerão de tal forma que o controle pode operar normalmente, mas não pode produzir qualquer evento. O método **RaiseEvent** será simplesmente ignorado; não provocará um erro.

## Evento EditProperty

Ocorre quando uma página de propriedades está aberta porque o desenvolvedor pressionou o botão de reticências para exibir uma determinada propriedade para edição.

### Sintaxe

**Sub** *object\_EditProperty*(*PropertyName* **As** String)

A sintaxe do evento **EditProperty** tem estas partes:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista <b>Applies To</b> .
<i>PropertyName</i>	Uma sequência de caracteres que identifica a propriedade que deve ser exibida e editada pela página de propriedades.

### Comentários

Este evento ocorre quando uma propriedade é designada a uma página de propriedades através da caixa de diálogo **Attributes**. Designar uma página de propriedades através da caixa de diálogo **Attributes** significa que a propriedade é exibida na janela de propriedade com reticências (...) a seu lado, e o desenvolvedor pode pressionar o botão de reticências e a página de propriedades será automaticamente aberta; o evento **EditProperty** é, então, produzido, de modo que o autor da página de propriedades possa colocar o cursor no campo correto.

## Evento EnterFocus

Ocorre quando o foco entra no objeto. O próprio objeto poderia estar recebendo o foco, ou um controle constituinte poderia estar recebendo o foco.

### Sintaxe

**Sub** *object\_EnterFocus*()

A sintaxe do evento **EnterFocus** tem estas partes:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Este evento é útil se *object* precisa saber que o foco agora está dentro dele.

O evento EnterFocus é produzido antes de qualquer evento GotFocus; o evento GotFocus somente será produzido em *object* ou em controle constituinte de *object* que efetivamente tenha recebido o foco.

## Propriedade EventsFrozen

Retorna um valor indicando se o recipiente está atualmente ignorando eventos produzidos pelo controle. A propriedade **EventsFrozen** não está disponível durante o tempo de criação, e somente leitura durante o tempo de execução do controle.

### Sintaxe

*object*.EventsFrozen

A sintaxe da propriedade **EventsFrozen** tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Quando a propriedade **EventsFrozen** é **True**, o recipiente está ignorando quaisquer eventos que esteja sendo produzido pelo controle. Caso o controle queira produzir um evento que não possa ser perdido, você deve colocá-los em fila até que **EventsFrozen** seja **False**.

## Evento ExitFocus

Ocorre quando o foco deixa o objeto. O próprio objeto pode estar perdendo o foco, ou um controle constituinte pode estar perdendo o foco.

### Sintaxe

Sub *object*.ExitFocus()

A sintaxe do evento ExitFocus tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Este evento é útil caso *object* precise saber que o foco o está deixando agora.

O evento ExitFocus é produzido após qualquer evento LostFocus; o evento LostFocus somente será produzido em *object* ou controle constituinte de *object* que efetivamente perca o foco.

## Objeto Extender

Um objeto **Extender** contém propriedades do controle que são realmente controladas pelo recipiente do controle, ao invés do próprio controle.

### Comentários

Algumas propriedades de um controle são oferecidas pelo recipiente ao invés do controle; estas são propriedades extensoras. Exemplos de propriedades extensoras são: **Name**, **Tag** e **Left**. O controle ainda precisa saber qual é o valor destas propriedades extensoras, e algumas vezes precisa poder alterar uma propriedade extensora; o objeto **Extender** entrega o acesso do controle a estas propriedades.

Algumas propriedades extensoras são padronizadas, enquanto que outras são específicas de determinados recipientes. Um controle pode acessar propriedades extensoras não-padrão, mas isto torna o controle específico do recipiente. Caso o controle faça uso de uma propriedade extensora, ele deve tratar o caso onde a propriedade extensora não é suportada pelo recipiente atual.

Quando o controle é compilado, o Visual Basic não tem como saber quais propriedades extensoras estarão disponíveis durante o tempo de execução do controle; portanto as referências a propriedades extensoras serão sempre de acoplamento posterior.

---

Um objeto **Extender** não está disponível quando o evento **Initialize** é produzido; mas estará disponível quando o evento **InitProperties** ou evento **ReadProperties** for produzido.

O objeto **Extender** tem diversas propriedades padrão:

- A propriedade **Name**, uma propriedade **String** somente leitura que contém o nome definido pelo usuário para o controle.
- A propriedade **Visible**, uma propriedade **Boolean** de leitura/gravação que especifica se o controle é ou não visível.
- A propriedade **Parent**, um objeto somente leitura que representa o recipiente do controle, por exemplo, um formulário no Visual Basic.
- A propriedade **Cancel**, uma propriedade **Boolean** somente leitura que indica se o controle o botão **Cancel** padrão do recipiente.
- A propriedade **Default**, uma propriedade **Boolean** somente leitura que indica se o controle o botão padrão para o recipiente.

O Visual Basic oferece mais métodos, propriedades e eventos extensores; outros recipientes não garantem oferecer estes métodos, propriedades e eventos extensões. Estes métodos, propriedades e eventos extensores específicos do Visual Basic são:

- A propriedade **Container**, um objeto somente leitura que representa o recipiente visual do controle.
- A propriedade **DragIcon**, uma propriedade **Pictures** para leitura/gravação que especifica o ícone a ser usado quando o controle é arrastado.
- A propriedade **DragMode**, uma propriedade **Integer** de leitura/gravação que especifica se o controle será automaticamente arrastado, ou se o usuário do controle deve chamar o método **Drag**.
- A propriedade **Enabled**, uma propriedade **Boolean** somente leitura que especifica se o controle está ativado. Esta propriedade extensora não está presente, a menos que o controle também tenha uma propriedade **Enabled** com a ID de procedimento correta. Para maiores informações, consulte o tópico "Permitindo que seus controles sejam ativados e desativados" no capítulo 9: Criando controles **ActiveX**.
- A propriedade **Height**, uma propriedade **Integer** de leitura/gravação que especifica a altura do controle em unidades de escala do recipiente.
- A propriedade **HelpContextID**, uma propriedade **Integer** de leitura/gravação que especifica a identificação de contexto a ser usada quando a tecla **F1** for pressionada quando o controle tem o foco.
- A propriedade **Index**, uma propriedade somente leitura que especifica a posição ocupada por esta ocorrência do controle em uma matriz de controle.
- A propriedade **Left**, uma propriedade **Integer** de leitura/gravação que especifica a posição desde a borda esquerda do controle até a borda esquerda do recipiente, especificada em unidades de escala do recipiente.
- A propriedade **TabIndex**, uma propriedade **Integer** que especifica a posição do controle na ordem de tabulação dos controles no recipiente.
- A propriedade **TabStop**, uma propriedade **Boolean** de leitura/gravação que especifica se **Tab** parará no controle.
- A propriedade **Tag**, uma propriedade **String** de leitura/gravação que contém um valor definido pelo usuário.
- A propriedade **ToolTipText**, uma propriedade **String** de leitura/gravação que contém o texto a ser exibido quando o cursor permanece sobre o controle por mais de um segundo.
- A propriedade **Top**, uma propriedade **Integer** de leitura/gravação que especifica a posição desde a borda superior do controle até a borda superior do recipiente, especificada em unidades de escala do recipiente.
- A propriedade **WhatThisHelpID**, uma propriedade **Integer** de leitura/gravação que especifica a identificação de contexto a ser usada quando a pop-up **O que é isto?** é usada no controle.
- A propriedade **Width**, uma propriedade de leitura/gravação que especifica a largura do controle em unidades de escala do recipiente.
- O método **Drag**, um método que inicia, termina ou cancela uma operação de arrasto do controle.
- O método **Move**, um método para mover a posição do controle.
- O método **SetFocus**, um método para definir o foco para o controle.
- O método **ShowWhatsThis**, um método para exibir um tópico selecionado em um arquivo de **Ajuda** usando a pop-pop **O que é isto?** oferecida pela **Ajuda**.
- O método **ZOrder**, um método para colocar o controle na frente ou atrás da ordem-z, dentro de seu nível gráfico.
- O evento **DragDrop**, um evento que é produzido quando outro controle no formulário é solto sobre este controle.

- O evento **DragOver**, um evento que é produzido quando outro controle do formulário é arrastado sobre este controle.
- O evento **GotFocus**, um evento que é produzido quando este controle recebe o foco.
- O evento **LostFocus**, um evento que é produzido quando este controle perde o foco.

## Propriedade Extender

Retorna o objeto **Extender** para este controle que contém as propriedades do controle que são rastreadas pelo recipiente. A propriedade **Extender** não está disponível durante o tempo de autoria do controle, e somente leitura durante o tempo de execução do controle.

### Sintaxe

*object*.**Extender**

A sintaxe da propriedade **Extender** tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

## Propriedade ForwardFocus

Retorna ou define um valor que determina qual controle recebe o foco quando uma das teclas de acesso do controle é pressionada. A propriedade **ForwardFocus** é para leitura/gravação durante o tempo de autoria do controle, e não está disponível durante o tempo de execução do controle.

### Configurações

As configurações para **ForwardFocus** são:

Configuração	Descrição
<b>True</b>	O próximo controle na ordem de tabulação receberá o foco quando uma das teclas de acesso do controle for pressionada.
<b>False</b>	Caso a propriedade <b>CanGetFocus</b> seja True, o próprio controle receberá o foco quando uma das teclas de acesso do controle for pressionada. Este é o valor padrão.

### Comentários

A propriedade **ForwardFocus** permite ao controle implementar o comportamento de um controle **Label** que tenha uma tecla de acesso.

Teclas de acesso são configuradas por meio da propriedade **Accesskeys**. Quando é pressionada uma tecla de acesso, em conjunto com a tecla ALT, o evento **AccessKeyPress** do controle é produzido.

## Método GoBac

Executa um salto de hiperlink de volta na lista de históricos.

### Sintaxe

*object*.**GoBac**

A sintaxe do método **GoBac** tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Caso o objeto esteja em um recipiente que suporte hiperlink OLE, o recipiente saltará até o local que está no passado na lista de históricos. Caso o objeto esteja em um recipiente que não suporte hiperlinks OLE, então este método provocará um erro.

## Método GoForward

Executa um salto de hiperlink para frente na lista de históricos.

### Sintaxe

*object*.**GoForward**

A sintaxe do método **GoForward** tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista

### Comentários

Caso o objeto esteja em um recipiente que suporte hyperlinks OLE, o recipiente saltará até o local que está no futuro na lista de históricos. Caso o objeto esteja em um recipiente que não suporta hyperlinks OLE, este método provocará um erro.

## Evento GotFocus (Objeto UserControl e Objeto UserDocument)

Ocorre no objeto ou controle constituinte quando o foco entra nele.

### Sintaxe

#### Sub *object*\_GotFocus()

A sintaxe do evento GotFocus tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Este evento GotFocus não é o mesmo evento extensor GotFocus manipulado pelo desenvolvedor que utiliza *object*. Este evento GotFocus é para o autor de *object*, e é interno a *object*.

Este evento é útil se *object* precisa saber que o foco está agora nele.

O próprio *object* pode receber foco somente quando a propriedade **CanGetFocus** for **True**, e não existir controles constituintes que possam receber o foco.

O evento EnterFocus é produzido antes do evento GotFocus.

Não produz o evento extensor GotFocus a partir deste evento.

## Evento Hide (Objeto UserControl)

Ocorre quando a propriedade **Visible** do objeto se altera para **False**.

### Sintaxe

#### Sub *object*\_Hide()

A sintaxe do evento Hide tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Para desenhar na tela no Windows, qualquer objeto deve ter uma janela, temporária ou permanente; os controles ActiveX do Visual Basic têm janelas permanentes. Antes que um controle tenha se situado em um formulário, sua janela não está no recipiente. O controle recebe eventos Hide quando a janela é removida.

Enquanto a janela do controle está no formulário, o objeto recebe um evento Hide quando a propriedade **Visible** do controle se altera para **False**.

O controle *não* recebe eventos Hide se o formulário estiver oculto e, em seguida exibido, ou se o formulário for minimizado e, em seguida, restaurado. A janela do controle permanece no formulário durante estas operações, e sua propriedade **Visible** não se altera.

Caso o controle esteja sendo exibido em um navegador da Internet, o evento Hide ocorre quando a página é movida para a lista de histórico.

**Observação:** Caso o controle seja utilizado com uma versão anterior do Visual Basic, que não a 5.0, o controle não receberá eventos Hide durante o tempo da criação. Isto ocorre porque versões antigas do Visual Basic não colocam nenhuma janela visível em um formulário durante o tempo de criação.

## Evento Hide (Objeto UserDocument)

Ocorre quando a propriedade **Visible** do objeto se altera para **False**.

### Sintaxe

#### Sub *object*\_Hide()

A sintaxe do evento Hide tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

## Comentários

Para desenhar na tela no Windows, um objeto precisa de uma janela, temporária ou permanente. Os documentos ActiveX do Visual Basic têm janelas permanentes. O objeto **UserDocument** recebe eventos Hide quando a janela é removida.

Enquanto a janela do *object* Windows está no recipiente, *object* recebe um evento Hide quando a propriedade **Visible** do *object* se altera para **False**.

*Object* não recebe eventos Hide se o recipiente for ocultado e novamente exibido, ou se o recipiente for minimizado e, em seguida, restaurado. A janela do *objeto* permanece no recipiente durante estas operações, e sua propriedade **Visible** não se altera.

Caso *object* esteja sendo exibido em um navegador da Internet, ocorre um evento Hide quando a página é movida para a lista de históricos, navegando para fora de *object* para outro documento, ou quando o Internet Explorer 3.0 é fechado enquanto o *object* está sendo visualizado ou está dentro do cache de documentos ativos. Use este evento para destruir qualquer referência global de objeto, antes de navegar até outro documento.

**Observação:** Se *object* estiver sendo usado com versões mais antigas do Visual Basic, que não o 5.0, o *object* não receberá eventos Hide durante o tempo de criação. Isto ocorre porque versões antigas do Visual Basic não colocavam nenhuma janela visível em um formulário durante o tempo de criação.

## Evento InitProperties

Ocorre quando uma nova ocorrência de um objeto é criada.

### Sintaxe

**Sub** *object* **InitProperties()**

A sintaxe do evento InitProperties tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

## Comentários

Este evento permite ao autor do objeto inicializar uma nova ocorrência do objeto. Este evento somente ocorre quando uma nova ocorrência de um objeto está sendo criada; isto é para permitir ao autor do objeto distinguir entre criar uma nova ocorrência do objeto e carregar uma ocorrência antiga do objeto.

Colocando código para inicializar novas ocorrências no evento InitProperties, ao invés do evento Initialize, o autor pode evitar casos em que carregar dados através de um evento ReadProperties em uma ocorrência antiga do objeto anula a inicialização do objeto.

## Propriedade InvisibleAtRuntime

Retorna ou define um valor que determina se um controle não deve ter uma janela visível durante o tempo de execução. A propriedade **InvisibleAtRuntime** é para leitura/gravação durante o tempo de criação, e não está disponível durante o tempo de execução do controle.

### Configurações

As configurações para **InvisibleAtRuntime** são:

Configuração	Descrição
<b>True</b>	Permite que o controle não tenha uma janela visível durante o tempo de execução. O recipiente do controle pode manter o controle invisível durante o tempo de execução, como, por exemplo, o controle <b>Timer</b> . O controle ainda está ativo e, portanto, o desenvolvedor que utiliza o controle ainda pode escrever programas que podem interagir com o controle. Não existirá propriedade <b>Visible</b> no objeto extensor.
<b>False</b>	O controle age como um controle normal durante o tempo de execução, onde o estado da propriedade extensora <b>Visible</b> determina a visibilidade do controle. Este é o valor padrão.

## Comentários

**Importante** Não utilize a propriedade extensora **Visible** para tornar o controle visível durante o tempo de execução. Se isto for feito, o controle ainda terá todo o ônus de um controle visível durante o tempo de execução. Além disso, as propriedades extensoras estão disponíveis para o desenvolvedor e usuário final, que podem tornar o controle visível.

---

Alguns recipientes podem não suportar a propriedade **InvisibleAtRuntime**; neste caso, o controle estará visível durante o tempo de execução.

Antes de criar um controle que seja visível durante o tempo de execução, considere a possibilidade de criar um objeto comum oferecido por um componente de código em processo (DLL ActiveX). Os objetos oferecidos por componentes de código em processo exigem menos recursos que controles, mesmo controles invisíveis. O único motivo para implementar um controle invisível é aproveitar um recurso que somente está disponível para os controles ActiveX.

## Propriedade LocaleID

Retorna um valor long que contém a identificação do local (idioma e país) do usuário.

### Sintaxe

*object*.LocaleID

A sintaxe da propriedade **LocaleID** tem esta parte:

Parte	Descrição
<i>Object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

A propriedade de ambiente **LocaleID** contém a identificação do idioma e país do usuário atual. Usando esta identificação, o controle pode modificar seu comportamento e aparência para ajustar-se ao idioma e país. Isto pode ir desde simples notificações de erro no idioma do usuário, até modificações mais complexas de propriedade, método e nomes de evento no idioma do usuário. Caso o recipiente não implemente esta propriedade de ambiente, o valor padrão será a System LocaleID atual.

## Evento LostFocus (Objeto UserControl e Objeto UserDocument)

Ocorre no objeto ou controle constituinte quando o foco o deixa.

### Sintaxe

**Sub** *object*.LostFocus()

A sintaxe do evento LostFocus tem estas partes:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Este evento LostFocus não é o mesmo evento extensor LostFocus manipulado pelo desenvolvedor que utiliza *object*. Este evento LostFocus é para o autor de *object*, e é interno em relação a *object*.

Este evento é útil se *object* precisa saber que o foco está nele neste momento.

O próprio *Object* pode receber o foco somente quando a propriedade **CanGetFocus** é **True**, e não houver controles constituintes que possam receber foco.

O evento LostFocus é produzido antes do evento ExitFocus.

## Propriedade MessageReflect

Retorna um valor booleano informando se o recipiente do controle trata automaticamente o reflexo de mensagem.

### Sintaxe

*object*.MessageReflect

A sintaxe da propriedade **MessageReflect** tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Configurações

Os valores de retorno booleanos possíveis da propriedade **MessageReflect** são:

Configuração	Descrição
<b>True</b>	O recipiente do controle refletirá mensagens.
<b>False</b>	O recipiente do controle não pode refletir mensagens. Caso



---

o recipiente não implemente esta propriedade de ambiente, este será o valor padrão.

### Comentários

Quando o controle está em uma subclasse, existem determinadas mensagens que são normalmente enviadas ao controle pai. Sob condições normais, estas mensagens são, na realidade, refletidas de volta ao controle que as enviou, de modo que o controle possa tratar sua própria mensagem. Este espelhamento de mensagem pode ser tratada pelo recipiente, que refletirá as mensagens de volta como eventos. A propriedade **MessageReflect** informa se o recipiente do controle dispõe de espelhamento de mensagem.

Caso o controle seja colocado em um recipiente que não espelhe mensagens, a operação do controle será seriamente comprometida; grande parte da operação de controle depende de mensagens refletidas.

## Método NavigateTo

Executa um salto de hyperlin até o alvo especificado.

### Sintaxe

*object.NavigateTo* *Target* [, *Location* [, *FrameName*]]

A sintaxe do método **NavigateTo** tem estas partes:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.
<i>Target</i>	Uma expressão de sequência de caracteres especificando a localização para onde deve ser dado o salto. Este pode ser um documento ou um URL.
<i>Location</i>	Uma expressão de sequência de caracteres especificando a localização dentro do URL especificado em <i>Target</i> para onde deve-se saltar. Caso <i>Location</i> não seja especificado, o salto será dado até o documento padrão.
<i>FrameName</i>	Uma expressão de sequência de caracteres especificando a moldura dentro do URL especificado em <i>Target</i> para onde deve-se saltar. Caso <i>FrameName</i> não seja especificado, o salto será dado até a moldura padrão.

### Comentários

Se o objeto está em um recipiente que suporte hyperlin OLE, o recipiente saltará até o local especificado. Caso o objeto esteja em um recipiente que não suporte hyperlin OLE, um aplicativo que seja registrado como suportando hyperlin é iniciado para tratar o pedido.

Caso *Target* não especifique uma localização válida, um erro é produzido.

## Propriedade ParentControls

Retorna uma coleção dos outros controles no recipiente do controle. A propriedade **ParentControls** não está disponível durante o tempo de criação do controle, e somente leitura durante o tempo de execução do controle.

### Sintaxe

*object.ParentControls*

A sintaxe da propriedade **ParentControls** tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Na maior parte dos casos, o recipiente do controle será um formulário; esta coleção funciona de maneira parecida como a coleção Controls no formulário, mas conterà também o próprio formulário.

Esta coleção é útil se o controle deseja executar alguma ação sobre os controles no formulário; ele pode iterar pela coleção.

Os controles não podem ser adicionados ou removidos pelo desenvolvedor que utiliza o controle nesta coleção; eles devem ser alterados de todas as maneiras permitidas pelo recipiente.

## Objeto PropertyBag

Um objeto **PropertyBag** contém informações que devem ser salvas e restauradas durante acionamentos de um objeto.

### Comentários

Um objeto **PropertyBag** é passado a um objeto por meio do evento **ReadProperties** e do evento **WriteProperties** para salvar e restaurar o estado do objeto. Usando os métodos do objeto **PropertyBag**, o objeto pode ler ou gravar suas próprias propriedades. O método **ReadProperty** do objeto **PropertyBag** é usado para ler um valor de uma propriedade, enquanto o método **WriteProperty** do objeto **PropertyBag** é usado para gravar o valor de uma propriedade. O valor de uma propriedade pode ser ele mesmo um objeto; neste caso o objeto **PropertyBag** tentará salvá-lo.

## Método PropertyChanged

Notifica o recipiente que o valor de uma propriedade foi alterado.

### Sintaxe

*object.PropertyChanged PropertyName*

A sintaxe do método **PropertyChanged** tem estas partes:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.
<i>PropertyName</i>	Uma expressão de seqüência de caracteres que representa um nome da propriedade cujo valor foi alterado pelo controle.

### Comentários

Notificando o recipiente de que o valor de uma propriedade foi alterado, o recipiente pode sincronizar sua janela de propriedade com os novos valores das propriedades do objeto. Além disso, o recipiente não saberia se uma ocorrência do objeto precisava ser salvo (através da produção de um evento **WriteProperties**) a menos que o recipiente fosse notificado de que o valor de uma propriedade fora alterado.

Este método precisa ser chamado, por exemplo, quando um usuário altera o valor de uma propriedade em uma página de propriedades, ou o próprio objeto altera o valor de uma propriedade. Este método também deve ser chamado quando uma propriedade acoplada a dados é modificada; caso contrário, a origem dos dados não será atualizada.

As propriedades que estão disponíveis somente durante o tempo de execução, não precisam chamar o método **PropertyChanged**, a menos que elas sejam acopladas a dados.

Como exemplo, o código abaixo mostra como é usado o método **PropertyChanged**:

```
Public Property Let Address(ByVal cValue As String)
    m_Address = cValue
    PropertyChanged "Address"
End Property
```

## Propriedade PropertyPages

Retorna ou define uma seqüência de caracteres que é o nome de uma página de propriedades que está associada a um controle.

### Sintaxe

*object.PropertyPages(index) [= PropPageName]*

A sintaxe da propriedade **PropertyPages** tem estas partes:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.
<i>index</i>	Índice de uma matriz de seqüências de caracteres.
<i>PropPageName</i>	Uma seqüência de caracteres contendo o nome de uma página de propriedades do projeto.

### Comentários

A propriedade **PropertyPages** é uma matriz de seqüências de caracteres contendo os nomes das páginas de propriedade no projeto que estão associadas a este controle. Uma página de propriedades pode ser adicionada à matriz definindo-se o último item na matriz (que está sempre va-

zia) Uma página de propriedades pode ser excluída da matriz definindo-se o elemento na matriz como uma seqüência de caracteres vazia.

O ordem dos nomes de páginas de propriedade na matriz determina a ordem em que as páginas aparecem na caixa de diálogo da página de propriedades do controle.

## Propriedade Public

Retorna ou define um valor que determina se um controle pode ser compartilhado com outros aplicativos. A propriedade **Public** é para leitura/gravação durante o tempo de criação do controle e não está disponível durante o tempo de execução.

### Configurações

As configurações para **Public** são:

Configuração	Descrição
<b>True</b>	O controle pode ser compartilhado com outros aplicativos. Este é o padrão para tipos de projeto ActiveX Control.
<b>False</b>	O controle não pode ser compartilhado com outros aplicativos. Quando o controle está contido em um projeto ActiveX Control, o controle não pode ser visualizado fora do projeto ActiveX Control. Isto significa que outros controles ou outros formulários no projeto podem usar o controle, mas aplicativos externos não podem. Este é o único valor válido para tipos de projeto que não o ActiveX control.

## Evento ReadProperties

Ocorre quando se carrega uma ocorrência antiga de um objeto que tem um estado salvo.

### Sintaxe

**Sub** *object\_ReadProperties(pb As PropertyBag)*

A sintaxe do evento **ReadProperties** tem estas partes:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.
<i>pb</i>	Um objeto do tipo classe PropertyBag que contém os dados salvos a serem carregados.

### Comentários

Quando ocorre este evento, o autor do objeto pode carregar no estado salvo de *pb*, chamando o método **ReadProperty** do objeto **PropertyBag** para cada valor que deve ser carregado. Este evento ocorre após o evento Initialize.

Sempre inclua interceptação de erros ao lidar com o evento ReadProperties, para proteger o controle contra valores de propriedade inválidos que possam ter sido inseridos por usuários ao editar o arquivo contendo os dados salvos por meio de processadores de texto. Entretanto, não se deve provocar um erro em um evento, uma vez que fazer isto pode ser fatal para o recipiente, assim qualquer interceptação de erro no procedimento de evento ReadProperties não deve incluir a provocação de erros.

## Método ReadProperty

Retorna um valor salvo de um objeto da classe **PropertyBag**.

### Sintaxe

*object.ReadProperty(DataName[, DefaultValue])*

A sintaxe do método **ReadProperty** tem estas partes:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.
<i>DataName</i>	Uma expressão de seqüência de caracteres que representa um valor de dados contido em PropertyBag.
<i>DefaultValue</i>	O valor a ser retornado se nenhum valor existir em PropertyBag.

---

## Comentários

O método **ReadProperty** retornará o valor dos dados salvos que são representados pela expressão de sequência de caracteres *DataName* ou *DefaultValue* se não existir valor salvo. *DataName* deve corresponder à expressão de sequência de caracteres que foi usada para armazenar o valor dos dados salvo na sacola de propriedades.

**Observação:** Especificar um valor padrão reduz o tamanho do arquivo pertencente ao recipiente do controle. Uma linha para a propriedade é escrita no arquivo somente se o valor a ser gravado for diferente do padrão. Onde possível, deve-se especificar valores padrão para as propriedades do controle ao inicializar, salvar e recuperar valores de propriedade.

## Propriedade RightToLeft

Retorna um valor booleano que indica a direção da exibição do texto e controla a aparência visual em um sistema bidirecional.

### Sintaxe

*object.RightToLeft*

A sintaxe da propriedade **RightToLeft** tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Configurações

Os valores booleanos de retorno da propriedade **RightToLeft** são:

Configuração	Descrição
<b>True</b>	O controle está sendo executado em uma plataforma bidirecional como, por exemplo do Windows 95 em Árabe ou Hebraico, e o texto está sendo executado da direita para a esquerda. O controle deve modificar seu comportamento, por exemplo, colocar barras de rolagem vertical no lado esquerdo de um texto ou caixa de listagem, colocar rótulo à direita das caixas de texto, etc.
<b>False</b>	O controle deve funcionar como se ele estivesse sendo executado em uma plataforma não-bidirecional como, por exemplo o Windows 95 em Inglês, e o texto está sendo executado da esquerda para a direita. Caso o recipiente não implemente esta propriedade de ambiente, este será o valor padrão.

## Propriedade ScaleUnits

Retorna um valor de sequência de caracteres que é o nome das unidades de coordenadas utilizadas pelo recipiente.

### Sintaxe

*object.ScaleUnits*

A sintaxe da propriedade **ScaleUnits** tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Esta sequência de caracteres representa as coordenadas usadas pelo recipiente do controle, por exemplo "twips". Esta sequência de caracteres pode ser usada pelo controle como um indicador de unidades ao se exibir valores de coordenada.

Caso o recipiente não implemente esta propriedade de ambiente, o valor padrão será uma sequência de caracteres vazia.

## Propriedade SelectedControls

Retorna uma coleção que contém todos os controles atualmente selecionados no formulário. A propriedade **SelectedControls** não está disponível durante o tempo de autoria da página de pro-

---

priedades, e é somente leitura durante o tempo de execução da página de propriedades.

### Sintaxe

*object*.**SelectedControls**

A sintaxe da propriedade **SelectedControls** tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Esta coleção é útil para que uma página de propriedades determine quais controles estão selecionados atualmente e, portanto, quais controles podem sofrer modificações em suas propriedades. Alguns recipientes somente permitem a seleção de um controle de cada vez; neste caso **SelectedControls** somente conterá um controle. Outros recipientes permitem que mais de um controle seja selecionado ao mesmo tempo; neste caso podem existir mais de um controle selecionado e a página de propriedades deve iterar pelos controles contidos na coleção **SelectedControls** e tentar definir as propriedades alteradas. Tratamento de erro adequado deve ser escrito para cuidar dos casos em que um determinado controle na coleção não sofre modificação em sua propriedade, ou quando o controle provoca um erro no momento da definição da propriedade.

## Evento SelectionChanged

Ocorre quando a seleção de controles no formulário sofre alteração.

### Sintaxe

**Sub** *object*.**SelectionChanged()**

A sintaxe do evento **SelectionChanged** tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

O disparo deste evento notifica a página de propriedades que a seleção de controles foi alterada e, portanto, a exibição dos valores de propriedade atuais pode precisar ser atualizada. A propriedade **SelectedControls** deve ler para encontrar o novo conjunto de controles selecionados.

O evento **SelectionChanged** também é provocado quando a página de propriedades é trazida pela primeira vez para um controle.

## Evento Show (Objeto UserControl)

Ocorre quando a propriedade **Visible** do objeto é alterada para **True**.

### Sintaxe

**Sub** *object*.**Show()**

A sintaxe do evento **Show** tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Para desenhar na tela no Windows, qualquer objeto deve ter uma janela, temporária ou permanente. Os controles ActiveX do Visual Basic têm janelas permanentes. Antes que um controle tenha sido colocado em um formulário, sua janela não está no recipiente. O controle recebe eventos **Show** quando a janela é adicionada.

Enquanto a janela do controle está no formulário, o controle recebe um evento **Show** quando a propriedade **Visible** é alterada para **True**.

O controle *não* recebe eventos **Show** se o formulário for ocultado e novamente exibido, ou se o formulário for minimizado e, em seguida, restaurado. A janela do controle permanece no formulário durante estas operações e sua propriedade **Visible** não sofre alteração.

Caso o controle esteja sendo mostrado em um navegador da Internet, ocorre um evento **Show** se o usuário retorna à página que contém o controle.

**Observação:** Caso o controle seja usado com versões anteriores do Visual Basic, que não a

5.0, o controle não receberá eventos Show durante o tempo de criação. Isto ocorre porque versões antigas do Visual Basic não colocam qualquer janela visível em um formulário durante o tempo de criação.

## Evento Show (Objeto UserDocument)

Ocorre quando a propriedade **Visible** do objeto se altera para **True**.

### Sintaxe

#### Sub *object*.Show()

A sintaxe do evento Show tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Para desenhar na tela no Windows, qualquer objeto deve ter uma janela, temporária ou permanente. Os documentos ActiveX do Visual Basic têm janelas permanentes. Antes que um *objeto* tenha sido colocado em um formulário, sua janela não está no recipiente. O objeto UserDocument recebe eventos Show quando a janela é adicionada.

Enquanto a janela do *objeto* está no recipiente, o *objeto* recebe um evento Show quando a propriedade **Visible** do *objeto* é alterada para **True**.

O *objeto* não recebe eventos Show se o recipiente é ocultado e novamente exibido, ou se o recipiente é minimizado e, em seguida, restaurado. A janela do *objeto* permanece no recipiente durante estas operações e sua propriedade **Visible** não sofre alteração.

Caso o *objeto* esteja sendo mostrado em um navegador da Internet, ocorre um evento Show quando o usuário navega até a página.

**Observação:** Caso o *objeto* seja usado com versões anteriores do Visual Basic, que não a 5.0, o *objeto* não receberá eventos Show durante o tempo de criação. Isto ocorre porque versões antigas do Visual Basic não colocam qualquer janela visível em um formulário durante o tempo de criação.

## Propriedade ShowGrabHandles

Retorna um valor booleano informando se o controle deve ou não conter alças de manipulação.

### Sintaxe

#### *object*.ShowGrabHandles

A sintaxe da propriedade **ShowGrabHandles** tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Configurações

O valores booleanos de retorno possíveis da propriedade **ShowGrabHandles** são:

Configuração	Descrição
<b>True</b>	O controle deve mostrar alças de manipulação, se necessário. Caso o recipiente não implemente esta propriedade de ambiente, este será o valor padrão.
<b>False</b>	O controle não deve mostrar alças de manipulação.

### Comentários

O comportamento padrão de um controle é mostrar automaticamente alças de manipulação quando o controle está em um recipiente que esteja em modo de criação (o modo de execução do controle). Entretanto, muitos recipientes não desejam que o controle exiba alças de manipulação, preferindo tratar a indicação de tamanho do controle de outra maneira. A propriedade **ShowGrabHandles** é a maneira como o recipiente notifica o controle de quem deve exibir as indicações de dimensionamento.

**Observação:** Todos os recipientes conhecidos preferem tratar eles mesmos da indicação de dimensionamento do controle e, portanto, defina a propriedade **ShowGrabHandles** como **False**. Provavelmente não é necessário tratar efetivamente do caso quando **ShowGrabHandles** é **True**.

---

## Propriedade ShowHatching

Retorna um valor booleano indicando se o controle deve exibir hachurado ao redor do controle.

### Sintaxe

*object*.ShowHatching

A sintaxe da propriedade **ShowHatching** tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Configurações

Os valores booleanos de retorno possíveis da propriedade **ShowHatching** são:

Configuração	Descrição
<b>True</b>	O controle deve mostrar marcas de hachura, se for necessário. Caso o recipiente não implemente esta propriedade de ambiente, este será o valor padrão.
<b>False</b>	O controle não deve mostrar marcas de hachura.

### Comentários

O comportamento padrão para um controle é mostrar automaticamente o hachurado quando o controle está em um recipiente que esteja no modo de criação (o modo de execução do controle) e o controle é que tem o foco. Entretanto, muitos recipientes não desejam que o controle mostre hachurado, preferindo tratar da indicação de foco de controle de outra forma. A propriedade **ShowHatching** é a maneira como o recipiente notifica o controle de quem deve exibir indicações de foco de controle.

**Observação:** Os formulários do Visual Basic não implementam esta propriedade de ambiente e, portanto, a propriedade **ShowHatching** é configurada com o valor padrão **True** quando o controle é colocado em um formulário Visual Basic. Entretanto, o Visual Basic não espera que o controle efetivamente faça coisa alguma em resposta ao valor **True** de **ShowHatching**, portanto não é necessário tratar efetivamente do caso quando **ShowHatching** for **True**.

## Propriedade SupportsMnemonics

Retorna um valor booleano informando se o recipiente do controle trata de teclas de acesso para o controle.

### Sintaxe

*object*.SupportsMnemonics

A sintaxe da propriedade **SupportsMnemonics** tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Configurações

Os valores booleanos de retorno possíveis da propriedade **SupportsMnemonics** são:

Configuração	Descrição
<b>True</b>	O recipiente do controle trata de teclas de acesso.
<b>False</b>	O recipiente do controle não trata de teclas de acesso. Caso o recipiente não implemente esta propriedade de ambiente, este será o valor padrão.

### Comentários

A maior parte dos recipientes de controles pode tratar de todos os processamentos de teclas de acesso para os controles contidos no recipiente. Isto inclui calcular quais controles devem receber uma determinada tecla de acesso. Caso um recipiente não possa processar teclas de acesso, isto é indicado por esta propriedade **SupportsMnemonics**, e o controle pode agir, por exemplo não exibindo o caractere de sublinhado como indicação de aceleradores de teclado.

## Propriedade TextAlign

Retorna um valor enumerado de tipo de TextAlignChoices informando qual tipo de alinhamento de texto o recipiente gostaria que fosse adotado pelo controle.

### Sintaxe

---

## *object.TextAlign*

A sintaxe da propriedade **TextAlign** tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Configurações

Os valores de retorno enumerados possíveis da propriedade **TextAlign** são:

Configuração	Descrição
<b>0-General</b>	Alinhamento geral: o texto à esquerda, números à direita. Caso o recipiente não implemente esta propriedade de ambiente, este será o valor padrão.
<b>1-Left</b>	Alinhar à esquerda.
<b>2-Center</b>	Centralizar.
<b>3-Right</b>	Alinhar à direita.
<b>4-FillJustify</b>	Preencher e justificar.

### Comentários

Esta propriedade de ambiente é a maneira como um recipiente comunica a um controle recipiente como executar a justificação; esta é uma dica do recipiente que o controle pode ou não aceitar.

## Propriedade **ToolboxBitmap**

Retorna ou define um bitmap que será usado como representação pictórica do controle na caixa de ferramentas. O tamanho do espaço para o bitmap na caixa de ferramentas é 16x15 pixels; o bitmap especificado por esta propriedade será escalado para estas dimensões se for necessário. A propriedade **ToolboxBitmap** é para leitura/gravação durante o tempo de criação do controle e não está disponível durante o tempo de execução do controle.

### Comentários

**Importante** Não atribua um ícone à propriedade **ToolboxBitmap**. Ícones não são bem escalados até o tamanho de bitmap Toolbox.

O Visual Basic usa automaticamente o nome da classe do controle como texto de dica de ferramenta quando os usuários passam com o ponteiro do mouse sobre o ícone na Barra de ferramentas.

**Dica** Ao criar bitmaps, lembre-se de que para muitas formas de daltonismo, as cores como o mesmo nível geral de brilho aparecerão como sendo a mesma. Pode-se evitar isto restringindo o bitmap a branco, preto e tons de cinza, ou através de uma cuidadosa seleção de cores.

## Propriedade **UIDead**

Retorna um valor booleano indicando se o controle deve ou não responder ao usuário.

### Sintaxe

*object.UIDead*

A sintaxe da propriedade **UIDead** tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Configurações

Os valores booleanos de retorno possíveis da propriedade **UIDead** são:

Configuração	Descrição
<b>True</b>	O controle não deve responder ao usuário.
<b>False</b>	O controle deve responder ao usuário. Caso o recipiente não implemente esta propriedade de ambiente, este será o valor padrão.

### Comentários

Esta propriedade é tipicamente usada para indicar que o recipiente está em modo de interrupção: durante este modo, o controle não deve responder a nenhuma entrada do usuário. Isto é, o con-



---

controle deve ignorar cliques de mouse e pressionamentos de tecla, e não altera o cursor do mouse mesmo quando o mouse encontra-se sobre a janela do controle. Um recipiente como, por exemplo, um formulário do Visual Basic definiria este sinalizador como TRUE quando o programador parasse o programa durante a execução — o recipiente não está no modo de criação, nem no modo de execução ainda; o Visual Basic simplesmente deseja que o controle esteja inoperante.

## Propriedade **UserMode**

Retorna um valor booleano indicando se o controle está sendo usado por um criador de formulário ou um usuário de formulário.

### Sintaxe

*object*.**UserMode**

A sintaxe da propriedade **UserMode** tem esta parte:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.

### Configurações

Os valores booleanos de retorno possíveis da propriedade **UserMode** são:

Configuração	Descrição
<b>True</b>	O controle está sendo atualmente usado por um usuário de formulário. Caso o recipiente não implemente esta propriedade de ambiente, este será o valor padrão. No Visual Basic, este é o modo de execução.
<b>False</b>	O controle está sendo usado atualmente por um criador de formulários (o desenvolvedor). No Visual Basic, este é o modo de criação.

## Evento **WriteProperties**

Ocorre quando uma ocorrência de um objeto deve ser salva. Este evento sinaliza ao objeto que o estado do objeto precisa ser salvo, de modo que o estado possa ser restaurado posteriormente. Na maior parte dos casos, o estado do objeto consiste apenas em valores de propriedade.

### Sintaxe

**Sub** *object*\_**WriteProperties**(*pb* As **PropertyBag**)

A sintaxe do evento **WriteProperties** tem estas partes:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.
<i>pb</i>	Um objeto do tipo da classe <b>PropertyBag</b> onde devem ser gravados os dados.

### Comentários

O autor de *object* pode fazer com que *object* salve o estado quando ocorrer o evento **WriteProperties**, chamando o método **WriteProperty** do objeto **PropertyBag** para cada valor que deva ser salvo.

**Observação:** A sacola de propriedade *pb* pode ser diferente da *pb* que foi passada ao evento **ReadProperties** mais recente.

O evento **WriteProperties** pode ocorrer múltiplas vezes durante a vida de uma ocorrência de *object*.

## Método **WriteProperty**

Grava um valor a ser salvo em um objeto da classe **PropertyBag**.

### Sintaxe

*object*.**WriteProperty**(*DataName*, *Value*[, *DefaultValue*])

A sintaxe do método **WriteProperty** tem estas partes:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista

---

<i>Applies To.</i>	
<i>DataMember</i>	Uma expressão de seqüência de caracteres para representar o valor de dados a ser colocado na sacola de propriedade.
<i>Value</i>	Os valores de dados a serem salvos na sacola de propriedade.
<i>DefaultValue</i>	O valor padrão dos dados.

### Comentários

O método **WriteProperty** gravará um valor de dados na sacola de propriedade, e o associará ao valor de seqüência de caracteres em *DataMember*. Este valor de seqüência de caracteres será utilizado para acessar o valor de dados quando o método **ReadProperty** é chamado para recuperar um valor de dados salvo da sacola de propriedade.

**Observação:** Especificar um valor padrão reduz o tamanho do arquivo pertencente ao recipiente do controle. Uma linha para a propriedade é escrita no arquivo somente se o valor a ser gravado for diferente do padrão. Sempre que possível, deve-se especificar valores padrão para as propriedades do controle ao inicializar, salvar e recuperar valores de propriedade.

## Objeto Picture

O objeto **Picture** permite manipular bitmaps, ícones, metarquivos imagens de metarquivos melhorados, GIF e JPEG atribuídas a objetos que tenham uma propriedade **Picture**.

### Sintaxe

#### Picture

### Comentários

Com freqüência, identifica-se um objeto **Picture** usando a propriedade **Picture** de um objeto que exiba elementos gráficos (por exemplo, um objeto **Form** ou um controle **PictureBox**). Caso tenha um controle **PictureBox** chamado Picture1, você pode configurar um objeto **Picture** de maneira igual a outro usando a instrução **Set**, como no exemplo abaixo:

```
Dim X As Picture
Set X = LoadPicture("PARTY.BMP")
Set Picture1.Picture = X
```

Pode-se usar uma matriz de objetos **Picture** para manter uma série de elementos gráficos na memória sem precisar de um formulário que contenha múltiplos controles **PictureBox** ou **Image**.

Pode-se criar um objeto **Picture** usando código como, por exemplo, `Dim X As New Picture`.

Caso queira criar um objeto **Picture**, você deve usar o objeto **StdPicture** desta forma:

```
Dim X As New StdPicture
```

## Propriedade Handle

Retorna um identificador ao elemento gráfico contido em um objeto **Picture**.

### Sintaxe

#### *object*.Handle

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista *Applies To*.

### Valor retornado

O valor retornado pela propriedade **Handle** depende da configuração atual da propriedade **Type** como mostra a tabela abaixo:

Propriedade Type	Valor retornado
1 (Bitmap)	Um identificador HBITMAP.
2 (Metarquivo)	Um identificador HMETAFILE.
3 (ícone)	Um identificador HICON ou HCURSOR.
4 (Metarquivo ampliado)	Um identificador HENHMETAFILE.

### Comentários

A propriedade **Handle** é útil quando se precisa passar um identificador a um elemento gráfico como parte de uma chamada de uma função em uma biblioteca de vínculos dinâmicos (DLL) da API Windows.

---

## Propriedade hPal

Retorna ou configura um identificador para a paleta de uma figura em um objeto **Picture**.

### Sintaxe

*object.hPal* [= *value*]

A sintaxe da propriedade **hPal** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	O identificador para a paleta da figura (HPAL).

### Comentários

A propriedade **hPal** é útil quando se precisa passar um identificador a uma paleta como parte de uma chamada a uma função em uma biblioteca de vínculos dinâmicos (DLL) ou à API Windows.

## Método Render

Desenha toda ou parte de uma imagem de origem a um objeto de destino.

### Sintaxe

*object.Render(hdc, xdest, ydest, destwid, desthgt, xsrc, ysrc, srcwid, srchgt, wbounds)*

A sintaxe do método **Render** tem estas partes:

Parte	Descrição
<i>object</i>	Obrigatório. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>hdc</i>	Obrigatório. O identificador do contexto do dispositivo do objeto de destino.
<i>xdest</i>	Obrigatório. A coordenada de x do canto superior esquerdo da área de desenho no objeto de destino. Esta coordenada está na unidade de escala do objeto de destino.
<i>ydest</i>	Obrigatório. A coordenada de y do canto superior esquerdo da área de desenho no objeto de destino. Esta coordenada está na unidade de escala do objeto de destino.
<i>destwid</i>	Obrigatório. A largura da área de desenho no objeto de destino, expressa nas unidades de escala do objeto de destino.
<i>desthgt</i>	Obrigatório. A altura da área de desenho no objeto de destino, expressa nas unidades de escala do objeto de destino.
<i>xsrc</i>	Obrigatório. A coordenada de x do canto superior esquerdo da área de desenho no objeto de origem. Esta coordenada está em unidades HIMETRIC.
<i>ysrc</i>	Obrigatório. A coordenada de y do canto superior esquerdo da área de desenho no objeto de origem. Esta coordenada está em unidades HIMETRIC.
<i>srcwid</i>	Obrigatório. A largura da área de desenho no objeto de origem, expressa em unidades HIMETRIC.
<i>srchgt</i>	Obrigatório. A altura da área de desenho no objeto de origem, expressa em unidades HIMETRIC.
<i>wbounds</i>	Obrigatório. Os limites de palavra em um <u>metarquivo</u> . Este argumento deve ser passado como um valor Null, a menos que se esteja desenhando em um metarquivo, caso em que o argumento é passado como um tipo definido pelo usuário correspondente a uma estrutura RECTL.

### Comentários

A maneira recomendada de se pintar parte de um elemento gráfico em um destino é através do método **PaintPicture**.

## Barra de ferramentas Add-In

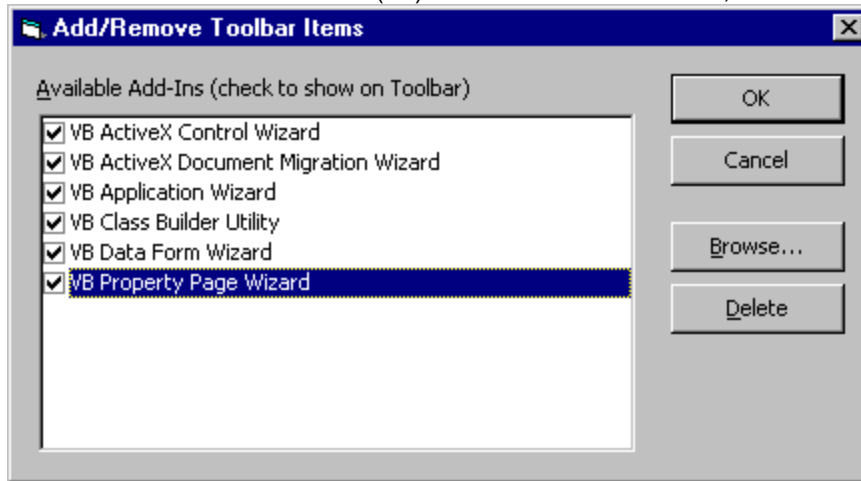


Uma barra de ferramentas onde podem ser colocados os suplementos e assistentes para acesso fácil e rápido pelo usuário. Para iniciar um suplemento ou assistente, clique simplesmente em seu

ícone na barra de ferramentas.

Os suplementos e assistentes colocados na barra de ferramentas **Add-In** não serão ativados até que seu botão seja clicado. A barra de ferramentas **Add-In** elimina a necessidade de se ativar o suplemento através da caixa de diálogo **Add-In Manager**.

Pode-se adicionar assistentes e suplementos à barra de ferramentas **Add-In** através do botão **Add/Remove Toolbar Items (+/-)**. Ao se clicar neste botão, obtém-se a caixa de diálogo abaixo:



Para adicionar um suplemento ou assistente à lista de suplementos disponíveis, clique no botão **Browse**. Aponte para um suplemento ou arquivo .Exe ou .Dll de assistente na caixa de diálogo e, em seguida, clique em **Open**. Ele deve aparecer na lista **Available Add-Ins**. Entretanto, ele não aparecerá na barra de ferramentas **Add-In**, a menos que sua caixa esteja selecionada na lista **Available Add-Ins**.

O botão **O** fecha a caixa de diálogo **Add/Remove Toolbar Items** e atualiza a barra de ferramentas **Add-In** contendo os itens selecionados.

O botão **Cancel** fecha a caixa de diálogo **Add/Remove Toolbar Items** e ignora qualquer modificação efetuada enquanto ela estava aberta.

Quando o botão **Delete** é clicado, o suplemento ou assistente atualmente selecionado é removido da lista **Available Add-Ins**. Observe que isto não provoca a remoção do suplemento ou assistente do sistema, nem suas referências na caixa de diálogo **Add-In Manager**. O botão **Delete** remove somente a entrada na lista **Available Add-Ins** da barra de ferramentas **Add-In**.

## Propriedade AddIns

Retorna uma coleção de quais suplementos podem ser usados para registrar seus componentes de automação no modelo de objeto de extensibilidade.

### Sintaxe

*object*.AddIns

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

## Método AddToaddinToolbar

Insere um botão na barra de ferramentas **Add-In** que se refere a um suplemento ou assistente.

### Sintaxe

*object*.AddToaddinToolbar (*sfilename* As String, *sprogid* As String, *showontoolbar* As Boolean, *forceaddintoolbar* As Boolean)

Parte	Descrição
<i>Object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>sfilename</i>	Obrigatório. Uma <u>expressão de sequência de caracteres</u> especificando o caminho até o suplemento ou assistente e o nome de seu arquivo .Exe ou .Dll.
<i>Sprogid</i>	Obrigatório. Uma <u>expressão de sequência de caracteres</u> especificando a ID programática (ProgID) do suplemento ou assistente.
<i>saddinname</i>	Obrigatório. Uma <u>expressão de sequência de ca-</u>

	<u>racteres</u> especificando o título do suplemento ou assistente.
<i>Showontoolbar</i>	Obrigatório. Uma <u>expressão Booleana</u> especificando se o suplemento ou assistente referido aparecerá na barra de ferramentas <b>Add-In</b> . <b>True</b> = sim, <b>False</b> = não.
<i>forceaddintoolbar</i>	Obrigatório. Uma <u>expressão Booleana</u> especificando se uma barra de ferramentas <b>Add-In</b> é automaticamente exibida na próxima vez em que o Visual Basic for inicializado. <b>True</b> = sim, <b>False</b> = não.

## Exemplo de método AddToaddinToolbar

Este exemplo utiliza o método **AddToaddinToolbar** para adicionar um botão à barra de ferramentas para um suplemento fictício chamado MyAdd.Dll. Configurando-se ForceAddInToolbar como **True** assegura-se que a barra de ferramentas **Add-In** será carregada na próxima vez em que o Visual Basic for inicializado.

Pode-se modificar o código abaixo em um pequeno aplicativo Visual Basic para servir como Instalação de um suplemento.

```
Sub Main()
    dim x as Object
    Set x=CreateObject("AddInToolbar.Manager")
    x.AddToaddinToolbar sFileName:="C:\VB5\MyAdd.DLL", _
        sProgID:="MyAddIn.Connect", _
        sAddInName:="MyAddIn Title" _
        ShowOnToolBar:=True, _
        ForceAddInToolbar:=True
End Sub
```

## Objeto AsyncProperty

O objeto **AsyncProperty** é passado ao evento AsyncReadComplete e contém os resultados do método **AsyncRead**.

## Propriedade AsyncType

Retorna ou define o tipo de dado retornado pela propriedade **Value**. Esta propriedade somente está disponível como um argumento do método **AsyncRead**.

### Sintaxe

*object.AsyncType* = *dataType*

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>dataType</i>	Um inteiro especificando o tipo de dados, como mostra em Configurações abaixo.

### Configurações

As configurações de *dataType* são:

Constante	Valor	Descrição
<b>vbAsyncTypePicture</b>	0	Padrão. Objeto Picture.
<b>VbAsyncTypeFile</b>	1	Os dados são fornecidos em um arquivo criado pelo Visual Basic.
<b>VbAsyncTypeByteArray</b>	2	Os dados são fornecidos como uma matriz de bytes que contém os dados recuperados.

## Propriedade Bindable

Retorna ou define a propriedade **Bindable** associada a um objeto **Member**.

### Sintaxe

*object.Bindable*

---

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

## Propriedade Browsable

Retorna ou define o atributo Browsable associado a um objeto **Member**.

### Sintaxe

*object*.**Browsable**

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

## Propriedade Category

Retorna ou define o atributo Category associado a um objeto **Member**.

### Sintaxe

*object*.**Category**

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

## Propriedade Connect

Retorna ou define o estado conectado de um suplemento.

### Sintaxe

*object*.**Connect**

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Retorna **True** se o suplemento estiver registrado e atualmente conectado (ativo).

Retorna **False** se o suplemento estiver registrado e não se encontrar conectado (inativo).

## Coleção ContainedVBControls

A coleção ContainedVBControls representa uma coleção de objetos **VBControl**.

## Propriedade ContinuousScroll

Retorna ou define um valor que determina se a rolagem é contínua, ou se **UserDocument** somente regenera quando o pressionamento de rolagem é liberado.

### Sintaxe

*object*.**ContinuousScroll** = *boolean*

Parte	Descrição
<i>Object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> que especifica se a rolagem é contínua ou não.

### Configurações

As configurações para boolean são:

Configuração	Descrição
<b>True</b>	Padrão. A rolagem é contínua.
<b>False</b>	UserDocument regenera somente quando o pressionamento é liberado.

## Propriedade Controls

Retorna uma referência a uma coleção de objetos **Control**.

### Sintaxe

*object*.**Controls**

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Pode-se manipular objetos **Control** usando a referência retornada pela propriedade **Controls**

---

## Objeto DataBinding

O objeto **DataBinding** representa uma propriedade acoplável de um componente.

### Sintaxe

#### DataBinding

### Comentários

Existe um objeto **DataBinding** para cada propriedade de um componente marcado como Bindable na caixa de diálogo **Procedure Attributes**.

O Visual Basic versão 4.0 suportava apenas uma propriedade de acoplamento de um controle a um banco de dados de cada vez. Entretanto, o Visual Basic 5.0 oferece a possibilidade de acoplar múltiplas propriedades de um controle a um banco de dados. Isto é mais comumente usado com controles **User**. Para maiores informações sobre isto, consulte o capítulo 9, "Criando componentes ActiveX" no *Guia de Ferramentas Componentes*.

## Propriedade DefaultBind

Retorna ou define o atributo DefaultBind de um objeto **Member**.

### Sintaxe

#### object.DefaultBind

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

## Propriedade DisplayBind

Retorna ou define o atributo DisplayBind de um objeto **Member**.

### Sintaxe

#### object.DisplayBind

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

## Propriedade Events

Fornecer propriedades que permitem aos suplementos conectarem-se a todos os eventos em Visual Basic for Applications.

### Sintaxe

#### object.Events

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

## Propriedade Hidden

Retorna ou define o atributo Hidden de um objeto **Member**.

### Sintaxe

#### object.Hidden

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

## Propriedades HScrollSmallChange, VScrollSmallChange

Retorna ou define a distância que **UserDocument** rolará quando o usuário clicar na seta de rolagem.

### Sintaxe

*object.HScrollSmallChange* = *single*

*object.VScrollSmallChange* = *single*

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>single</i>	A distância em twips que o <b>UserDocument</b> rolará quando o usuário clicar na seta de rolagem.

### Comentários

Não existe uma propriedade "LargeChange" ao contrário das propriedades **HScrollSmallChange**

---

e **VScrollSmallChange**. O "LargeChange" é determinado pelas propriedades **ViewPortHeight** e **ViewPortWidth** do objeto **ViewPort**.

## Objeto Hyperlin

Usando as propriedades e métodos do objeto **Hyperlin**, seu documento ActiveX ou controle ActiveX pode-se pedir um recipiente que reconheça hyperlins como, por exemplo o Microsoft Internet Explorer, para saltar até um determinado URL.

### Comentários

Use o método **NavigateTo** para saltar até um URL. Por exemplo, o código abaixo pressupõe que exista um documento ActiveX chamado "axdMyDoc":

```
UserDocument.Hyperlin.NavigateTo _  
"c:\mydocs\axdmydoc.vbd"
```

Caso seu documento ActiveX esteja contido em um recipiente que reconheça hyperlins (como, por exemplo, o Internet Explorer) e o recipiente mantenha um histórico dos documentos, utilize os métodos **GoBac** ou **GoForward** para avançar ou recuar na lista. Entretanto, certifique-se de usar verificação de erros, conforme mostra o exemplo abaixo:

```
Private Sub cmdGoForward_Click()  
    On Error GoTo noDocInHistory  
    UserDocument.Hyperlin.GoForward  
    Exit Sub  
noDocInHistory:  
    Resume Next  
End Sub
```

## Propriedade Hyperlin

Retorna uma referência ao objeto **Hyperlin**.

### Sintaxe

*object*.**Hyperlin**

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

## Propriedade IsDirty

Retorna um valor indicando se este componente foi editado desde a última vez em que foi salvo.

### Sintaxe

*object*.**IsDirty**

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

## Evento ItemReloaded

Ocorre após um componente ter sido recarregado.

### Sintaxe

**Private Sub** *object*\_**ItemReloaded**(*vbcomponent* **As VBComponent**)

A sintaxe do evento ItemReloaded tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>vbcomponent</i>	Um objeto <b>VBComponent</b> representando o componente que foi recarregado.

## Propriedade Lines

Retorna uma seqüência de caracteres contendo o bloco de linhas especificado.

### Sintaxe

*object*.**Lines**(*startline* **As Long**, *count* **As Long**)

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto



	na lista Applies To.
<i>startline</i>	Um <u>tipo de dados Long</u> especificando o número da linha onde se deve começar.
<i>count</i>	Um <u>tipo de dados Long</u> especificando o número de linhas a serem selecionadas.

## Método LogEvent

Registra um evento no alvo de log do aplicativo. Em plataformas Windows NT, o método grava no Log de eventos do NT. Em plataformas Windows 95, o método grava no arquivo especificado na propriedade **LogPath**; como padrão, se nenhum arquivo for especificado, os eventos serão gravados em um arquivo chamado `vbevents`.

### Sintaxe

*object*.LogEvent (*logBuffer*, *eventType*)

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>logBuffer</i>	Obrigatório. Sequências de caracteres a serem gravadas no Registro.
<i>eventType</i>	Opcional. Um inteiro Long que especifica o tipo de eventos, como mostrado em Configurações.

### Configurações

As configurações para *eventType* são:

Constante	Valor	Descrição
<code>vbLogEventTypeError</code>	1	Erro.
<code>vbLogEventTypeWarning</code>	2	Aviso.
<code>vbLogEventTypeInformation</code>	4	Informação.

### Comentários

As diretrizes para o processo de log estão disponíveis no Win32 SD, e estas diretrizes devem ser seguidas ao se registrar no Log de eventos do NT ou ao arquivo especificado na propriedade **LogPath** (em plataformas Windows 95).

## Propriedade LogMode

Retorna um valor que determina como o log (através do método **LogEvent**) será executado. Somente leitura durante o tempo de execução.

### Sintaxe

*object*.LogMode = *mode*

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>mode</i>	Longo. Determina o método de conexão, como mostra em Configurações abaixo.

### Configurações

As configurações de *mode* são:

Constante	Valor	Descrição
<b>vbLogAuto</b>	0	Se estiver sendo executada no Windows 95, esta opção registra mensagens no arquivo especificado na propriedade <b>LogFile</b> . Se estiver sendo executado no Windows NT, as mensagens são registradas no Log de eventos do NT com a sequência de caracteres App.Title usada como fonte do aplicativo.
<b>VbLogOff</b>	1	Desativa todo o processo de log. As mensagens de partes da IU, assim como do método <b>LogEvent</b> são ignoradas e descartadas.
<b>VbLogToFile</b>	2	Força o log a um arquivo. Se não existir re-

		nhum nome de arquivo válido em <b>LogPath</b> , o log é ignorado e a propriedade é definida como <b>vbLogOff</b> .
<b>VbLogToNT</b>	3	Força o log no Log de eventos do NT. Se não estiver sendo executado no Windows NT, o Log de eventos não estará disponível, o log é ignorado e a propriedade é definida como <b>vbLogOff</b> .
<b>VbLogOverwrite</b>	0x10	Indica que o arquivo de log deve ser recriado toda vez que o aplicativo for inicializado. Este valor pode ser combinado com outras opções de modo usando o operador OR. A ação padrão para o log é anexar ao arquivo existente. No caso de Log de eventos do NT, este sinalizador não tem sentido.
<b>VbLogThreadID</b>	0x20	Indica que a identificação de segmento atual seja pré-anexada à mensagem na forma "[T:0nnn] ". Este valor pode ser combinado a outras opções de modo usando o operador OR. A ação padrão é mostrar a identificação de segmento somente quando o aplicativo é multisegmentado (explicitamente marcado como de segmento seguro, ou implementado como um aplicativo de segmentos múltiplos implícito como, por exemplo, um servidor local com a propriedade de exemplificação definida como mono uso, de segmentos múltiplos).

#### Tipo de retorno

Long

## Propriedade LogPath

Retorna o caminho e nome de arquivo do arquivo usado para capturar o resultado do método **LogEvent**. Não disponível durante o tempo de criação; somente leitura durante o tempo de execução.

#### Sintaxe

*object*.LogPath = *path*

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>path</i>	Seqüência de caracteres. O caminho e nome de arquivo de um arquivo de log.

#### Comentários

A propriedade **LogMode** determina como será executado o log. Se nenhum **LogPath** for definido, o método **LogEvent** grava no arquivo NT LogEvent do NT.

## Propriedade Members

Contém identificadores que têm alcance a nível de módulo e que podem ser considerados propriedades, métodos ou eventos do objeto **CodeModule** especificado.

#### Sintaxe

*object*.Members

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

## Propriedades MinHeight, MinWidth

Retorna ou define a altura ou largura mínima de Viewport onde apareçam barras de rolagem no recipiente.

---

### Sintaxe

*object.MinHeight = single*

*object.MinWidth = single*

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>single</i>	A altura ou largura de um UserDocument onde aparecerão barras de rolagem em um recipiente.

### Comentários

Os valores padrão das propriedades **MinHeight** e **MinWidth** são definidos pelas propriedades **Height** e **Width** de **UserDocument**.

**MinWidth** e **MinHeight** não têm qualquer efeito se a propriedade **ScrollBars** for definida como **False**.

## Propriedade Moveable

Retorna ou define um valor que especifica se o objeto pode ser movido.

### Sintaxe

*object.Moveable = boolean*

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> que especifica se o objeto pode ser movido.

### Configurações

As configurações de *boolean* são:

Constante	Valor	Descrição
<b>True</b>	-1	O objeto pode ser movido.
<b>False</b>	0	O objeto não pode ser movido.

## Propriedade NonModalAllowed

Retorna um valor que indica se um formulário pode ser mostrado de maneira não-modal (sem janela restrita). Não disponível durante o tempo de criação.

### Sintaxe

*object.nonModalAllowed*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Tipo de retorno

Boolean

## Propriedade Palette

Retorna ou define uma imagem que contém a paleta a ser usada para o controle.

### Sintaxe

*object.Palette = path*

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>path</i>	O caminho da imagem de bitmap contendo a paleta a ser usada.

### Comentários

Pode-se usar um arquivo .dib, .gif, ou .pal para definir a paleta, assim como arquivos .bmp.

## Propriedade PaletteMode

Retorna ou define um valor que determina qual paleta a ser usada para os controles em um objeto.

## Sintaxe

*object.PaletteMode* = *integer*

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.
<i>integer</i>	Determina o modo de paleta a ser usado, conforme descrito em Configurações, abaixo.

## Configurações

As configurações para *integer* são:

Constante	Valor	Descrição
<b>vbPaletteModeHalfTone</b>	0	(Padrão) Usa a paleta Halftone.
<b>VbPaletteModeUseZOrder</b>	1	Usa a paleta do controle em primeiro plano que tenha uma paleta.
<b>VbPaletteModeCustom</b>	2	Usa a paleta especificada na propriedade <b>Palette</b> .
<b>VbPaletteModeContainer</b>	3	Usa a paleta do recipiente que suporta a propriedade <b>Palette</b> do ambiente. Aplica-se somente a UserControls.
<b>VbPaletteModeNone</b>	4	Não usa paleta alguma. Aplica-se somente a UserControls.
<b>VbPaletteModeObject</b>	5	Usa a paleta do criador ActiveX (aplica-se somente a criadores ActiveX que contenham uma paleta.)

## Comentários

Caso nenhuma paleta esteja disponível, a paleta de meio-tom torna-se a paleta padrão.

**Observação:** Em versões anteriores do Visual Basic, **PaletteMode** correspondia a **UseZOrder**.

## Propriedade **PropertyName**

O comportamento da propriedade **PropertyName** depende do contexto onde ela está sendo usada.

- Método **AsyncRead** — Define o nome da propriedade que será associada à propriedade **Value** do objeto **AsyncProperty**.
- Evento **AsyncReadComplete** — Especifica o nome da propriedade que está sendo lida atualmente. Este deve corresponder a um nome designado ao objeto **AsyncProperty** ao chamar o método **AsyncRead**.
- **DataBinding Object** — Somente leitura. Retorna o nome da propriedade a que se refere o objeto **DataBinding**.

## Sintaxe

*object.PropertyName* = *string*

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.
<i>string</i>	O nome de uma propriedade a ser salva ou recuperada.

## Exemplo de propriedade **PropertyName**

O exemplo atribui um valor à propriedade **PropertyName** no método **AsyncRead**. O mesmo valor será usado para designar o resultado do método ao controle **PictureBox**. Para experimentar o exemplo, coloque o controle **PictureBox** em um objeto **UserDocument**. Cole o código na seção General, e pressione F5 para executar. Inicie o Internet Explorer 3.0 (ou mais recente) e digite o caminho e nome até o arquivo **UserDocument.vbd** na caixa **Address**.

```
Private Sub UserDocument_InitProperties()  
    Dim strPath As String  
    ' Definir a variável como um caminho válido para um  
    ' bitmap em seu computador.  
    strPath = "C:\Program Files\DevStudio\VB\" & _  
        "Samples\PGuide\VCR\Bfly1.bmp"
```

---

```

    AsyncRead strPath, vbAsyncTypeFile, _
        PropertyName:= "butterfly"
End Sub

Private Sub UserDocument_AsyncReadComplete (AsyncProp _
    As AsyncProperty)
    ' Usar a instrução Select para determinar qual
    ' propriedade está sendo retornada.
    Select Case AsyncProp.PropertyName
        Case "butterfly"
            Picture1.Picture = _
                LoadPicture(AsyncProp.Value)
    End Select
End Sub

```

## Objeto PropertyPage

O objeto básico usado para criar uma página de propriedade ActiveX.

### Comentários

As páginas de propriedade oferecem uma alternativa à janela **Properties** para visualização de propriedades. Pode-se agrupar diversas propriedades relacionadas em uma página, ou usar uma página para oferecer uma interface do tipo caixa de diálogo para uma propriedade complexa demais para a janela **Properties**. Um objeto **PropertyPage** representa uma página, que quer dizer uma guia na caixa de diálogo **Property Pages**.

## Propriedade PropertyPage

Retorna ou define o atributo PropertyPage de um objeto **Member**.

### Sintaxe

*object*.**PropertyPage**

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

## Método Quit (suplementos)

Tenta sair do Visual Basic.

### Sintaxe

*object*.**Quit**

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

## Método RemoveAddInFromToolbar

Remove um botão da barra de ferramentas **Add-In** que se refere a um suplemento ou assistente.

### Sintaxe

*object*.**RemoveAddInFromToolbar** (*saddinname* As String)

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>saddinname</i>	Obrigatório. Uma <u>expressão de sequência de caracteres</u> especificando o nome do suplemento ou assistente a ser removido da barra de ferramentas <b>Add-In</b> (conforme especificado pelo parâmetro <i>saddinname</i> do método <b>AddToAddInToolbar</b> ).

## Exemplo de método RemoveAddInFromToolbar

Este exemplo remove um botão existente da barra de ferramentas **Add-In** que se refere a um suplemento fictício chamado MyAddIn Title:

```

Sub Main()
    dim x as Object
    Set x=CreateObject("AddInToolbar.Manager")
    x.RemoveAddInFromToolbar sAddInName:="MyAddIn Title"
End Sub

```

---

## Propriedade RequestEdit

Retorna ou define o atributo RequestEdit de um objeto **Member**.

### Sintaxe

*object*.RequestEdit

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

## Método SelectAll

Seleciona todos os controles contidos em um formulário.

### Sintaxe

*object*.SelectAll

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

## Método SetViewport

Define as coordenadas esquerda e superior do **UserDocument** que estará visível no Viewport.

### Sintaxe

*object*.SetViewPort *left*, *top*

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>left</i>	Obrigatório. Um valor do tipo Single que especifica a coordenada esquerda do UserDocument.
<i>top</i>	Obrigatório. Um valor do tipo Single que especifica a coordenada superior do UserDocument.

## Exemplo do método SetViewport

O exemplo utiliza o método **SetViewport** para colocar automaticamente o controle **TextBox** com o foco no canto superior esquerdo do Viewport do recipiente. Para experimentar o exemplo, coloque uma matriz de três ou mais controles **TextBox** em um objeto **UserDocument**. Cole o código abaixo da seção General. Pressione F5 para executar o projeto e, em seguida, execute o Internet Explorer (3.0 ou mais recente). No Internet Explorer, digite o caminho e nome de arquivo do documento ActiveX (UserDocument1.vbd) na caixa **Address** (o arquivo estará no mesmo diretório que o executável Visual Basic). Quando o documento ActiveX for exibido, digite qualquer texto distintivo no primeiro controle **TextBox**. Pressione TAB para mover até o próximo controle, para ver o efeito do método **SetViewPort**.

```
Private Sub Text1_GotFocus(Index As Integer)
    UserDocument.SetViewPort Text1(Index).Left, _
        Text1(Index).Top
End Sub

Private Sub UserDocument_Initialize()
    ' O recipiente deve ser grande o suficiente para
    ' que apareçam barras de rolagem. Para assegurar
    ' isto, defina as propriedades MinHeight e
    ' MinWidth como maiores que o recipiente.
    UserDocument.MinHeight = 10000
    UserDocument.MinWidth = 10000
End Sub
```

## Método Size

Altera a largura e altura de um objeto **UserControl**.

### Sintaxe

---

*object*.**Size** *width, height*

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>width</i>	Obrigatório. A largura do objeto em twips.
<i>height</i>	Obrigatório. A altura do objeto em twips.

#### Comentários

As propriedades **Width** e **Height** de um objeto **UserControl** são sempre fornecidas em Twips, independente de **ScaleMode**.

## Propriedade StandardMethod

Retorna ou define o atributo StandardMethod de um objeto **Member**.

#### Sintaxe

*object*.**StandardMethod**

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

## Método StartLogging

Define o alvo e modo de log de uma operação.

#### Sintaxe

*object*.**StartLogging** *logTarget, logMode*

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>logTarget</i>	Caminho e nome de arquivo do arquivo usado para capturar o resultado do método <b>LogEvent</b> .
<i>logMode</i>	Um valor que determina como o processo de log (através do método <b>LogEvent</b> ) será executado. Ver Configurações abaixo.

#### Configurações

As configurações para *logMode* são:

Constante	Valor	Descrição
<b>vbLogAuto</b>	0	Caso esteja sendo executado no Windows 95, esta opção registra as mensagens nos arquivos especificados na propriedade <b>LogFile</b> . Caso esteja sendo executado no Windows NT, as mensagens são registradas no log de eventos do aplicativo NT, com a sequência de caracteres App.Title usada como a origem do aplicativo.
<b>VbLogOff</b>	1	Desativa todo o processo de log. As mensagens de partes da IU, assim como do método <b>LogEvent</b> são ignoradas e descartadas.
<b>VbLogToFile</b>	2	Força o log em um arquivo. Se nenhum nome de arquivo válido existir em <b>LogPath</b> , o processo de log é ignorado, e a propriedade é definida como <b>vbLogOff</b> .
<b>VbLogToNT</b>	3	Força o processo de log no Log de eventos do NT. Se não estiver sendo executado no Windows NT, ou se o Log de eventos não estiver disponível, o processo de log é ignorado e a propriedade é definida como <b>vbLogOff</b> .
<b>VbLogOverwrite</b>	0x10	Indica que o arquivo de log deve ser recriado toda vez que o aplicativo for inicializado. Este valor pode ser combinado com outras opções de modo usando o operador <b>OR</b> . A ação padrão para a rotina de registro é anexar ao arquivo já existente. No caso de Log de eventos

**VbLogThreadID** 0x20

do NT, este sinalizador não tem significado algum.  
Indica que a identificação de segmento atual deve ser pré-anexada à mensagem, no formato "[T:0nnn] ". Este valor pode ser combinado com outras opções de modo usando o operador **OR**. A ação padrão é mostrar a identificação de segmento somente quando o aplicativo é multisegmentado (explicitamente marcado como segmentado de forma segura, ou implementado como um aplicativo de segmentos múltiplos implícito como, por exemplo, um servidor local com uma propriedade de exemplificação definida como mono uso, de segmentos múltiplos).

## Propriedade ThreadID

Retorna a Win32 ID do segmento em execução. (Usado para chamadas da API Win32.)

### Sintaxe

*object.ThreadID*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Tipo de retorno

Long

## Propriedade UIDefault

Retorna ou define o atributo UIDefault do objeto **Member**.

### Sintaxe

*object.UIDefault*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

## Propriedade UnattendedApp

Retorna um valor que determina se um aplicativo será executado sem qualquer interface de usuário.

### Sintaxe

*object.UnattendedApp*= *boolean*

A sintaxe **UnattendedApp** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> que especifica se o aplicativo será executado sem qualquer interface de usuário.

### Configurações

As configurações para *boolean* são:

Constante	Valor	Descrição
<b>True</b>	-1	O aplicativo não tem interface de usuário.
<b>False</b>	0	O aplicativo tem uma interface de usuário.

## Objeto UserControl

O objeto **UserControl** é o objeto básico usado para criar um controle ActiveX.

### Comentários

Um controle ActiveX criado com o Visual Basic é sempre constituído de um objeto **UserControl** mais quaisquer controles — referidos como controles constitutivos — escolhidos para serem colocados no **UserControl**.

Como os formulários do Visual Basic, objetos **UserControl** têm módulos de código e criadores v-



---

suais. Coloque os controles constituintes no criador de objeto **UserControl**, exatamente como colocaria controles em um formulário.

## Objeto UserDocument

Base de um documento ActiveX, o objeto **UserDocument** se parece com um objeto **Form** Visual Basic padrão com algumas exceções.

### Comentários

O objeto **UserDocument** tem a maioria, mas nem todos, dos eventos que são encontrados em um objeto **Form**. Os eventos presentes em um **Form** que não são encontrados no **UserDocument** incluem: Activate, Deactivate, LinClose, LinError, LinExecute, LinOpen, Load, QueryUnload e Unload.

Os eventos presentes no **UserDocument**, mas não encontrados em um objeto **Form** incluem: AsyncReadComplete, EnterFocus, ExitFocus, Hide, InitProperties, ReadProperties, Scroll, Show e WriteProperties.

Não se pode colocar objetos incorporados (por exemplo, um documento Excel ou Word) ou um controle **OLE Container** em um **UserDocument**.

## Propriedade VBProjects

Retorna a coleção VBProjects que representa todos os projetos atualmente abertos no IDE do Visual Basic.

### Sintaxe

*object.VBProjects*

O espaço reservado *object* representa um expressão de objeto que avalia para um objeto na lista Applies To.

## Propriedades ViewportHeight, ViewportLeft, ViewportTop, ViewportWidth

Retorna o valor da altura, esquerda, superior ou largura do Viewport.

### Sintaxe

*object.ViewportHeight*

*object.ViewportLeft*

*object.ViewportTop*

*object.ViewportWidth*

A espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Tipo de retorno

Single

### Comentários

O aplicativo usado para visualizar o documento ActiveX controla o tamanho do Viewport. Entretanto, pode-se usar as propriedades **MinHeight** e **MinWidth** para redimensionar o **UserDocument**. Por exemplo, o código abaixo redimensiona um controle **PictureBox**, de acordo com o tamanho das propriedades de largura e altura da esquerda do Viewport.

```
Private Sub UserDocument_Resize()  
    Picture1.Width = UserDocument.ViewportWidth - _  
        Picture1.Left  
    Picture1.Height = UserDocument.ViewportHeight - _  
        Picture1.Top  
End Sub
```

## Propriedade Windows

Retorna o objeto **Window**, que representa uma janela no IDE do Visual Basic.

### Sintaxe

*object.Window*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

---

## Propriedade StartUpPosition

Retorna ou define um valor especificando a posição de um objeto quando aparece pela primeira vez.

### Sintaxe

*object*.**StartUpPosition** = *position*

A sintaxe da propriedade **StartUpPosition** tem estas partes:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.
<i>StartUpPosition</i>	Um inteiro que especifica a posição do objeto, como mostrado em Configurações.

### Configurações

Pode-se usar uma das quatro configurações para **StartUpPosition**:

Constante	Valor	Descrição
<b>vbStartUpManual</b>	0	Nenhuma configuração inicial especificada.
<b>vbStartUpOwner</b>	1	Centraliza no item a que pertence o <b>UserForm</b> .
<b>vbStartUpScreen</b>	2	Centraliza na tela inteira.
<b>vbStartUpWindowsDefault</b>	3	Posiciona no canto superior esquerdo da tela.

### Comentários

Pode-se definir a propriedade **StartUpPosition** por meio de programa ou a partir da janela de propriedade.

## Propriedade Charset

Define ou retorna o conjunto de caracteres usado na fonte.

### Sintaxe

*object*.**Charset** [ = *value* ]

A sintaxe da propriedade **Charset** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Um valor inteiro que especifica o conjunto de caracteres usados pela fonte, conforme descrito em Configurações.

### Configurações

Estas são algumas das configurações comuns para *value*:

Value	Descrição
0	Caracteres-padrão do Windows
2	O conjunto de caracteres de símbolos.
128	O conjunto de caracteres de byte duplo (DBCS) exclusivo da versão japonesa do Windows
255	Caracteres ampliados normalmente exibidos por aplicativos DOS.

### Comentários

A definição da propriedade **Charset** como um destes valores disponíveis seleciona o conjunto de caracteres somente se ele estiver disponível na fonte atual.

## Propriedade DataBindings

Retorna a coleção de objetos **DataBindings** contendo as propriedades acopláveis disponíveis ao desenvolvedor.

### Sintaxe

*object*.**DataBindings**

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na

---

lista Applies To.

---

## Objeto Control

O nome de classe de todos os controles internos do Visual Basic.

### Sintaxe Control

#### Comentários

Pode-se dimensionar uma variável como um objeto **Control**, e referir-se a ele como seria feito a um controle em um formulário. Isto é demonstrado por:

```
Dim C as Control  
Set C = Command1
```

## Propriedade IsBindable

Retorna um valor Booleano indicando se a propriedade é acoplável. Esta propriedade é somente leitura.

### Sintaxe *object.IsBindable*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

#### Comentários

Usa-se esta propriedade para determinar se a propriedade é acoplável.

**Observação:** Esta propriedade normalmente é utilizada em um assistente, para verificar se uma propriedade é acoplável.

## Propriedade IsDataSource

Retorna um valor Booleano indicando se a propriedade é uma origem de dados. Esta propriedade é somente leitura.

### Sintaxe *object.IsDataSource*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

#### Comentários

Usa-se esta propriedade para determinar se ela é uma origem de dados e pode ser anexada a um controle de dados.

**Observação:** Esta propriedade normalmente é utilizada em um assistente para verificar se uma propriedade é uma origem de dados.

## Método Resync (Remote Data)

Intercepta os valores de conflito de lote para a linha atual.

### Sintaxe *object.Resync*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

#### Comentários

O método **Resync** somente é válido ao se utilizar Client-Batch Cursors.

**Resync** resincroniza as colunas na linha atual na biblioteca de cursores com os dados atuais no servidor (visíveis para sua transação). Caso a linha não tenha sido modificada, este método altera as propriedades **Value** e **OriginalValue** para corresponder ao que está atualmente no servidor.

Caso a linha tenha sido modificada, este método somente ajusta a propriedade **OriginalValue**, de modo que não sejam perdidas as edições. Este segundo caso é útil quando se deseja evitar um conflito de concorrência otimista.

O último caso em que isto é utilizado é quando se está lidando com uma linha que se tenta atualizado usando **BatchUpdate**, mas ocorreu um conflito devido a uma falha de verificação de concorrência. Neste caso, este método ajustará **BatchConflictValue** para refletir a versão mais recente da coluna no servidor.

## Método Files

Retorna uma coleção de nomes de arquivo usados pelo formato vbCFFiles (uma coleção **Da-**

**taObjectFiles**) que por sua vez contém uma lista de todos os nomes de arquivo usados por um objeto **DataObject**; por exemplo, os nomes de arquivo que o usuário arrasta de e para o Windows File Explorer.

### Sintaxe

*object.Files(index)*

A sintaxe da coleção **Files** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto <b>DataObject</b> .
<i>index</i>	Um inteiro que é um índice de uma matriz de nomes de arquivos.

### Comentários

A coleção **Files** é preenchida por nomes de arquivos somente quando o objeto **DataObject** contém dados do tipo **vbCFFiles**. O objeto **DataObject** pode conter diversos tipos de dados. Pode-se iterar pela coleção para recuperar a lista de nomes de arquivo.

A coleção **Files** pode ser preenchida para permitir que aplicativos do Visual Basic funcionem como uma origem de arraste de uma lista de arquivos.

## Propriedade ObjectAcceptFormats

Retorna a lista de formatos que podem ser aceitos por um objeto.

### Sintaxe

*object.ObjectAcceptFormats(number)*

A sintaxe da propriedade **ObjectAcceptFormats** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Uma <u>expressão numérica</u> que avalia para um inteiro indicando o elemento na matriz.

### Comentários

A lista é uma matriz de seqüência de caracteres baseada em zero. Os elementos da matriz podem ser usados para definir a propriedade **Format** ao se obter dados de um objeto usando as propriedades **Data** e **DataText**.

## Propriedade StandardSize

Define a página de propriedade como um tamanho padrão.

### Sintaxe

*object.StandardSize [= value]*

A sintaxe da propriedade **StandardSize** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Uma expressão de inteiro especificando o tipo de ponteiro de mouse exibido, conforme descrito em Configurações.

### Configurações

As configurações para *value* são:

Constante	Valor	Descrição
<b>Custom</b>	0	(Padrão) Tamanho determinado pelo objeto.
<b>Small</b>	1	Define <b>PropertyPage StandardSize</b> como 101 pixels de altura por 375 pixels de largura.
<b>Large</b>	2	Define <b>PropertyPage StandardSize</b> como 179 pixels de altura por 375 pixels de largura.

---

## Objeto Global

Um objeto **Global** é um objeto de aplicativo que permite acessar propriedades e métodos a nível de aplicativo.

### Sintaxe Global

#### Comentários

**Global** é um tipo de dados Object. Visto que o objeto **Global** é um objeto de aplicativo automaticamente referido, não é necessário codificar uma referência específica para este objeto.

## Propriedade App

Retorna o objeto **App**, um objeto global acessado com a palavra-chave **App**. Ela determina ou especifica informações sobre o título do aplicativo, informações de versão, o caminho e nome de seu arquivo executável e arquivos da **Ajuda**, e se uma ocorrência anterior do aplicativo está sendo executada.

### Sintaxe App

#### Comentários

O objeto **App** não tem eventos ou métodos.

## Propriedade Clipboard

Retorna um objeto **Clipboard**, que oferece acesso à **Área de transferência** do sistema.

### Sintaxe Clipboard

#### Comentários

O objeto **Clipboard** é usado para manipular texto e elementos gráficos na **Área de transferência**. Pode-se usar este objeto para permitir a um usuário copiar, recortar e colar texto ou elementos gráficos no seu aplicativo. Antes de copiar qualquer material para o objeto **Clipboard**, deve-se limpar seu conteúdo executando um método **Clear** como `Clipboard.Clear`.

Observe que o objeto **Clipboard** é compartilhado por todos os aplicativos do Windows e, assim, o conteúdo está sujeito a alterações toda vez que se alterna para outro aplicativo.

O objeto **Clipboard** pode conter diversas partes de dados, desde que cada parte esteja em um formato diferente. Por exemplo, pode-se usar o método **SetData** para colocar um bitmap na **Área de transferência** com o formato **vbCFDIB** e, em seguida, usar o método **SetText** com o formato **vbCFTText** para colocar texto na **Área de transferência**. Pode-se, então, usar o método **GetText** para recuperar o texto ou o método **GetData** para recuperar o elemento gráfico. Os dados da **Área de transferência** serão perdidos quando outro conjunto de dados do mesmo formato for colocado na **Área de transferência** através de código ou de um comando de menu.

## Propriedade Forms

Retorna uma coleção Forms, que é uma coleção cujos elementos representam cada formulário carregado em um aplicativo. A coleção inclui o formulário MDI do aplicativo, formulários MDI filha e formulários não-MDI. A coleção **Forms** tem uma única propriedade, **Count**, que especifica o número de elementos na coleção.

### Sintaxe Forms(index)

O espaço reservado *index* representa um inteiro entre 0 e `Forms.Count - 1`.

#### Comentários

Pode-se usar a coleção **Forms** para iterar por todos os formulários carregados em um aplicativo. Ela identifica uma variável global intrínseca chamada **Forms**. Pode-se passar **Forms(index)** a uma função cujo argumento seja especificado como uma classe **Forms**.

## Propriedade Printer

Retorna um objeto **Printer**, que permite comunicar-se com uma impressora do sistema (inicialmente a impressora padrão do sistema).

### Sintaxe Printer

---

## Comentários

Usam-se métodos gráficos para se desenhar texto e elementos gráficos no objeto **Printer**. Uma vez que o objeto **Printer** contém o resultado que se deseja imprimir, pode-se usar o método **EndDoc** para enviar o resultado diretamente à impressora padrão do aplicativo. Deve-se verificar e possivelmente revisar o layout dos seus formulários se eles forem impressos. Se utilizar o método **PrintForm** para imprimir um formulário, por exemplo, imagens gráficas podem ser recortadas na parte inferior da página e o texto transferido para a página seguinte.

## Propriedade Printers

Retorna uma coleção **Printers** que permite reunir informações sobre todas as impressoras disponíveis no sistema.

### Sintaxe

**Printers**(*index*)

O espaço reservado *index* representa um inteiro no intervalo de 0 a `Printers.Count-1`.

### Comentários

A coleção **Printers** permite consultar as impressoras disponíveis, de modo que se possa especificar uma impressora padrão para o seu aplicativo. Por exemplo, caso se deseje descobrir quais das impressoras disponíveis utiliza um driver de impressora específico. O código abaixo pesquisa todas as impressoras disponíveis para localizar a primeira impressora com sua orientação de página definida como retrato e, em seguida, define a mesma como a impressora padrão:

```
Dim X As Printer
For Each X In Printers
    If X.Orientation = vbPRORPortrait Then
        ' Definir a impressora como padrão do sistema.
        Set Printer = X
        ' Interromper a procura pela impressora.
        Exit For
    End If
Next
```

Designa-se uma das impressoras da coleção **Printers** como a impressora padrão utilizando-se a instrução **Set**. O exemplo anterior designa a impressora identificada pela variável de objeto *x*, como impressora padrão para o aplicativo.

**Observação:** Caso seja utilizada a coleção **Printers** para especificar uma determinada impressora, como em `Printers(3)`, pode-se somente acessar propriedades somente leitura. Para ler e gravar as propriedades de uma impressora individual, deve-se primeiro tornar esta impressora a impressora padrão do aplicativo.

## Propriedade Screen

Retorna um objeto **Screen**, que permite manipular formulários de acordo com seu posicionamento na tela e controlar o ponteiro do mouse fora dos formulários do seu aplicativo durante o tempo de execução. O objeto **Screen** é acessado com a palavra-chave **Screen**.

### Sintaxe

**Screen**

### Comentários

O objeto **Screen** é a área de trabalho inteira do Windows. Usando-se o objeto **Screen**, pode-se definir a propriedade **MousePointer** do objeto **Screen** como o ponteiro de ampulheta enquanto um formulário de janela restrita é exibido.

## Eventos Activate, Deactivate

- Ativar — ocorre quando um objeto torna-se a janela ativa.
- Desativar — ocorre quando um objeto não é mais a janela ativa.

### Sintaxe

**Private Sub** *object\_Activate*( )

**Private Sub** *object\_Deactivate*( )

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

---

## Comentários

Um objeto pode tornar-se ativo pela ação do usuário como, por exemplo, clicar nele, ou usando-se os métodos **Show** ou **SetFocus** no código.

O evento **Activate** somente pode ocorrer quando um objeto está visível. Por exemplo, um formulário carregado com a instrução **Load** não está visível, a menos que seja usado o método **Show**, ou que a propriedade **Visible** do formulário seja configurada como **True**.

Os eventos **Activate** e **Deactivate** somente ocorrem ao se mover o foco dentro do aplicativo. Mover o foco de e para um objeto em outro aplicativo não dispara nenhum dos dois eventos. O evento **Deactivate** não ocorre ao se carregar um objeto.

O evento **Activate** ocorre antes do evento **GotFocus**; o evento **LostFocus** ocorre antes do evento **Deactivate**.

Estes eventos ocorrem para formulários MDI filhobvbyb somente quando o foco muda de um filho a outro. Em um objeto **MDIForm** com dois formulários filho, por exemplo, os formulários filho recebem estes eventos quando o foco se move entre eles. Entretanto, quando o foco se alterna de um formulário filho para um formulário filho não-MDI, o **MDIForm** pai recebe os eventos **Activate** e **Deactivate**.

Se um arquivo .exe criado pelo Visual Basic exibe uma caixa de diálogo criada por um arquivo .dll também criado em Visual Basic, o formulário do arquivo .exe obterá os eventos **Deactivate** e **LostFocus**. Isto pode ser inesperado, pois deve-se obter o evento **Deactivate**:

- Caso o objeto seja um componente fora de processo.
- Caso o objeto não tenha sido escrito em Visual Basic.
- No ambiente de desenvolvimento ao chamar-se uma DLL criada em Visual Basic.

## Evento Change

Indica que o conteúdo de um controle foi alterado. Como e quando este evento ocorre com o controle:

- **ComboBox** — altera o texto na parte da caixa de texto do controle. Ocorre somente se a propriedade **Style** estiver configurada como 0 (Dropdown Combo) ou 1 (Simple Combo) e o usuário altera o texto ou a configuração da propriedade **Text** é alterada através de código.
- **DirListBox** — altera o diretório selecionado. Ocorre quando o usuário clica duas vezes em um novo diretório, ou quando se modifica a configuração da propriedade **Path** através de código.
- **DriveListBox** — altera a unidade de disco selecionada. Ocorre quando o usuário seleciona uma nova unidade de disco, ou quando a configuração da propriedade **Drive** é modificada através de código.
- **HScrollBar** e **VScrollBar** (barras de rolagem horizontal e vertical) — movem a parte da caixa de rolagem da barra de rolagem. Ocorre quando o usuário rola ou quando a configuração da propriedade **Value** é modificada através de código.
- **Label** — altera o conteúdo de **Label**. Ocorre quando um vínculo DDE atualiza dados, ou quando a configuração da propriedade **Caption** é alterada através de código.
- **PictureBox** — altera o conteúdo de **PictureBox**. Ocorre quando um vínculo DDE atualiza dados, ou quando a configuração da propriedade **Picture** é modificada por meio de código.
- **TextBox** — altera o conteúdo da caixa de texto. Ocorre quando um vínculo DDE atualiza dados, quando um usuário altera o texto, ou quando a propriedades **Text** é modificada através de código.

## Sintaxe

**Private Sub** *object\_Change*([*index As Integer*])

A sintaxe do evento **Change** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista <b>Applies To</b> .
<i>index</i>	Um inteiro que identifica exclusivamente um controle, caso ele esteja em uma <u>matriz de controle</u>

## Comentários

O procedimento de evento **Change** pode sincronizar ou coordenar a exibição de dados entre controles. Por exemplo, pode-se usar um procedimento de evento **Change** de barra de rolagem para atualizar a configuração da propriedade **Value** da barra de rolagem em um controle **TextBox**. Ou, pode-se usar um procedimento de evento **Change** para exibir dados e fórmulas em uma área de trabalho e os resultados em outra área.

Procedimentos de evento **Change** também são úteis para atualizar propriedades em controles de sistema de arquivo (**DirListBox**, **DriveListBox** e **FileListBox**). Por exemplo, pode-se atualizar a configuração da propriedade **Path** para que um controle **DirListBox** reflita uma alteração na con-



figuração da propriedade **Drive** de um controle **DriveListBox**.

**Observação:** Um procedimento de evento **Change** pode, algumas vezes, provocar um evento em cascata. Isto ocorre quando o evento **Change** do controle altera o conteúdo do controle, por exemplo, configurando uma propriedade no código que determina o valor do controle, por exemplo como a configuração da propriedade **Text** de um controle **TextBox**. Para evitar-se um evento em cascata:

- Se possível, evite escrever um procedimento de evento **Change** para um controle que altere conteúdo de controle. Caso tal procedimento seja escrito, certifique-se de definir um sinalizador que impeça mais alterações enquanto as alterações atuais estiverem sendo feitas.
- Evite criar dois ou mais controles cujos procedimentos de evento **Change** afetem um ao outro, por exemplo, dois controles **TextBox** que atualizam um ao outro durante seus eventos **Change**.
- Evite usar uma função ou instrução **MsgBox** neste evento para controles **HScrollBar** e **VScrollBar**.

## Evento Clic

Ocorre quando o usuário pressiona e, em seguida, solta o botão do mouse sobre um objeto. Pode também ocorrer quando o valor de um controle é alterado.

Para um objeto **Form**, este evento ocorre quando o usuário clica em uma área vazia ou controle desativado. Para um controle, este evento ocorre quando o usuário:

- Clica em um controle com o botão direito ou esquerdo do mouse. Com um controle **CheckBox**, **CommandButton**, **Listbox** ou **OptionButton**, o evento **Clic** ocorre somente quando o usuário clica o botão esquerdo do mouse.
- Seleciona um item em um controle **ComboBox** ou **ListBox**, pressionando as teclas de direção, ou clicando o botão do mouse.
- Pressiona a BARRA DE ESPAÇOS quando um controle **CommandButton**, **OptionButton** ou **CheckBox** tem o foco.
- Pressiona ENTER quando um formulário tem um controle **CommandButton** com sua propriedade **Default** configurada como **True**.
- Pressiona ESC quando um formulário tem um botão **Cancel** — um controle **CommandButton** com sua propriedade **Cancel** configurada como **True**.
- Pressiona um tecla de acesso de um controle. Por exemplo, se a legenda de um controle **CommandButton** é "&Go", pressionando-se ALT+G dispara-se o evento.

Pode-se ainda disparar o evento **Clic** no código:

- Configurando a propriedade **Value** de um controle **CommandButton** como **True**.
- Configurando a propriedade **Value** de um controle **OptionButton** como **True**.
- Alterando a configuração da propriedade **Value** de um controle **CheckBox**.

### Sintaxe

**Private Sub Form\_Clic( )**

**Private Sub** *object\_Clic***(***[index As Integer]***)**

A sintaxe do evento **Clic** tem estas partes:

Part	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista <b>Applies To</b> .
<i>index</i>	Um inteiro que identifica exclusivamente um controle, caso ele esteja em uma <u>matriz de controle</u> .

### Comentários

Tipicamente, anexa-se um procedimento de evento **Clic** a um controle **CommandButton**, objeto **Menu** ou controle **PictureBox** para executar comandos e ações do tipo comando. Para os outros controles aplicáveis, usa-se este evento para disparar ações em resposta a uma alteração no controle.

Pode-se usar a propriedade **Value** de um controle para testar o estado do controle a partir do código. Clicando-se em um controle, geram-se eventos **MouseDown** e **MouseUp** além do evento **Clic**. A ordem em que estes três eventos ocorrem varia de controle para controle. Por exemplo, para os controles **ListBox** e **CommandButton**, os eventos ocorrem nesta ordem: **MouseDown**, **Clic**, **MouseUp**. Mas, para os controles **FileListBox**, **Label** ou **PictureBox**, os eventos ocorrem nesta ordem: **MouseDown**, **MouseUp** e **Clic**. Ao se anexar procedimentos de evento para estes eventos relacionados, certifique-se de que suas ações não sejam conflitantes. Se a ordem de eventos for importante em seu aplicativo, teste o controle para determinar a ordem dos eventos.

**Observação:** Para distinguir entre os botões direito, esquerdo e central do mouse, use os eventos **MouseDown** e **MouseUp**.

Se houver código no evento Clic, o evento DblClic nunca será desencadeado porque o evento Clic é o primeiro evento a disparar entre os dois. Como resultado, o clique do mouse é interceptado pelo evento Clic, assim o evento DblClic nunca ocorre.

## Evento DragDrop

Ocorre quando uma operação de arrastar-e-soltar termina como resultado de se arrastar um controle sobre um objeto e soltar o botão do mouse, ou usando-se o método **Drag** com seu argumento *action* configurado como 2 (Soltar).

### Sintaxe

**Private Sub Form\_DragDrop(source As Control, x As Single, y As Single)**

**Private Sub MDIForm\_DragDrop(source As Control, x As Single, y As Single)**

**Private Sub object\_DragDrop([index As Integer,]source As Control, x As Single, y As Single)**

A sintaxe do evento DragDrop tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	Um inteiro que identifica com exclusividade um controle, caso ele esteja em uma <u>matriz de controle</u> .
<i>source</i>	O controle que está sendo arrastado. Pode-se incluir propriedades e métodos no procedimento de evento com este argumento — por exemplo, <code>Source.Visible = 0</code> .
<i>x, y</i>	Um número que especifica a posição atual horizontal ( <i>x</i> ) e vertical ( <i>y</i> ) do ponteiro do mouse dentro do formulário ou controle-destino. Estas coordenadas são sempre expressas em termos do sistema de coordenadas do alvo, conforme configurado pelas propriedades <b>ScaleHeight</b> , <b>ScaleWidth</b> , <b>ScaleLeft</b> e <b>ScaleTop</b> .

### Comentários

Use um procedimento de evento DragDrop para controlar o que acontece após uma operação de arraste ter terminado. Por exemplo, pode-se mover o controle de origem até um novo local, ou copiar um arquivo de um local para outro.

Quando controles múltiplos podem ser potencialmente usados em um argumento *source*:

- Usa-se a palavra-chave **TypeOf** com a instrução **If** para determinar o tipo de controle usado com *source*.
- Usa-se a propriedade **Tag** do controle para identificá-lo e, em seguida, usa-se um procedimento de evento DragDrop.

**Observação:** Usa-se a propriedade **DragMode** e o método **Drag** para especificar a maneira como o arraste é iniciado. Uma vez que o arraste tenha sido iniciado, pode-se manipular os eventos que precedem um evento DragDrop com um procedimento de evento DragOver.

## Evento DragOver

Ocorre quando uma operação arrastar-e-soltar está sendo executada. Pode-se usar este evento para monitorar o ponteiro do mouse à medida que ele entra, sai ou permanece diretamente sobre um alvo válido. A posição do ponteiro do mouse determina o objeto de destino que recebe este evento.

### Sintaxe

**Private Sub Form\_DragOver(source As Control, x As Single, y As Single, state As Integer)**

**Private Sub MDIForm\_DragOver(source As Control, x As Single, y As Single, state As Integer)**

**Private Sub object\_DragOver([index As Integer,]source As Control, x As Single, y As Single, state As Integer)**

A sintaxe do evento DragOver tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	Um inteiro que identifica com exclusividade um controle, caso ele esteja em uma <u>matriz de controle</u> .
<i>source</i>	O controle que está sendo arrastado. Pode-se referir a pro-

	priedades e métodos no procedimento de evento com este argumento — por exemplo, <code>Source.Visible = False</code> .
<i>x, y</i>	Um número que especifica a posição atual horizontal ( <i>x</i> ) e vertical ( <i>y</i> ) do ponteiro do mouse dentro de um formulário ou controle alvo. Estas coordenadas são sempre expressas em termos do sistema de coordenadas, conforme definido pelas propriedades <b>ScaleHeight</b> , <b>ScaleWidth</b> , <b>ScaleLeft</b> e <b>ScaleTop</b> .
<i>State</i>	Um inteiro que corresponde ao estado de transição do controle que está sendo arrastado em relação a um formulário ou controle destino: 0 = Entrar (o controle de origem está sendo arrastado dentro do alcance de um destino). 1 = Sair (o controle de origem está sendo arrastado para fora do alcance de um destino). 2 = Sobre (o controle de origem está sendo movido de uma posição a outra, dentro do destino).

### Comentários

Use um procedimento de evento `DragOver` para determinar o que acontece após o arraste ter-se iniciado, e antes que um controle seja solto sobre o destino. Por exemplo, pode-se verificar um intervalo de destino válido, selecionando-se o destino (definindo-se a propriedade **BacColor** ou **ForeColor** no código) ou exibindo-se um ponteiro de arraste especial (definindo-se a propriedade **DragIcon** ou **MousePointer** no código).

Use o argumento *state* para determinar as ações em pontos de transição importantes. Por exemplo, pode-se selecionar um possível destino quando *state* é configurado como 0 (Entrar), e restaurar a aparência anterior do objeto quando *state* está configurado como 1 (Sair).

Quando um objeto recebe um evento `DragOver` enquanto *state* está definido como 0 (Entrar):

- Se o controle de origem é solto sobre o objeto, este objeto recebe um evento `DragDrop`.
- Se o controle de origem não é solto sobre o objeto, este objeto recebe outro evento `DragOver` quando *state* está configurado como 1 (Sair).

**Observação:** Usa-se a propriedade **DragMode** e o método **Drag** para especificar a maneira como o arraste é iniciado. Para obter sugestões de técnicas com o argumento *source*, consulte "Comentários" no tópico "Evento `DragDrop`".

## Evento `DropDown`

Ocorre quando a parte da lista de um controle **ComboBox** está para ser exibida; este evento não ocorre se a propriedade **Style** de um controle **ComboBox** está configurada como 1 (Caixa de combinação simples).

### Sintaxe

**Private Sub** *object*\_**DropDown**([*index* As Integer])

A sintaxe do evento `DropDown` tem estas partes:

Part	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	Um inteiro que identifica com exclusividade um controle, caso ele esteja em uma <u>matriz de controle</u> .

### Comentários

Usa-se um procedimento de evento `DropDown` para efetuar atualizações finais em uma lista **ComboBox**, antes que o usuário efetue uma seleção. Isto permite adicionar ou remover itens da lista usando os métodos **AddItem** ou **RemoveItem**. Esta flexibilidade é útil quando se deseja alguma inter-relação entre controles — por exemplo, caso aquilo que se deseja carregar em uma lista **ComboBox** dependa do que o usuário seleciona em um grupo **OptionButton**.

## Evento `GotFocus`

Ocorre quando um objeto recebe o foco, por ação do usuário, por exemplo, usando a tecla de tabulação ou clicando no objeto, ou alterando o foco no código através do método **SetFocus**. Um formulário somente recebe o foco quando todos os controles visíveis estão desativados.

### Sintaxe

**Private Sub** *Form*\_**GotFocus**( )

---

**Private Sub *object*\_GotFocus(*[index As Integer]*)**

A sintaxe do evento GotFocus tem estas partes:

<b>Parte</b>	<b>Descrição</b>
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	Um inteiro que identifica com exclusividade um controle, caso ele esteja em uma <u>matriz de controle</u> .

**Comentários**

Tipicamente, usa-se o procedimento de evento GotFocus para especificar as ações que ocorrem quando um controle ou formulário recebe foco pela primeira vez. Por exemplo, anexando-se um procedimento de evento GotFocus a cada controle de um formulário, pode-se orientar o usuário exibindo-se curtas instruções ou mensagens de barra de status. Pode-se ainda fornecer dicas visuais ativando, desativando ou exibindo outros controles que dependam do controle que tem o foco.

**Observação:** Um objeto somente pode receber o foco se suas propriedades **Enabled** e **Visible** estiverem definidas como **True**. Para personalizar a interface de teclado em Visual Basic para mover o foco, configura-se a ordem de tabulação ou especificam-se teclas de acesso para controles em um formulário.

## Eventos eyDown, eyUp

Ocorrem quando o usuário pressiona (eyDown) ou solta (eyUp) uma tecla, enquanto um objeto tem o foco. (Para interpretar caracteres ANSI, usa-se o evento eyPress.)

**Sintaxe**

**Private Sub Form\_eyDown(*eycode As Integer*, *shift As Integer*)**

**Private Sub *object*\_eyDown(*[index As Integer]*,*eycode As Integer*, *shift As Integer*)**

**Private Sub Form\_eyUp(*eycode As Integer*, *shift As Integer*)**

**Private Sub *object*\_eyUp(*[index As Integer]*,*eycode As Integer*, *shift As Integer*)**

A sintaxe dos eventos eyDown e eyUp tem estas partes:

<b>Part</b>	<b>Descrição</b>
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	Um inteiro que identifica com exclusividade um controle, caso ele esteja em uma <u>matriz de controle</u> .
<i>eycode</i>	Um código de tecla, por exemplo, <b>vbeyF1</b> (a tecla F1) ou <b>vbeyHome</b> (a tecla HOME). Para especificar os códigos de tecla, use as constantes contidas na <u>biblioteca de objetos</u> do (VB)Visual Basic no <u>Object Browser</u> .
<i>shift</i>	Um inteiro que corresponde ao estado das teclas SHIFT, CTRL e ALT no momento do evento. O argumento <i>shift</i> é um campo de bit com os bits menos importantes correspondendo à tecla SHIFT (bit 0), à tecla CTRL (bit 1) e à tecla ALT (bit 2). Estes bits correspondem, respectivamente, aos valores 1, 2 e 4. Alguns, todos ou nenhum dos bits podem estar configurados, indicando que alguma, todas ou nenhuma das teclas está pressionada. Por exemplo, se tanto CTRL quanto ALT estiverem pressionadas, o valor de <i>shift</i> será 6.

**Comentários**

Em ambos os eventos, o objeto que tem o foco recebe todos os pressionamentos de tecla. Um formulário somente pode ter o foco se não contiver controles ativados e visíveis. Embora os eventos eyDown e eyUp possam aplicar-se à maioria das teclas, eles são mais comumente usados para:

- Teclas de caracteres estendidos como, por exemplo, teclas de função.
- Teclas de direção.
- Combinações de teclas com modificadores de teclado padrão.
- Distinguir entre o teclado numérico e as teclas numéricas comuns.

Use os procedimentos de evento eyDown e eyUp se precisar responder tanto ao pressionamento quanto à liberação de uma tecla.

eyDown e eyUp não são chamados para:

- A tecla ENTER se o formulário tiver um controle **CommandButton** com a propriedade **Default**

- configurada como **True**.
- A tecla ESC se o formulário tiver um controle **CommandButton** com a propriedade **Cancel** configurada como **True**.
- A tecla TAB.

eyDown e eyUp interpretam as maiúsculas e minúsculas de cada caractere através de dois argumentos: *eycode*, que indica a tecla física (retornando, assim, A e a como a mesma tecla) e *shift*, que indica o estado *shift+tecla* e, portanto, retorna A ou a.

Caso seja necessário testar o argumento *shift*, pode-se usar as constantes *shift* que definem os bits dentro de argumentos. As constantes têm os valores abaixo:

Constante	Valor	Descrição
<b>vbShiftMas</b>	1	Máscara de bits da tecla SHIFT.
<b>VbCtrlMas</b>	2	Máscara de bits da tecla CTRL.
<b>VbAltMas</b>	4	Máscara de bits da tecla ALT.

As constantes funcionam como máscaras de bits que podem ser usadas para testar qualquer combinação de teclas.

Testa-se uma condição designando primeiro cada resultado a uma variável inteira temporária e, em seguida, comparando *shift* a uma máscara de bits. Usa-se o operador **And** com o argumento *shift* para testar se a condição é maior que 0, indicando que o modificador foi pressionado, como no exemplo abaixo:

```
ShiftDown = (Shift And vbShiftMas) > 0
```

Em um procedimento, pode-se testar qualquer combinação de condições, como neste exemplo:

```
If ShiftDown And CtrlDown Then
```

**Observação:** Caso a propriedade **eyPreview** esteja configurada como **True**, um formulário recebe estes eventos antes que controles no formulário os recebam. Utilize a propriedade **eyPreview** para criar rotinas globais de manipulação de teclado.

## Evento eyPress

Ocorre quando o usuário pressiona e solta uma tecla ANSI.

### Sintaxe

```
Private Sub Form_eyPress(eyascii As Integer)
```

```
Private Sub object_eyPress([index As Integer,]eyascii As Integer)
```

A sintaxe do evento eyPress tem estas partes:

Part	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	Um inteiro que identifica com exclusividade um controle, caso ele esteja em uma <u>matriz de controle</u> .
<i>eyascii</i>	Um inteiro que retorna um código numérico de tecla ANSI padrão. <i>eyascii</i> é passado como referência; sua alteração envia um caractere diferente ao objeto. Alterando-se <i>eyascii</i> para 0, cancela-se o pressionamento de tecla, de modo que o objeto não recebe caractere algum.

### Comentários

O objeto com o foco recebe o evento. Um formulário somente pode receber o evento, se não tiver nenhum controle ativado e visível, ou se a propriedade **eyPreview** estiver definida como **True**. Um evento eyPress envolve qualquer caractere de teclado imprimível, a tecla CTRL combinada com um caractere do alfabeto padrão ou um dos poucos caracteres especiais como, por exemplo, a tecla ENTER ou BACSPACE. Um evento eyPress é útil para interceptar pressionamentos de teclas em um controle **TextBox** ou **ComboBox**. Ele permite testar imediatamente pressionamentos de teclas para validade, ou para formatar caracteres à medida que são digitados. A alteração do valor do argumento *eyascii* altera o caractere exibido.

Pode-se converter o argumento *eyascii* em um caractere usando-se a expressão:

```
Chr(eyAscii)
```

Pode-se, em seguida, executar operações com seqüências de caracteres e traduzir o caractere de volta para um número ANSI que o controle possa interpretar usando a expressão:

```
eyAscii = Asc(char)
```

Usa-se os procedimentos de evento eyDown e eyUp para tratar todo pressionamento de tecla não

reconhecido por eyPress como, por exemplo, teclas de função, teclas de edição, teclas de navegação e qualquer combinação destas com modificadores de teclado. Diferente dos eventos eyDown e eyUp, o eyPress não indica o estado físico do teclado; ao invés ele passa um caractere.

eyPress interpreta maiúscula e minúscula de cada caractere como códigos de tecla separados e, portanto, como dois caracteres separados. eyDown e eyUp interpretam a maiúscula e a minúscula de cada caractere por meio de dois argumentos: *eycode*, que indica a tecla física (retornando, assim A e a como a mesma tecla), e *shift*, que indica o estado de *shift+tecla* e, portanto, retorna A ou a.

Caso a propriedade **eyPreview** esteja definida como **True**, um formulário recebe o evento antes que os controles do formulário o receba. Utilize a propriedade **eyPreview** para criar rotinas globais de manipulação de teclado.

**Observação:** O número ANSI para a combinação de teclas CTRL+@ é 0. Uma vez que o Visual Basic reconhece um valor de *eyascii* de 0 como uma seqüência de caracteres de comprimento zero (""), evite usar CTRL+@ em seus aplicativos.

## Evento LinClose

Ocorre quando termina uma conversação DDE. Qualquer um dos aplicativos em uma conversação DDE pode terminar a conversação a qualquer momento.

### Sintaxe

**Private Sub Form\_LinClose( )**

**Private Sub MDIForm\_LinClose( )**

**Private Sub *object*\_LinClose([*index* As Integer])**

A sintaxe do evento LinClose tem estas partes:

<b>Parte</b>	<b>Descrição</b>
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	Um inteiro que identifica com exclusividade um controle, caso ele esteja em uma <u>matriz de controle</u> .

### Comentários

Tipicamente, usa-se um procedimento de evento LinClose para notificar o usuário de que a conversação DDE foi encerrada. Pode-se ainda incluir informações para a solução de problemas no restabelecimento de uma conexão, ou onde procurar ajuda. Para mensagens curtas utiliza-se a função **MsgBox**.

## Evento LinError

Ocorre quando existe um erro durante uma conversação DDE. Este evento somente é reconhecido como resultado de um erro relacionado com DDE que ocorra quando o código do Visual Basic estiver sendo executado. O número do erro é passado como um argumento.

### Sintaxe

**Private Sub Form\_LinError(*linerr* As Integer)**

**Private Sub MDIForm\_LinError(*linerr* As Integer)**

**Private Sub *object*\_LinError([*index* As Integer,]*linerr* As Integer)**

A sintaxe do evento LinError tem estas partes:

<b>Parte</b>	<b>Descrição</b>
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>linerr</i>	Número de erro do erro relacionado com DDE, conforme descrito em valores de retorno.
<i>index</i>	Um inteiro que identifica com exclusividade um controle, caso ele esteja em uma <u>matriz de controle</u> .

### Valores de retorno

A tabela abaixo lista todos os números de erro retornados para o argumento *linerr* e uma curta explicação de cada erro:

<b>Value</b>	<b>Descrição</b>
1	O outro aplicativo pediu dados no formato errado. Este erro pode ocorrer diversas vezes em seguida, enquanto o Visual Basic tenta achar um formato que o outro aplicativo reconhe-

	ça.
6	O aplicativo de <u>destino</u> tentou continuar uma conversação DDE após você ter configurado a propriedade <b>LinMode</b> em seu formulário de <u>origem</u> como 0 (Nenhum).
7	Todos os vínculos de origem estão sendo usados (existe um limite de 128 vínculos por origem).
8	Para controles de destino: um vínculo automático ou método <b>LinRequest</b> falhou ao atualizar os dados contidos no controle. Para formulários de origem: o destino tentou <u>enviar</u> dados a um controle e a tentativa falhou.
11	Memória insuficiente para DDE.

### Comentários

Usa-se um procedimento de evento LinErros para notificar o usuário sobre um determinado erro que tenha ocorrido. Pode-se ainda incluir código para corrigir o problema ou informações para a solução de problemas no restabelecimento de um conexão, ou sobre onde procurar ajuda. Para mensagens curtas, utiliza-se a função **MsgBox**.

## Evento LinExecute

Ocorre quando uma seqüência de caracteres de comando é enviada por um aplicativo de destino em uma conversação DDE. O aplicativo de destino espera que o aplicativo de origem execute a operação descrita na seqüência de caracteres.

### Sintaxe

**Private Sub** *object\_LinExecute*(*cmdstr As String*, *cancel As Integer*)

A sintaxe do evento LinExecute tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>cmdstr</i>	A expressão de seqüência de caracteres de comando enviada pelo aplicativo de destino.
<i>cancel</i>	Um inteiro que informa o destino se a seqüência de caracteres de comando foi aceita ou recusada. Definindo-se <i>cancel</i> como 0 informa-se ao destino que a seqüência de caracteres de comando foi aceita. Definindo-se <i>cancel</i> como qualquer valor diferente de zero informa-se ao destino que a seqüência de caracteres de comando foi rejeitada. (O padrão é configurado como -1, indicando <i>cancel</i> .)

### Comentários

Não existe sintaxe obrigatória para *cmdstr*. A forma como o aplicativo responde a diferentes seqüências de caracteres é um problema do programador. Caso não se tenha criado um procedimento de evento LinExecute, o Visual Basic rejeita seqüências de caracteres de comando de aplicativos de destino.

## Evento LinNotify

Ocorre quando a origem alterou os dados definidos pelo vínculo DDE se a propriedade **LinMode** do controle de destino estiver configurada como 3 (Notificar).

### Sintaxe

**Private Sub** *object\_LinNotify*(*[index As Integer]*)

A sintaxe do evento LinNotify tem estas partes:

Part	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	Um inteiro que identifica com exclusividade um controle, caso ele esteja em uma <u>matriz de controle</u> .

### Comentários

Tipicamente, no evento LinNotify o código notifica o usuário, obtém imediatamente os novos dados ou adia a obtenção de dados para mais tarde. Pode-se usar o método **LinRequest** para obter

---

os novos dados da origem.

## Evento LinOpen

Ocorre quando uma conversação DDE está sendo iniciada.

### Sintaxe

**Private Sub Form\_LinOpen(*cancel As Integer*)**

**Private Sub MDIForm\_LinOpen(*cancel As Integer*)**

**Private Sub *object*\_LinOpen(*[index As Integer,]cancel As Integer*)**

A sintaxe do evento LinOpen tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>cancel</i>	Um inteiro que determina se a conversação DDE foi ou não estabelecida. Deixando-se <i>cancel</i> configurado como 0 (o padrão) estabelece-se a conversação. Configurando-se <i>cancel</i> como qualquer valor diferente de zero, recusa-se a conversação.
<i>index</i>	Um inteiro que identifica com exclusividade um controle, caso ele esteja em uma <u>matriz de controle</u> .

### Comentários

Este evento ocorre para formulários quando um aplicativo de destino está iniciando uma conversação DDE com o formulário. Ela ocorre em controles, quando um controle está iniciando uma conversação DDE com um aplicativo de origem.

## Evento Load

Ocorre quando um formulário é carregado. Para um formulário de inicialização, ocorre quando um aplicativo inicia como resultado de uma instrução **Load**, ou como resultado de uma referência a propriedades ou controle de um formulário não-carregado.

### Sintaxe

**Private Sub Form\_Load( )**

**Private Sub MDIForm\_Load( )**

### Comentários

Tipicamente, usa-se um procedimento de evento Load para incluir código de inicialização em um formulário — por exemplo, código que especifique as configurações padrão para controles, indique-se o conteúdo a ser carregado em controles **ComboBox** ou **ListBox**, e inicializam-se variáveis a nível de formulário.

O evento Load ocorre após o evento Initialize.

Ao ser feita uma referência no código a uma propriedade de um formulário não carregado, este é automaticamente carregado, mas não é tornado automaticamente visível, a menos que a propriedade **MDIChild** seja configurada como **True**. Caso um objeto **MDIForm** não esteja carregado e um formulário MDI filho esteja carregado, tanto o **MDIForm** quanto o formulário filho são automaticamente carregados, e ambos tornam-se visíveis. Outros formulários não são exibidos até que se utilize o método **Show**, ou se configure a propriedade **Visible** como **True**.

O código abaixo em um evento **MDIForm** Load carrega automaticamente um formulário MDI filho (pressupondo-se que *Form1* tenha sua propriedade **MDIChild** configurada como **True**):

```
Dim NewForm As New Form1
```

```
NewForm.Caption = "New Form" ' Carrega o formulário por referência.
```

Uma vez que todos os formulários filhos tornam-se visíveis ao serem carregados, a referência à propriedade **Caption** carrega o formulário e o torna visível.

**Observação:** Quando são criados procedimentos para eventos relacionados como, por exemplo **Activate**, **GotFocus**, **Paint** e **Resize**, certifique-se de que suas ações não entrem em conflito e que eles não provoquem eventos recursivos.

## Evento LostFocus

Ocorre quando um objeto perde o foco, por ação do usuário, por exemplo, usando a tecla de tabulação ou clicando em outro objeto, ou alterando o foco no código usando o método **SetFocus**.

### Sintaxe

**Private Sub Form\_LostFocus( )**

**Private Sub *object*\_LostFocus(*[index As Integer]*)**



---

A sintaxe do evento LostFocus tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	Um inteiro que identifica com exclusividade um controle, caso ele esteja em uma <u>matriz de controle</u> .

### Comentários

Um procedimento de evento LostFocus é útil principalmente para a verificação e atualizações de validação. Usando-se LostFocus, pode-se fazer com que a validação ocorra à medida que o usuário move o foco para fora do controle. Outro uso para este tipo de procedimento de evento é permitir, ocultar e exibir outros objeto como em um procedimento de evento GotFocus. Pode-se inverter ou alterar condições que são configuradas no procedimento de evento GotFocus do objeto.

Caso um arquivo .exe criado pelo Visual Basic exiba uma caixa de diálogo criada por um arquivo .dll também criado em Visual Basic, o formulário do arquivo .exe obterá eventos Deactivate e LostFocus. Isto pode ser inesperado, porque não se deve obter o evento Deactivate:

- Se o objeto for um componente fora de processo.
- Se o objeto não tiver sido escrito em Visual Basic.
- No ambiente de desenvolvimento ao se chamar uma DLL criada em Visual Basic.

## Eventos MouseDown, MouseUp

Ocorre quando o usuário pressiona (MouseDown) ou libera (MouseUp) um botão de mouse.

### Sintaxe

**Private Sub Form\_MouseDown(*button* As Integer, *shift* As Integer, *x* As Single, *y* As Single)**

**Private Sub MDIForm\_MouseDown(*button* As Integer, *shift* As Integer, *x* As Single, *y* As Single)**

**Private Sub *object*\_MouseDown([*index* As Integer,]*button* As Integer, *shift* As Integer, *x* As Single, *y* As Single)**

**Private Sub Form\_MouseUp(*button* As Integer, *shift* As Integer, *x* As Single, *y* As Single)**

**Private Sub MDIForm\_MouseUp(*button* As Integer, *shift* As Integer, *x* As Single, *y* As Single)**

**Private Sub *object*\_MouseUp([*index* As Integer,]*button* As Integer, *shift* As Integer, *x* As Single, *y* As Single)**

As sintaxes de evento MouseDown e MouseUp têm estas partes:

Part	Descrição
<i>object</i>	Retorna uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>Index</i>	Retorna um inteiro que identifica com exclusividade um controle que esteja em um <u>matriz de controle</u> .
<i>button</i>	Retorna um inteiro que identifica o botão que foi pressionado (MouseDown) ou liberado (MouseUp) para causar o evento. O argumento <i>button</i> é um campo de bit com bits correspondendo ao botão esquerdo (bit 0), botão direito (bit 1) e botão central (bit 2). Estes bits correspondem respectivamente aos valores 1, 2 e 4. Somente um dos bits é configurado, indicando o botão que causou o evento.
<i>shift</i>	Retorna um inteiro que corresponde ao estado das teclas SHIFT, CTRL e ALT quando o botão especificado no argumento <i>button</i> está pressionado ou solto. Um bit é configurado se a tecla estiver pressionada. O argumento <i>shift</i> é um campo de bit, com os bits menos importantes correspondendo à tecla SHIFT (bit 0), à tecla CTRL (bit 1) e à tecla ALT (bit 2). Estes bits correspondem, respectivamente, aos valores 1, 2 e 4. O argumento <i>shift</i> indica o estado destas teclas. Algum, todos ou nenhum dos bits pode ser definido, indicando que alguma, todas ou nenhuma das teclas está pressionada. Por exemplo, se tanto CTRL quando ALT estiverem pressionadas, o valor de

*shift* será 6.

*x, y* Retorna um número que especifica a localização atual do ponteiro do mouse. Os valores *x* e *y* são sempre expressos em termos do sistema de coordenadas definido pelas propriedades **ScaleHeight**, **ScaleWidth**, **ScaleLeft** e **ScaleTop** do objeto.

### Comentários

Usa-se um procedimento de evento **MouseDown** ou **MouseUp** para especificar ações que ocorrerão quando um determinado botão do mouse é pressionado ou solto. Diferente dos eventos **Clic** e **DbuClic**, os eventos **MouseDown** e **MouseUp** permitem distinguir entre os botões direito, esquerdo e central do mouse. Pode-se ainda escrever código para combinações de mouse e teclado que utilizem os modificadores de teclado **SHIFT**, **CTRL** e **ALT**.

Aplicam-se tanto aos eventos **Clic** quando **DbuClic**:

- Se um botão de mouse está pressionado enquanto o ponteiro está sobre um formulário ou controle, este objeto "captura" o mouse e recebe todos os eventos de mouse até, e incluindo, o último evento **MouseUp**. Isto implica em que as coordenadas de ponteiro de mouse *x, y* retornadas por um evento de mouse podem não estar sempre na área interna do objeto que os recebe.
- Se os botões do mouse forem pressionados em seguida, o objeto que captura o mouse após o primeiro pressionamento recebe todos os eventos de mouse, até que todos os botões sejam liberados.

Caso seja necessário testar os argumentos *button* ou *shift*, pode-se usar as constantes listadas na biblioteca de objetos (VB) do Visual Basic no Object Browser para definir os bits dentro do argumento:

Constante (Botão)	Valor	Descrição
<b>vbLeftButton</b>	1	O botão esquerdo está pressionado
<b>vbRightButton</b>	2	O botão direito está pressionado
<b>vbMiddleButton</b>	4	O botão central está pressionado
Constante (Shift)	Valor	Descrição
<b>vbShiftMas</b>	1	A tecla SHIFT está pressionada.
<b>VbCtrlMas</b>	2	A tecla CTRL está pressionada.
<b>VbAltMas</b>	4	A tecla ALT está pressionada.

As constantes funcionam então como máscaras de bits que podem ser usadas para se testar qualquer combinação de botões sem ter que calcular o valor exclusivo do campo de bit para cada combinação.

**Observação:** Pode-se usar um procedimento de evento **MouseMove** para responder a um evento causado pela movimentação do mouse. O argumento *button* para **MouseDown** e **MouseUp** diferem do argumento *button* usado para **MouseMove**. Para **MouseDown** e **MouseUp**, o argumento *button* indica exatamente um botão por evento, ao passo que para **MouseMove**, ele indica o estado atual de todos os botões.

## Evento MouseMove

Ocorre quando o usuário move o mouse.

### Sintaxe

**Private Sub Form\_MouseMove(button As Integer, shift As Integer, x As Single, y As Single)**

**Private Sub MDIForm\_MouseMove(button As Integer, shift As Integer, x As Single, y As Single)**

**Private Sub object\_MouseMove([index As Integer,] button As Integer, shift As Integer, x As Single, y As Single)**

A sintaxe do evento **MouseMove** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista <u>Applies To</u> .
<i>index</i>	Um inteiro que identifica com exclusividade um controle, caso ele esteja em uma <u>matriz de controle</u> .
<i>button</i>	Um inteiro que corresponde ao estado dos botões do mouse onde um bit é configurado se o botão estiver pressionado. O argumento <i>button</i> é um campo de bit com bits correspondendo

	ao botão esquerdo (bit 0), botão direito (bit 1) e botão central (bit 2). Estes bits correspondem, respectivamente, aos valores 1, 2 e 4. Ele indica o estado completo dos botões do mouse; algum, todos ou nenhum destes três bits pode ser definido, indicando que algum, todos ou nenhum dos botões do mouse está pressionado.
<i>shift</i>	Um inteiro que corresponde ao estado das teclas SHIFT, CTRL e ALT. Um bit é configurado se a tecla estiver pressionada. O argumento <i>shift</i> é um campo de bit com os bits menos importantes correspondendo à tecla SHIFT (bit 0), à tecla CTRL (bit 1) e à tecla ALT (bit 2). Estes bits correspondem, respectivamente, aos valores 1, 2 e 4. O argumento <i>shift</i> indica o estado destas teclas. Algum, todos ou nenhum dos bits pode ser definido, indicando que alguma, todas ou nenhuma das teclas está pressionada. Por exemplo, se tanto CTRL quando ALT estiverem pressionadas, o valor de <i>shift</i> será 6.
<i>x, y</i>	Um número que especifica a localização atual do ponteiro do mouse. Os valores <i>x</i> e <i>y</i> são sempre expressos em termos do sistema de coordenadas configurado pelas propriedades <b>ScaleHeight</b> , <b>ScaleWidth</b> , <b>ScaleLeft</b> e <b>ScaleTop</b> do objeto.

### Comentários

O evento `MouseMove` é gerado continuamente, à medida que o ponteiro do mouse se move pelos objetos. A menos que outro objeto tenha capturado o mouse, um objeto reconhece um evento `MouseMove` sempre que a posição do mouse está dentro de suas bordas.

Caso seja necessário testar os argumentos *button* ou *shift*, pode-se usar constantes listadas na biblioteca de objetos (VB) do Visual Basic no Object Browser para definir os bits dentro do argumento:

Constante (Botão)	Valor	Descrição
<b>vbLeftButton</b>	1	O botão esquerdo está pressionado.
<b>vbRightButton</b>	2	O botão direito está pressionado.
<b>vbMiddleButton</b>	4	O botão central está pressionado.
Constante (Shift)	Valor	Descrição
<b>vbShiftMas</b>	1	A tecla SHIFT está pressionada.
<b>vbCtrlMas</b>	2	A tecla CTRL está pressionada.
<b>vbAltMas</b>	4	A tecla ALT está pressionada.

As constantes, portanto, funcionam como máscaras de bits que podem ser usadas para se testar qualquer combinação de botões sem ter que calcular o valor exclusivo dos bits do campo para cada combinação.

Testa-se uma condição designando-se primeiro cada resultado a uma variável inteira temporária e, em seguida, comparando-se os argumentos *button* ou *shift* a uma máscara de bits. Usa-se o operador **And** com cada um dos argumentos para testar se a condição é maior que zero, indicando que a tecla ou botão está pressionado, como neste exemplo:

```
LeftDown = (Button And vbLeftButton) > 0
CtrlDown = (Shift And vbCtrlMas) > 0
```

A seguir, em um procedimento, pode-se testar qualquer combinação de condições, como neste exemplo:

```
If LeftDown And CtrlDown Then
```

**Observação:** Pode-se usar procedimentos de evento `MouseDown` e `MouseUp` para responder a eventos causados pelo pressionamento e liberação de botões de mouse.

O argumento *button* para `MouseMove` é diferente do argumento *button* para `MouseDown` e `MouseUp`. Para `MouseMove`, o argumento *button* indica o estado atual de todos os botões; um único evento `MouseMove` pode indicar que algum, todos ou nenhum botão está pressionado. Para `MouseDown` e `MouseUp`, o argumento *button* indica exatamente um botão por evento.

A qualquer momento em que se mova uma janela dentro de um evento `MouseMove`, pode-se provocar um evento em cascata. Eventos `MouseMove` são gerados quando a janela se move sob o ponteiro. Um evento `MouseMove` pode ser gerado, mesmo que o mouse esteja perfeitamente estacionário.

## Evento Paint

Ocorre quando parte ou todo um objeto é exibido após ter sido movido ou ampliado, ou após uma

---

janela que o estava cobrindo ter sido movida.

#### Sintaxe

**Private Sub Form\_Paint( )**

**Private Sub *object\_Paint*([*index As Integer*])**

A sintaxe do evento Paint tem estas partes:

Part	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	Um inteiro que identifica com exclusividade um controle, caso ele esteja em uma <u>matriz de controle</u> .

#### Comentários

O procedimento de evento Paint é útil quando se tem resultados de métodos gráficos no código. Com o procedimento Paint, pode-se assegurar que tal resultado seja regenerado quando necessário.

O evento Paint é chamado quando o método **Refresh** é utilizado. Caso a propriedade **AutoRedraw** esteja configurada como **True**, a regeneração ou redesenho é automático, assim nenhum evento Paint é necessário.

Caso a propriedade **ClipControls** esteja configurada como **False**, os métodos gráficos no evento Paint afetam somente áreas recém-expostas do formulário; caso contrário, os métodos gráficos regeneram todas as áreas do formulário não cobertas por controles (exceto controles **Image**, **Label**, **Line** e **Shape**).

Usar um método **Refresh** em um procedimento de evento Resize regenera o objeto inteiro, toda vez que o usuário redimensiona o formulário.

**Observação:** Usar o procedimento de evento Paint para determinadas tarefas pode provocar um evento em cascata. Em geral, evita-se usar um procedimento de evento Paint para fazer o que se segue:

- Mover ou dimensionar um formulário ou controle.
- Alterar qualquer variável que afete o tamanho ou aparência como, por exemplo, configurar a propriedade **BackColor** de um objeto.
- Chamar um método **Refresh**.

Um procedimento de evento Resize pode ser mais adequado para algumas destas tarefas.

## Evento PathChange

Ocorre quando o caminho é alterado configurando-se a propriedade **FileName** ou **Path** no código.

#### Sintaxe

**Private Sub *object\_PathChange*([*index As Integer*])**

A sintaxe do evento PathChange tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	Um inteiro que identifica com exclusividade um controle, caso ele esteja em uma <u>matriz de controle</u> .

#### Comentários

Pode-se usar um procedimento de evento PathChange para responder a alterações de caminho em um controle **FileListBox**. Ao designar-se uma seqüência de caracteres contendo um novo caminho à propriedade **FileName**, o controle **FileListBox** chama o evento PathChange.

## Evento PatternChange

Ocorre quando o padrão de listagem de arquivos como, por exemplo, **"\*.\***", é alterado pela configuração da propriedade **FileName** ou **Pattern** no código.

#### Sintaxe

**Private Sub *object\_PatternChange*([*index As Integer*])**

A sintaxe do evento PatternChange tem estas partes:

Parte	Descrição
-------	-----------

<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	Um inteiro que identifica com exclusividade um controle, caso ele esteja em uma <u>matriz de controle</u> .

### Comentários

Pode-se usar um procedimento de evento para responder a alterações de padrão em um controle **FileListBox**. Ao se atribuir uma seqüência de caracteres contendo um novo padrão à propriedade **FileName**, **FileListBox** aciona o evento PathChange.

## Evento QueryUnload

Ocorre antes que um formulário ou aplicativo seja fechado. Quando um objeto **MDIForm** é fechado, o evento QueryUnload ocorre primeiro para o formulário MDI e, em seguida, para todos os formulários MDI filhos. Se nenhum formulário cancelar o evento QueryUnload, o evento Unload ocorrerá primeiro para todos os outros formulários e, em seguida, em um formulário MDI. Quando um formulário filho ou objeto **Form** é fechado, o evento QueryUnload naquele formulário ocorre antes do evento Unload do formulário.

### Sintaxe

**Private Sub Form\_QueryUnload(cancel As Integer, unloadmode As Integer)**

**Private Sub MDIForm\_QueryUnload(cancel As Integer, unloadmode As Integer)**

A sintaxe do evento QueryUnload tem estas partes:

<b>Parte</b>	<b>Descrição</b>
<i>cancel</i>	Um inteiro. A configuração deste argumento como qualquer valor diferente de 0, interrompe o evento QueryUnload em todos os formulários carregados e impede o fechamento do formulário e do aplicativo.
<i>Unloadmode</i>	Um valor ou constante indicando a causa do evento QueryUnload, conforme descrito em valores de retorno.

### Valores de retorno

O argumento *unloadmode* retorna os valores abaixo:

<b>Constante</b>	<b>Valor</b>	<b>Descrição</b>
<b>vbFormControlMenu</b>	0	O usuário escolheu o comando <b>Close</b> no menu <b>Control</b> do formulário.
<b>VbFormCode</b>	1	A instrução <b>Unload</b> é acionada a partir de código.
<b>VbAppWindows</b>	2	A sessão de ambiente operacional atual do Microsoft Windows está terminando.
<b>VbAppTasManager</b>	3	O <b>Gerenciador de Tarefas</b> do Microsoft Windows está fechando o aplicativos.
<b>vbFormMDIForm</b>	4	Um formulário MDI filho está fechando porque o formulário MDI está sendo fechado.

Estas constantes estão incluídas na biblioteca de objetos (VB) do Visual Basic no Object Browser.

### Comentários

Este evento é tipicamente usado para certificar-se de que não existam tarefas inacabadas nos formulários incluídos em um aplicativo, antes que este seja fechado. Por exemplo, caso um usuário ainda não tenha salvo novos dados em qualquer formulário, seu aplicativo poderá pedir ao usuário que salve os dados.

Quando um aplicativo fecha, pode-se utilizar o procedimento de evento QueryUnload ou Unload para configurar a propriedade **Cancel** como **True**, interrompendo o processo de fechamento. Entretanto, o evento QueryUnload ocorre em todos os formulários antes que qualquer um seja descarregado, e o evento Unload ocorre à medida que cada formulário é descarregado.

## Evento Resize

Ocorre quando um objeto é exibido pela primeira vez, ou quando o estado da janela de um objeto se altera. (Por exemplo, um formulário é maximizado, minimizado ou restaurado.)

### Sintaxe

**Private Sub Form\_Resize( )**

---

**Private Sub *object*\_Resize(*height* As Single, *width* As Single)**

A sintaxe do evento Resize tem estas partes:

<b>Parte</b>	<b>Descrição</b>
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>height</i>	Número especificando a nova altura do controle.
<i>Width</i>	Número especificando a nova largura do controle.

**Comentários**

Usa-se o procedimento de evento Resize para mover ou redimensionar controles quando o formulário pai é redimensionado. Pode-se também usar este procedimento de evento para recalcular variáveis e propriedades como, por exemplo, **ScaleHeight** e **ScaleWidth**, que podem depender do tamanho do formulário. Caso se queira que os elementos gráficos mantenham tamanhos proporcionais ao formulário ao serem redimensionados, aciona-se o evento Paint utilizando-se o método **Refresh** em um procedimento de evento Resize.

Sempre que a propriedade **AutoRedraw** é configurada como **False** e o formulário é redimensionado, o Visual Basic também chama os eventos relacionados, Resize e Paint, nesta ordem. Ao se anexar procedimentos a estes eventos relacionados, certifique-se de que suas ações não entrem em conflito.

Quando a propriedade **SizeMode** de um controle de recipiente **OLE** é configurada como 2 (Auto-size), o controle é automaticamente dimensionado de acordo com o tamanho de exibição do objeto contido no controle. Caso o tamanho de exibição do objeto seja alterado, o controle é automaticamente redimensionado para ajustar-se ao objeto. Quando isto ocorre, o evento Resize é acionado para o objeto, antes que o controle do recipiente **OLE** seja redimensionado. As partes *height* e *width* indicam o tamanho ideal para se exibir o objeto (este tamanho é determinado pelo aplicativo que o criou). Pode-se dimensionar o controle de maneira diferente alterando-se os valores das partes *height* e *width* no evento Resize.

## Evento RowColChange

Ocorre quanto a célula atual muda para uma célula diferente.

**Sintaxe**

**Private Sub *object*\_RowColChange ([*index* As Integer, *lastrow* As String, *lastcol* As Integer])**

A sintaxe do evento RowColChange tem estas partes:

<b>Parte</b>	<b>Descrição</b>
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	Um inteiro que identifica com exclusividade um controle, caso ele esteja em uma <u>matriz de controle</u> .
<i>lastrow</i>	(Para controle <b>DBGrid</b> ) Uma <u>expressão de sequência de caracteres</u> especificando a posição anterior da linha.
<i>lastcol</i>	(Para controle <b>DBGrid</b> ) Um inteiro especificando a posição anterior da coluna.

**Comentários**

Este evento ocorre toda vez que o usuário clica em uma célula diferente da célula atual, ou quando se altera por meio de programa a célula atual dentro de uma seleção usando as propriedades **Col** e **Row**.

O evento SelChange também ocorre quando o usuário clica em uma nova célula, mas não ocorre quando se altera por meio do programa o intervalo selecionado, sem alterar a célula atual.

Para o controle **DBGrid**, a posição da célula atual é fornecida pelas propriedades **Boomar** e **ColIndex**. A posição anterior da célula é especificada por *lastrow* e *lastcol*. Caso os dados sejam editados em seguida, movida a posição da célula atual para uma nova linha, os eventos de atualização para a linha original são terminados antes que outra célula torne-se a célula atual.

## Evento Scroll

Ocorre quando a caixa de rolagem em um controle **ScrollBar**, ou um objeto que contém uma barra de rolagem é reposicionado ou rolado horizontal e verticalmente.

**Sintaxe**

**Private Sub *dbgrid*\_Scroll([*cancel* As Integer])**

**Private Sub *object*\_Scroll( )**

---

A sintaxe do evento Scroll tem estas partes:

Part	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>cancel</i>	Determina se a operação de rolagem é bem sucedida, e se <b>ScrollBar</b> ou <b>DBGrid</b> são regeneradas, conforme descrito em Comentários.

### Comentários

Para um controle **DBGrid**, este evento ocorre quando o usuário rola a grade horizontal ou verticalmente, mas antes que a grade seja regenerada para exibir os resultados da operação de rolagem.

Para um controle **ComboBox**, este evento somente ocorre quando as barras de rolagem na parte drop-down do controle são manipuladas.

Configurar *cancel* como **True** provoca falha na operação de rolagem de **DBGrid** e não ocorre a operação de regeneração. Caso o método **Refresh** seja acionado dentro deste evento, a grade é regenerada em sua nova disposição (rolada), mesmo que *cancel* esteja configurado como **True**. Entretanto, neste caso, a grade é novamente regenerada porque a operação de rolagem falha, e ela volta à sua posição anterior.

Pode-se usar este evento para realizar cálculos ou para manipular controles que devem ser coordenados com alterações em curso nas barras de rolagem. Ao contrário, utiliza-se o evento **Change** quando se deseja que ocorra uma atualização apenas uma vez, após a alteração do controle **ScrollBar**.

**Observação:** Evite usar uma instrução ou função **MsgBox** neste evento.

## Evento SelChange

Ocorre quando o intervalo atual se altera para uma célula ou intervalo diferente de células.

### Sintaxe

**Private Sub DBGrid\_SelChange ([*cancel* As Integer])**

**Private Sub *object*\_SelChange( )**

A sintaxe do evento SelChange tem estas partes:

Part	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>cancel</i>	Determina se a seleção volta à posição anterior à ocorrência do evento.

### Comentários

O evento SelChange ocorre sempre que o usuário clica em uma célula diferente da célula atual, e à medida que um usuário arrasta para selecionar um novo intervalo de células. Um usuário também pode selecionar um intervalo de células pressionando a tecla SHIFT e usando as teclas de direção.

Pode-se disparar este evento via código para um controle **DBGrid**, alterando-se a área selecionada usando as propriedades **SelStartCol** e **SelEndCol**.

O evento RowColChange também ocorre quando um usuário clica em uma nova célula, mas não ocorre enquanto um usuário arrasta a seleção pelo controle **DBGrid**, ou quando se altera a seleção por meio do programa, sem mover a célula atual.

A configuração de *cancel* como **True** no controle **DBGrid** faz com que a seleção retorne à célula ou intervalo ativo antes da ocorrência do evento.

## Evento Timer

Ocorre quando um intervalo predefinido para um controle **Timer** tenha decorrido. A frequência do intervalo é armazenada na propriedade **Interval** do controle, que especifica a duração do tempo em milisegundos.

### Sintaxe

**Private Sub *object*\_Timer([*index* As Integer])**

A sintaxe do evento Timer tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	Um inteiro que identifica com exclusividade um controle, caso ele esteja em uma <u>matriz de controle</u> .

### Comentários

Usa-se este procedimento de evento para informar ao Visual Basic o que fazer após cada intervalo do controle **Timer** ter decorrido. Quando se está trabalhando com o evento Timer:

- A propriedade **Interval** especifica o intervalo entre eventos Timer em milissegundos.
- Toda vez que a propriedade **Enabled** do controle **Timer** é definida como **True** e a propriedade **Interval** é maior que 0, o evento Timer aguarda pelo período especificado na propriedade **Interval**.

## Evento Unload

Ocorre quando um formulário está para ser removido da tela. Quando este formulário é recarregado, o conteúdo de todos os seus controles é reinicializado. Este evento é desencadeado por um usuário fechando o formulário com o comando **Close** do menu **Control**, ou com uma instrução **Unload**.

### Sintaxe

**Private Sub** *object\_Unload*(*cancel* As Integer)

A sintaxe do evento Unload tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>Cancel</i>	Inteiro que determina se o formulário é removido da tela. Se <i>cancel</i> for 0, o formulário será removido. A configuração de <i>cancel</i> como qualquer valor diferente de zero impede a remoção do formulário.

### Comentários

A configuração de *cancel* como qualquer valor diferente de zero impede que o formulário seja removido, mas não interrompe outros eventos como, por exemplo, sair do ambiente operacional do Microsoft Windows. Usa-se o evento QueryUnload para impedir a saída do Windows.

Usa-se um procedimento de evento Unload para verificar se o formulário deve ser descarregado, ou para especificar ações que se queira que ocorram quando o formulário for descarregado. Pode-se ainda incluir qualquer código de validação a nível de formulário que seja necessário para fechar o formulário ou salvar os dados nele contidos para um arquivo.

O evento QueryUnload ocorre antes do evento Unload. O evento Unload ocorre antes do evento Terminate.

O evento Unload pode ser provocado pelo uso da instrução **Unload** ou pela escolha pelo usuário do comando **Close** em um menu **Controle** de formulário, saindo do aplicativo com o botão **Finalizar tarefa** na **Lista de tarefas** do Windows, fechando o formulário MDI do qual é filho o formulário atual, ou saindo do ambiente operacional do Microsoft Windows enquanto o aplicativo está sendo executado.



---

## Exemplo de eventos Activate, Deactivate

Este exemplo atualiza o texto na barra de status para exibir a legenda do formulário ativo. Para testar este exemplo, crie um objeto **Form** (Form1) e um novo objeto **MDIForm** (MDIForm1). No MDIForm1, desenhe um controle **PictureBox** contendo um controle **Label**. Em Form1, configure a propriedade **MDIChild** como **True**. Cole o código do procedimento de evento MDIForm\_Load na seção Declarations do objeto **MDIForm**. Cole o código do procedimento de evento Form\_Activate na seção Declarations do formulário filho MDI e, em seguida, pressione F5.

```
Private Sub MDIForm_Load ()
    Form1.Caption = "Form #1"           ' Definir a legenda de Form1.
    Dim NewForm As New Form1           ' Criar um novo formulário filho.
    Load NewForm
    NewForm.Caption = "Form #2"         ' Definir a legenda do novo formulário.
    NewForm.Show                       ' Exibir o novo formulário.
End Sub

Private Sub Form_Activate ()
    ' Definir o texto da barra de status.
    MDIForm1.Label1.Caption = "Current form: " & Me.Caption
End Sub
```

## Exemplo de evento Change

Este exemplo exibe a configuração numérica da propriedade **Value** de uma barra de rolagem horizontal em um controle **TextBox**. Para testar este exemplo, crie um formulário com um controle **TextBox** e um controle **HScrollBar** e, em seguida, cole o código na seção Declarations de um formulário que contenha uma barra de rolagem horizontal (controle **HScrollBar**) e um controle **TextBox**. Pressione F5 e clique na barra de rolagem horizontal.

```
Private Sub Form_Load ()
    HScroll1.Min = 0                   ' Definir Minimum.
    HScroll1.Max = 1000                 ' Definir Maximum.
    HScroll1.LargeChange = 100         ' Definir LargeChange.
    HScroll1.SmallChange = 1           ' Definir SmallChange.
End Sub

Private Sub HScroll1_Change ()
    Text1.Text = HScroll1.Value
End Sub
```

## Exemplo de evento Clic

Neste exemplo, cada vez que um controle **PictureBox** é clicado ele se move diagonalmente pelo formulário. Para testar este exemplo, cole o código na seção Declarations de um formulário que contenha um controle **PictureBox** posicionado no canto inferior esquerdo do formulário e, em seguida, pressione F5 e clique em **PictureBox**.

```
Private Sub Picture1_Clic ()
    Picture1.Move Picture1.Left + 750, Picture1.Top - 550
End Sub
```

## Exemplo de evento DragDrop

Este exemplo demonstra o efeito visual de se soltar um controle **PictureBox** sobre outro controle **PictureBox**. Para testar este exemplo, cole o código na seção Declarations de um formulário que contenha três controles **PictureBox**. Defina a propriedade **DragMode** de Picture1 e Picture2 como 1 (Automático). Use a propriedade **Picture** para atribuir bitmaps a Picture1 e Picture2 e, em seguida, pressione F5 e arraste Picture1 ou Picture2 sobre Picture3.

```
Private Sub Picture3_DragDrop (Source As Control, X As Single, Y As Single)
    If TypeOf Source Is PictureBox Then
        ' Definir o bitmap Picture3 ao mesmo controle de origem.
        Picture3.Picture = Source.Picture
    End If
End Sub
```

---

## Exemplo de evento DragOver

Este exemplo demonstra uma maneira de indicar um alvo válido para soltar. O ponteiro transforma-se, da seta padrão para um ícone especial quando um controle **TextBox** é arrastado sobre um controle **PictureBox**. O ponteiro retorna ao padrão quando a origem é arrastada para qualquer outro lugar. Para testar este exemplo, cole o código na seção Declarations de um formulário que contenha um pequeno **TextBox** e um **PictureBox**. Defina a propriedade **DragMode** do controle **TextBox** como 1 e, em seguida, pressione F5 e arraste o **TextBox** sobre **PictureBox**.

```
Private Sub PictureBox1_DragOver (Source As Control, X As Single, Y As Single, State As Integer)
    Select Case State
        Case vbEnter
            ' Carregar o ícone.
            Source.DragIcon = LoadPicture("ICONS\ARROWS\POINT03.ICO")
        Case vbLeave
            Source.DragIcon = LoadPicture() ' Descarregar o ícone.
    End Select
End Sub

Private Sub PictureBox1_DragDrop (Source As Control, X As Single, Y As Single)
    Source.DragIcon = LoadPicture() ' Descarregar o ícone.
End Sub
```

## Exemplo de evento DropDown

Este exemplo atualiza um controle **ComboBox** com base na seleção do usuário em um grupo de botões de opção. Para testar este exemplo, cole o código na seção Declarations de um formulário que contenha um controle **ComboBox** e dois controles **OptionButton**. Defina a propriedade **Name** de ambos os controles **OptionButton** como **OptionGroup** e, em seguida, pressione F5 e clique nos controles **OptionButton**. O controle **ComboBox** reflete diferentes couriers, dependendo do **OptionButton** selecionado.

```
Private Sub Form_Load ()
    Combol.Text = "" ' Limpar caixa de combinação.
End Sub

Private Sub Combol_DropDown ()
    Combol.Clear ' Excluir itens existentes.
    If OptionGroup(0).Value = True Then
        Combol.AddItem "Gray Goose Express", 0
        Combol.AddItem "Wild Fargo Carriers", 1
    Else
        Combol.AddItem "Summit Technologies Overnight"
    End If
End Sub
```

## Exemplo de evento GotFocus

Este exemplo exibe uma mensagem de barra de status quando um botão em um grupo **OptionButton** recebe o foco. Para testar este exemplo, cole o código na seção Declarations de um formulário que contenha dois controles **OptionButton** e um controle **Label**. Defina a propriedade **Name** para ambos os controles **OptionButton** como **OptionGroup** e, em seguida, pressione F5 e clique nos controles **OptionButton**.

```
Private Sub Form_Load ()
    Label1.AutoSize = True
End Sub

Private Sub OptionGroup1_GotFocus (Index As Integer)
    Select Case Index
        Case 0
            Label1.Caption = "Option 1 has the focus."
        Case 1
            Label1.Caption = "Option 2 has the focus."
    End Select
End Sub

Private Sub OptionGroup1_LostFocus (Index As Integer)
```

---

```
Label1.Caption = ""  
End Sub
```

## Exemplo de eventos eyDown, eyUp

Este exemplo demonstra um identificador genérico de teclado que responde à tecla F2 e a todas as combinações de tecla ALT, SHIFT e CTRL associadas. As constantes de tecla estão listadas na biblioteca de objetos (VB) do Visual Basic no Object Browser. Para testar este exemplo, cole o código na seção Declarations de um formulário que contenha um controle **TextBox** e, em seguida, pressione F5 e pressione F2 com diversas combinações das teclas ALT, SHIFT e CTRL.

```
Private Sub Text1_eyDown (eyCode As Integer, Shift As Integer)  
    Dim ShiftDown, AltDown, CtrlDown, Txt  
    ShiftDown = (Shift And vbShiftMas) > 0  
    AltDown = (Shift And vbAltMas) > 0  
    CtrlDown = (Shift And vbCtrlMas) > 0  
    If eyCode = vbeyF2 Then ' Exibe as combinações de teclas.  
        If ShiftDown And CtrlDown And AltDown Then  
            Txt = "SHIFT+CTRL+ALT+F2."  
        ElseIf ShiftDown And AltDown Then  
            Txt = "SHIFT+ALT+F2."  
        ElseIf ShiftDown And CtrlDown Then  
            Txt = "SHIFT+CTRL+F2."  
        ElseIf CtrlDown And AltDown Then  
            Txt = "CTRL+ALT+F2."  
        ElseIf ShiftDown Then  
            Txt = "SHIFT+F2."  
        ElseIf CtrlDown Then  
            Txt = "CTRL+F2."  
        ElseIf AltDown Then  
            Txt = "ALT+F2."  
        ElseIf SHIFT = 0 Then  
            Txt = "F2."  
        End If  
        Text1.Text = "You pressed. " & Txt  
    End If  
End Sub
```

## Exemplo de evento eyPress

Este exemplo converte o texto inserido em um controle **TextBox** em maiúsculas. Para testar este exemplo, cole o código na seção Declarations de um formulário que contenha um **TextBox** e, em seguida, pressione F5 e digite alguma coisa no **TextBox**.

```
Private Sub Text1_eyPress (eyAscii As Integer)  
    Char = Chr(eyAscii)  
    eyAscii = Asc(UCase(Char))  
End Sub
```

## Exemplo de evento LinError

Este exemplo é anexado a um controle **TextBox**, MyTextBox, que trata erros selecionados. O procedimento exibe uma mensagem (adaptada da lista de erros no tópico de evento LinErros) com base no número de erro passado como argumento LinErr. Pode-se adaptar este código para um formulário de origem substituindo Form\_LinError por MyTextBox\_LinError. Este exemplo é apenas para ilustração.

```
Private Sub MyTextBox_LinError (LinErr As Integer)  
    Dim Msg  
    Select Case LinErr  
        Case 1  
            Msg = "Data in wrong format."  
        Case 11  
            Msg = "Out of memory for DDE."  
    End Select  
    MsgBox Msg, vbExclamation, "MyTextBox"  
End Sub
```

---

## Exemplo de evento LinExecute

Este exemplo define um conjunto de comandos para destinos usarem em conversação DDE que será respondida por seu aplicativo. Este exemplo é somente para ilustração.

```
Private Sub Form_LinExecute (CmdStr As String, Cancel As Integer)
    Cancel = False
    Select Case LCase(CmdStr)
    Case "{big}"
        WindowState = 2      ' Maximizar window.
    Case "{little}"
        WindowState = 1      ' Minimizar window.
    Case "{hide}"
        Visible = False      ' Ocultar formulário.
    Case "{view}"
        Visible = True        ' Exibir formulário.
    Case Else
        Cancel = True         ' A execução não é permitida.
    End Select
End Sub
```

## Exemplo de evento LinNotify

Este exemplo está anexado a um controle **PictureBox**, Picture1, que tem suas propriedades **LinTopic** e **LinItem** definidos para especificar um elemento gráfico na origem, e sua propriedade **LinMode** definida como 3 (Notificar). Quando a origem altera estes dados, o procedimento atualiza imediatamente o controle **PictureBox** somente se **PictureBox** está no formulário ativo; caso contrário ele define um sinalizador de variável. Este exemplo é somente para ilustração.

```
Private Sub Picture1_LinNotify ()
    If Screen.ActiveForm Is Me Then
        Picture1.LinRequest      ' A figura está no formulário ativo, portanto
        atualize.
    Else
        NewDataFlag = True      ' Pressupõe-se que seja uma variável em nível
        de módulo.
    End If
End Sub
```

## Exemplo de evento Load

Este exemplo carrega itens em um controle **ComboBox** quando um formulário é carregado. Para testar este exemplo, cole o código na seção Declarations de um formulário que contenha uma **ComboBox** e, em seguida, pressione F5.

```
Private Sub Form_Load ()
    Combo1.AddItem "Mozart"      ' Adicionar itens à lista.
    Combo1.AddItem "Beethoven"
    Combo1.AddItem "Roc 'n Roll"
    Combo1.AddItem "Reggae"
    Combo1.ListIndex = 2         ' Definir a seleção padrão.
End Sub
```

---

## Exemplo de evento LostFocus

Este exemplo altera a cor de um controle **TextBox** quando ele recebe ou perde o foco (selecioneado com o mouse ou com a tecla TAB) e exibe o texto adequado. Para testar este exemplo, cole o código na seção Declarations de um formulário que contenha dois controles **TextBox** e um controle **Label** e, em seguida, pressione F5 e mova o foco entre Text1 e Text2.

```
Private Sub Text1_GotFocus ()
    ' Mostrar o foco com vermelho.
    Text1.BackColor = RGB(255, 0, 0)
    Label1.Caption = "Text1 has the focus."
End Sub

Private Sub Text1_LostFocus ()
    ' Mostre a perda do foco com azul.
    Text1.BackColor = RGB(0, 0, 255)
    Label1.Caption = "Text1 doesn't have the focus."
End Sub
```

## Exemplo de eventos MouseDown, MouseUp

Este exemplo demonstra um aplicativo de pintura simples. O procedimento de evento MouseDown funciona com um procedimento de evento MouseMove relacionado, para permitir a pintura quando qualquer botão do mouse é pressionado e arrastado. O procedimento de evento MouseUp desativa a pintura. Para testar este exemplo, cole o código na seção Declarations de um formulário e, em seguida, pressione F5, clique no formulário e mova o mouse enquanto o botão do mouse está pressionado.

```
Dim PaintNow As Boolean
Private Sub Form_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
    PaintNow = True    ' Ativar a pintura.
End Sub

Private Sub Form_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)
    PaintNow = False   ' Desativar a pintura.
End Sub

Private Sub Form_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)
    If PaintNow Then
        PSet (X, Y)    ' Desenhar um ponto.
    End If
End Sub

Private Sub Form_Load ()
    DrawWidth = 10      ' Usar um pincel mais largo.
    ForeColor = RGB(0, 0, 255)    ' Definir a cor da pintura.
End Sub
```

## Exemplo de evento MouseMove

Este exemplo demonstra um aplicativo simples de pintura. O procedimento de evento MouseDown funciona com um procedimento de evento MouseMove para permitir a pintura quando qualquer botão do mouse estiver pressionado. O procedimento de evento MouseUp desativa a pintura. Para testar este exemplo, cole o código na seção Declarations de um formulário e, em seguida, pressione F5, clique no formulário e mova o mouse enquanto o botão do mouse está pressionado.

```
Dim PaintNow As Boolean ' Declara variável.
Private Sub Form_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
    PaintNow = True    ' Pincel ativado.
End Sub

Private Sub Form_MouseUp (Button As Integer, X As Single, Y As Single)
    PaintNow = False   ' Desativar pintura.
End Sub
```

---

```

Private Sub Form_MouseMove (Button As Integer, Shift As Integer, X As
Single, Y As Single)
    If PaintNow Then
        PSet (X, Y)      ' Desenhando um ponto.
    End If
End Sub

Private Sub Form_Load ()
    DrawWidth = 10      ' Usar pincel mais largo.
    ForeColor = RGB(0, 0, 255)      ' Definir a cor de desenho.
End Sub

```

## Exemplo de evento Paint

Este exemplo desenha um diamante que intercepta o ponto médio de cada lado de um formulário e se ajusta automaticamente, à medida que o formulário é redimensionado. Para testar este exemplo, cole o código na seção Declarations de um formulário e, em seguida, pressione F5 e redimensione o formulário.

```

Private Sub Form_Paint ()
    Dim HalfX, HalfY      ' Declarar variáveis.
    HalfX = ScaleLeft + ScaleWidth / 2      ' Definir com metade da largu-
ra.
    HalfY = ScaleTop + ScaleHeight / 2      ' Definir como metade da altu-
ra.
    ' Desenhando um diamante.
    Line (ScaleLeft, HalfY) - (HalfX, ScaleTop)
    Line -(ScaleWidth + ScaleLeft, HalfY)
    Line -(HalfX, ScaleHeight + ScaleTop)
    Line -(ScaleLeft, HalfY)
End Sub

Private Sub Form_Resize
    Refresh
End Sub

```

## Exemplo de evento PathChange

Este exemplo demonstra como atualizar um controle **Label** para refletir o caminho atual até um controle **FileListBox**. Clicando-se duas vezes em um nome de diretório, é exibida uma lista dos arquivos do diretório na **FileListBox**; ela também exibe o caminho completo do diretório no controle **Label**. Para testar este exemplo, cole o código na seção Declarations de um formulário que contenha o controle **Label**, um controle **DirListBox** e um controle **FileListBox**, e, em seguida, pressione F5. Clique duas vezes em um diretório para alterar o caminho.

```

Private Sub File1_PathChange ()
    Label1.Caption = "Path: " & Dir1.Path      ' Mostrar o caminho em Label.
End Sub

Private Sub Dir1_Change ()
    File1.Path = Dir1.Path                      ' Definir o caminho do arquivo.
End Sub

Private Sub Form_Load ()
    Label1.Caption = "Path: " & Dir1.Path      ' Mostrar o caminho em Label.
End Sub

```

---

## Exemplo de evento PatternChange

Este exemplo atualiza um controle **FileListBox** contendo arquivos que correspondem ao padrão inserido em um controle **TextBox**. Caso um caminho completo seja inserido na **TextBox**, por exemplo, C:\BIN\\*.EXE, o texto é automaticamente dividido nos componentes padrão e caminho. Para testar este exemplo, cole o código na seção Declarations de um formulário que contenha um controle **TextBox**, um controle **Label**, um controle **FileListBox** e um controle **CommandButton** e, em seguida, pressione F5 e digite um padrão válido de arquivo em **TextBox**.

```
Private Sub Form_Load ()
    Command1.Default = True           ' Definir a propriedade Default.
    Command1.Caption = "O"           ' Definir Caption.
End Sub

Private Sub Command1_Click () ' Botão O clicado.
    ' O texto é dividido nos componentes caminho e padrão.
    File1.FileName = Text1.Text
    Label1.Caption = "Path: " & File1.Path
End Sub

Private Sub File1_PatternChange ()
    Text1.Text = File1.Pattern       ' Definir o Text como o novo padrão.
End Sub
```

## Exemplo de evento QueryUnload

Este exemplo utiliza um objeto **MDIForm** contendo dois formulários filhos MDI. Ao se escolher o comando **Close** no menu **Control** para fechar o formulário, uma mensagem é exibida diferente da que seria exibida se fosse escolhido o comando **Exit** do menu **File**. Para testar este exemplo, crie um **MDIForm** e, em seguida, utilize o Menu Editor para criar um menu **File** contendo um comando **Exit** chamado **FileExit**. Certifique-se de que este item de menu esteja ativo. No Form1, defina a propriedade **MDIChild** como **True**. Cole o código nas seções Declarations dos respectivos formulários e, em seguida, pressione F5 para executar o programa.

```
' Colar na seção Declarations de MDIForm1.
Private Sub MDIForm_Load ()
    Dim NewForm As New Form1 ' Nova ocorrência de Form1.
    NewForm.Caption = "Form2" ' Definir a legenda e exibir.
End Sub

Private Sub FileExit_Click ()
    Unload MDIForm1 ' Sair do aplicativo.
End Sub

Private Sub MDIForm_QueryUnload (Cancel As Integer, UnloadMode As Integer)
    Dim Msg ' Declarar a variável.
    ' Definir o texto da mensagem.
    Msg = "Do you really want to exit the application?"
    ' Se o usuário clica no botão No, encerre QueryUnload.
    If MsgBox(Msg, vbQuestion + vbYesNo, Me.Caption) = vbNo Then Cancel = True
End Sub

' Colar na seção Declarations de Form1.
Private Sub Form_QueryUnload (Cancel As Integer, UnloadMode As Integer)
    Dim Msg ' Declarar a variável.
    If UnloadMode > 0 Then
        ' Se estiver saindo do aplicativo.
        Msg = "Do you really want to exit the application?"
    Else
        ' Se estiver fechando o formulário.
        Msg = "Do you really want to close the application?"
    End If
    ' Se o usuário clicar no botão No, interrompa QueryUnload.
    If MsgBox(Msg, vbQuestion + vbYesNo, Me.Caption) = vbNo Then Cancel = True
End Sub
```

---

## Exemplo de evento Resize

Este exemplo redimensiona automaticamente um controle **TextBox** para preencher o formulário sempre que o formulário for redimensionado. Para testar este exemplo, cole o código na seção Declarations de um formulário de propriedade **MultiLine** que contenha uma **TextBox**. Defina a propriedade **MultiLine** do controle **TextBox** como **True**, sua propriedade **ScrollBars** como 3 e sua propriedade **BorderStyle** como 0; em seguida, pressione F5 e redimensione o formulário.

```
Private Sub Form_Load ()
    Text1.Text = "" ' Limpar o texto.
End Sub

Private Sub Form_Resize ()
    Text1.Move 0,0, ScaleWidth, ScaleHeight
End Sub
```

## Exemplo de evento Scroll

Este exemplo altera o tamanho de um controle **Shape** para corresponder ao valor de uma barra de rolagem horizontal (**HScrollBar**) à medida que você arrasta a caixa de rolagem pela barra de rolagem. Para testar este exemplo, cole o código na seção Declarations de um formulário contendo um controle **Shape**, um controle **Label** e um controle **HScrollBar**. Defina a propriedade **Index** do controle **Shape** como 0 para criar uma matriz de controle. A seguir, pressione F5 e mova a barra de rolagem.

```
Private Sub Form_Load ()
    ' Mover e dimensionar o primeiro controle Shape.
    Shape1(0).Move HScroll1.Left, HScroll1.Top * 1.5, HScroll1.Width,
HScroll1.Height
    Label1.Caption = "" ' Definir a legenda de Label.
    Load Shape1(1) ' Criar um segundo Shape.
    ' Mover e dimensionar o segundo controle Shape.
    Shape1(1).Move Shape1(0).Left, Shape1(0).Top, 1, Shape1(0).Height
    Shape1(1).BacStyle = 1 ' Definir BacStyle como Opaco.
    Shape1(1).Visible = True ' Exibir a segunda Shape.
    HScroll1.Min = 1 ' Definir os valores da barra de rolagem.
    HScroll1.Max = HScroll1.Width
End Sub

Private Sub HScroll1_Change ()
    Label1.Caption = "Changed" ' Exibir mensagem após a alteração.
End Sub

Private Sub HScroll1_Scroll ()
    Shape1(1).BackColor = &HFF0000 ' Definir a cor de Shape como Azul.
    Label1.Caption = "Changing" ' Exibir mensagem durante a rolagem.
    Shape1(1).Width = HScroll1.Value ' Dimensionar Shape como Scroll Value.
End Sub
```

## Exemplo de evento Timer

Este exemplo demonstra um relógio digital. Para testar este exemplo, cole o código na seção Declarations de um formulário que contenha um controle **Label** e um controle **Timer** e, em seguida, pressione F5.

```
Private Sub Form_Load ()
    Timer1.Interval = 1000 ' Definir o intervalo de Timer.
End Sub

Private Sub Timer1_Timer ()
    Label1.Caption = Time ' Atualizar a exibição da hora.
End Sub
```

Este exemplo move um controle **PictureBox** por um formulário. Para testar este exemplo, cole o código na seção Declarations de um formulário que contenha um controle **Timer** e um controle **PictureBox** e, em seguida, pressione F5. Para um efeito visual melhor, pode-se atribuir um bitmap à **PictureBox** usando a propriedade **Picture**.

```
Dim DeltaX, DeltaY As Integer ' Declarar variáveis.
Private Sub Timer1_Timer ()
```



```

Picture1.Move Picture1.Left + DeltaX, Picture1.Top + DeltaY
If Picture1.Left < ScaleLeft Then DeltaX = 100
If Picture1.Left + Picture1.Width > ScaleWidth + ScaleLeft Then
    DeltaX = -100
End If
If Picture1.Top < ScaleTop Then DeltaY = 100
If Picture1.Top + Picture1.Height > ScaleHeight + ScaleTop Then
    DeltaY = -100
End If
End Sub

Private Sub Form_Load ()
    Timer1.Interval = 1000 ' Definir Interval.
    DeltaX = 100 ' Inicializar variáveis.
    DeltaY = 100
End Sub

```

## Exemplo de evento Unload

Este exemplo demonstra um procedimento simples para fechar um formulário ,enquanto são apresentadas ao usuário diversas caixas de mensagem. Em um aplicativo real, pode-se adicionar chamadas a procedimentos Sub de uso geral que emulem o processamento dos comandos Exit, Save e Save As do menu File do Visual Basic. Para testar este exemplo, cole o código na seção Declarations de um formulário e, em seguida, pressione F5. Uma vez que o formulário tenha sido exibido, pressione ALT+F4 para fechá-lo.

```

Private Sub Form_Unload (Cancel As Integer)
    Dim Msg, Response ' Declarar variáveis.
    Msg = "Save Data before closing?"
    Response = MsgBox(Msg, vbQuestion + vbYesNoCancel, "Save Dialog")
    Select Case Response
        Case vbCancel ' Não permitir o fechamento.
            Cancel = -1
            Msg = "Command has been closed."
        Case vbYes
            ' Digite o código para salvar dados aqui.
            Msg = "Data saved."
        Case vbNo
            Msg = "Data not saved."
    End Select
    MsgBox Msg, vbOnly, "Confirm" ' Exibir mensagem.
End Sub

```

## Evento ItemChec

Ocorre quando a propriedade **Style** de um controle **ListBox** é configurada como 1 (caixas de seleção) e a caixa de seleção de um item no controle **ListBox** está selecionada ou limpa.

### Sintaxe

**Private Sub** *object\_ItemChec*(*[index As Integer]*)

A sintaxe do evento ItemChec tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	Um inteiro que identifica com exclusividade o item na caixa de listagem que foi clicado.

### Comentários

**Observação:** O evento ItemChec não ocorre quando um item de lista está apenas selecionado; ao invés, ele ocorre quando a caixa de seleção do item da lista está selecionada ou em branco.

O evento ItemChec também pode ocorrer por meio de programa sempre que um elemento em uma matriz Selected da **ListBox** é alterado (e sua propriedade **Style** é configurada como 1.) O evento ItemChec ocorre antes do evento Clic.

---

## Objeto Screen

Manipula formulários de acordo com sua posição na tela e controla o ponteiro do mouse fora dos formulários de seu aplicativo durante o tempo de execução. O objeto **Screen** é acessado com a palavra-chave **Screen**.

### Sintaxe

#### Screen

### Comentários

O objeto **Screen** é a área de trabalho inteira do Windows. Usando o objeto **Screen**, você pode configurar a propriedade **MousePointer** do objeto **Screen** como a ampulheta ponteiro enquanto um formulário de janela estrita é exibido.

## Método Refresh

Força uma atualização gráfica completa de um formulário ou controle.

### Sintaxe

*object.Refresh*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Use o método **Refresh** quando quiser:

- Exibir completamente um formulário enquanto outro formulário está sendo carregado.
- Atualizar o conteúdo de uma caixa de listagem do sistema de arquivos como, por exemplo, o controle **FileListBox**.
- Atualizar as estruturas de dados de um controle **Data**.

**Refresh** não pode ser usada em formulários MDI, mas pode ser usado em formulários MDI filho.

Você não pode usar **Refresh** em controles **Menu** ou **Timer**.

Geralmente, a pintura de um formulário ou controle é tratada automaticamente enquanto nenhum evento está ocorrendo. Entretanto, podem ocorrer situações onde você deseje que o formulário ou controle seja atualizado imediatamente. Por exemplo, se você usar uma caixa de listagem de arquivo, uma caixa de listagem de diretório ou uma caixa de listagem de drive para exibir o status atual da estrutura do diretório, você pode usar **Refresh** para atualizar a lista toda vez que ocorrer uma alteração na estrutura do diretório.

Você pode usar o método **Refresh** em um controle **Data** para abrir ou reabrir o banco de dados (se as configurações da propriedade **DatabaseName**, **ReadOnly**, **Exclusive** ou **Connect** tiverem sofrido alteração) e recriar o dynaset na propriedade **Recordset** do controle.

## Exemplo do método Refresh

Este exemplo usa o método **Refresh** para atualizar um controle **FileListBox** à medida que arquivos de teste são criados. Para experimentar este exemplo, cole o código na seção Declarations de um formulário com um controle **FileListBox** chamado File1 e, em seguida, execute o exemplo e clique no formulário.

```
Private Sub Form_Click ()
    ' Declarar variáveis.
    Dim FilName, Msg as String, I as Integer
    File1.Pattern = "TestFile.*"      ' Definir o padrão do arquivo.
    For I = 1 To 8 ' Executar oito vezes.
        FilName = "TESTFILE." & I
        ' Criar um arquivo vazio.
        Open FilName For Output As FreeFile
        File1.Refresh ' Atualizar a caixa de listagem de arquivos.
        Close ' Fechar o arquivo.
    Next I
    Msg = "Escolher O para remover os arquivos de teste criados."
    MsgBox Msg ' Exibir a mensagem.
    Kill "TESTFILE.*" ' Remover os arquivos de teste.
    File1.Refresh ' Atualizar a caixa de listagem de arquivos.
End Sub
```

---

## Método SetFocus

Move o foco para o controle ou formulário especificado.

### Sintaxe

*object*.SetFocus

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

O objeto precisa ser um objeto **Form**, objeto **MDIForm** ou controle que possa receber o foco. Após acionar o método **SetFocus**, qualquer entrada do usuário é direcionada ao formulário ou controle específico.

Você somente pode mover o foco para um formulário ou controle visível quando um formulário e controles em um formulário não estão visíveis, até que o evento Load do formulário tenha terminado, você não pode usar o método **SetFocus** para mover o foco para o formulário que está sendo carregado em seu próprio evento Load, a menos que primeiro você utilize o método **Show** para exibir o formulário, antes que o procedimento de evento Form\_Load tenha terminado.

Você também não pode mover o foco para um formulário ou controle se a propriedade **Enabled** estiver configurada como **False**. Se a propriedade **Enabled** estiver configurada como **False** durante o tempo de criação, você deve primeiro configurá-la como **True** antes que ela possa receber o foco, usando o método **SetFocus**.

## Evento DblClic

Ocorre quando o usuário pressiona e solta o botão do mouse e, em seguida, pressiona e solta novamente sobre um objeto.

Para um formulário, o evento DblClic ocorre quando o usuário clica duas vezes em um controle desativado ou área vazia de um formulário. Para um controle, ele ocorre quando o usuário:

- Clica duas vezes em um controle com o botão esquerdo do mouse.
- Clica duas vezes em um item em um controle **ComboBox** cuja propriedade **Style** esteja configurada como 1 (Simples) ou em um controle **FileListBox**, **ListBox**, **DBCombo** ou **DBList**.

### Sintaxe

**Private Sub** Form\_DblClic ( )

**Private Sub** *object\_DblClic* (*index As Integer*)

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	Identifica o controle se ele estiver em uma <u>matriz de controle</u> .

### Comentários

O argumento *Index* identifica com exclusividade um controle se ele estiver em uma matriz de controle. Você pode usar um procedimento de evento DblClic para uma ação implícita como, por exemplo, clicar duas vezes em um ícone para abrir uma janela ou documento. Você também pode usar este tipo de procedimento para executar múltiplas etapas com uma única ação, como clicar duas vezes para selecionar um item em uma caixa de listagem e para fechar a caixa de diálogo.

Para produzir estes efeitos de atalho em Visual Basic, você pode usar um procedimento de evento DblClic para uma caixa de listagem ou arquivo de caixa de listagem em conjunto com um botão padrão — um controle **CommandButton** com sua propriedade **Default** configurada como **True**. Como parte do procedimento de evento DblClic para a caixa de listagem, você simplesmente chama o evento Clic do botão padrão.

Para aqueles objetos que recebem eventos de mouse, os eventos ocorrem nesta ordem: MouseDown, MouseUp, Clic, DblClic e MouseUp.

Se DblClic não ocorrer dentro do limite de tempo de dupla clicagem do sistema, o objeto reconhecerá outro evento Clic. O limite de tempo de dupla clicagem pode variar porque o usuário pode configurar a velocidade de dupla clicagem no **Painel de controle**. Quando você está anexando procedimentos para estes eventos relacionados, certifique-se de que suas ações não entrem em conflito. Os controles que não recebem eventos DblClic podem receber dois cliques ao invés de um DblClic.

**Observação:** Para distinguir entre os botões esquerdo, direito e central do mouse, use os eventos MouseDown e MouseUp.

Se existir código no evento Clic, o evento DblClic nunca será disparado.

---

## Exemplo do evento DblClic

Este exemplo exibe um item de lista selecionado em um controle **TextBox** quando o controle **CommandButton** é clicado ou quando um item de lista é clicado duas vezes. Para experimentar este exemplo, cole o código na seção Declarations de um objeto **Form** que contenha um controle **ListBox**, um controle **TextBox** e um controle **CommandButton**. Em seguida, execute o exemplo e clique no controle **CommandButton** ou clique duas vezes em um item do controle **ListBox**.

```
Private Sub Form_Load ()
    List1.AddItem "John" ' Adicionar entradas na caixa de listagem.
    List1.AddItem "Paul"
    List1.AddItem "George"
    List1.AddItem "Ringo"
End Sub

Private Sub List1_DblClic ()
    Command1.Value = True ' Disparar o evento Clic.
End Sub

Private Sub Command1_Clic ()
    Text1.Text = List1.Text ' Exibir a seleção.
End Sub
```

## Propriedade HelpFile (App, CommonDialog, MenuLine)

Especifica o caminho e o nome de um arquivo da **Ajuda** do Microsoft Windows usado pelo seu aplicativo para exibir a **Ajuda** ou documentação on-line.

### Sintaxe

*object.HelpFile* [ = *filename* ]

A sintaxe da propriedade **HelpFile** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>filename</i>	Uma <u>expressão de sequência de caracteres</u> especificando o caminho e nome de arquivo do arquivo da <b>Ajuda</b> do Windows para seu aplicativo.

### Comentários

Se você criou um arquivo da **Ajuda** do Windows para o seu aplicativo e configurou a propriedade **HelpFile** do aplicativo, o Visual Basic chama automaticamente a **Ajuda** quando o usuário pressiona a tecla F1. Se existir um número de contexto na propriedade **HelpContextID** para o controle ativo ou para o formulário ativo, a **Ajuda** exibirá um tópico correspondente ao contexto da **Ajuda** atual. De outra forma, ela exibirá a tela de conteúdo principal.

Você também pode usar a propriedade **HelpFile** para determinar qual arquivo da **Ajuda** será exibido quando o usuário pedir **Ajuda** no **Object Browser** para um componente ActiveX.

**Observação:** Criar um arquivo da **Ajuda** exige o Microsoft Windows Help Compiler, que está disponível com o Visual Basic, Professional Edition.

## Exemplo da propriedade HelpFile

Este exemplo usa tópicos no arquivo do Visual Basic Help e demonstra como especificar números de contexto para tópicos da **Ajuda**. Para experimentar este exemplo, cole o código na seção Declarations de um objeto **Form** que contenha um controle **TextBox** e um controle **Frame** com um controle **OptionButton** dentro dele. Execute o exemplo. Uma vez que o programa esteja sendo executado, mova o foco até um dos componentes e pressione F1.

```
' Os números de contexto efetivos do arquivo do Visual Basic Help.
' Definir constantes.
Const winPictureBox = 2016002
Const winCommandButton = 2007557
```

```
Private Sub Form_Load ()
```

---

```
App.HelpFile = "VB5.HLP"  
Text1.HelpContextID = winPictureBox  
Form1.HelpContextID = winCommandButton  
End Sub
```

## Objeto Clipboard

Proporciona o acesso à Área de transferência do sistema.

### Sintaxe

#### Clipboard

### Comentários

O objeto **Clipboard** é usado para manipular texto e elementos gráficos na **Área de transferência**. Você pode usar este objeto para permitir que um usuário copie, recorte e cole texto ou elementos gráficos em seu aplicativo. Antes de copiar qualquer material para o objeto **Clipboard**, você deve limpar seu conteúdo executando um método **Clear** como, por exemplo, `Clipboard.Clear`.

Observe que o objeto **Clipboard** é compartilhado por todos os aplicativos do Windows e, portanto, o conteúdo está sujeito a alteração quando se alterna para outro aplicativo.

O objeto **Clipboard** pode conter diversos fragmentos de dados, desde que cada fragmento esteja em um formato diferente. Por exemplo você pode usar o método **SetData** para colocar um bitmap na **Área de transferência** com o formato **vbCFDIB** e, em seguida, usar o método **SetText** com o formato **vbCFTText** para colocar texto na **Área de transferência**. Em seguida, você pode usar o método **GetText** para recuperar o texto ou o método **GetData** para recuperar o elemento gráfico. Dados na **Área de transferência** estarão perdidos quando outro conjunto de dados do mesmo formato for colocado na **Área de transferência** através de código ou de um comando de menu.

## Objeto Printer, Coleção Printers

O objeto **Printer** permite que você se comunique com uma impressora do sistema (inicialmente a impressora padrão do sistema).

A coleção **Printers** permite que você reúna informações sobre as impressoras disponíveis no sistema.

### Sintaxe

#### Printer

#### Printers(*index*)

O espaço reservado *index* representa um número inteiro com um intervalo de 0 a `Printers.Count-1`.

### Comentários

Use métodos gráficos para desenhar texto ou elementos gráficos no objeto **Printer**. Uma vez que o objeto **Printer** contém o resultado que você deseja imprimir, você pode usar o método **EndDoc** para enviar o resultado diretamente à impressora padrão do aplicativo.

Você deve verificar, e possivelmente revisar o layout de seus formulários se você os imprime. Se você usa o método **PrintForm** para imprimir um formulário, por exemplo, imagens gráficas podem ser recortadas na parte inferior da página, e o texto transportado para a página seguinte.

A coleção **Printers** permite que você consulte as impressoras disponíveis, de modo que se possa especificar uma impressora padrão para seu aplicativo. Por exemplo, caso você queira descobrir qual das impressoras disponíveis utiliza um driver de impressora específico. O código abaixo pesquisa todas as impressoras disponíveis para localizar a primeira impressora com sua orientação de página configurada como retrato, e a define como impressora padrão:

```
Dim X As Printer  
For Each X In Printers  
    If X.Orientation = vbPRORPortrait Then  
        ' Configurar impressora como padrão do sistema.  
        Set Printer = X  
        ' Encerrar a procura de impressora.  
        Exit For  
    End If  
Next
```

---

Você designa uma das impressoras da coleção **Printers** como impressora padrão usando a instrução **Set**. O exemplo anterior designa a impressora identificada pela variável de objeto **X**, a impressora padrão do aplicativo.

**Observação:** Se você usa a coleção **Printers** para especificar uma determinada impressora, como em `Printers(3)`, você somente pode acessar propriedades somente para leitura. Para ler e gravar as propriedades de uma impressora individual, você deve primeiro tornar esta impressora a impressora padrão do aplicativo.

## Objeto Font

O objeto **Font** contém informações necessárias para formatar o texto para ser exibido na interface de um aplicativo ou para resultado impresso.

### Sintaxe

#### Font

### Comentários

Com frequência, você identifica um objeto **Font** usando a propriedade **Font** de um objeto que exibe texto (por exemplo, um objeto **Form** ou o objeto **Printer**).

Você não pode criar um objeto **Font** usando código do tipo `Dim X As New Font`. Se você deseja criar um objeto **Font**, deve usar o objeto **StdFont** assim:

```
Dim X As New StdFont
```

Se você coloca um controle **TextBox** chamado `Text1` em um formulário, pode alterar dinamicamente seu objeto **Font** para outro usando a instrução **Set**, como no exemplo abaixo:

```
Dim X As New StdFont
X.Bold = True
X.Name = "Arial"
Set Text1.Font = X
```

## Instrução Load

Carrega um formulário ou controle na memória.

### Sintaxe

#### Load *object*

O espaço reservado *object* é o nome de um objeto **Form**, objeto **MDIForm** ou elemento de matriz de controle a ser carregado.

### Comentários

Você não precisa usar a instrução **Load** com formulários, a menos que queira carregar um formulário sem exibi-lo. Qualquer referência a um formulário (exceto em uma instrução **Set** ou **If...TypeOf**) carrega-o automaticamente, se ele ainda não tiver sido carregado. Por exemplo, o método **Show** carrega um formulário antes de exibi-lo. Uma vez que o formulário tenha sido carregado, suas propriedades e controles podem ser alterados pelo aplicativo, esteja o formulário visível ou não. Sob determinadas circunstâncias, você pode carregar todos os seus formulários durante a inicialização e os exibir posteriormente, à medida que forem necessários.

Quando o Visual Basic carrega um objeto **Form**, ele configura as propriedades do formulário como seus valores iniciais e, em seguida, executa o procedimento de evento `Load`. Quando o aplicativo é iniciado, o Visual Basic carrega automaticamente e exibe o formulário de inicialização do aplicativo.

Se você carrega um **Form** cuja propriedade **MDIChild** esteja configurada como **True** (em outras palavras, o formulário filho) antes de carregar um **MDIForm**, o **MDIForm** é automaticamente carregado antes do formulário filho. Formulários MDI filho não podem ser ocultados e, assim, são imediatamente visíveis após o final do procedimento de evento `Form_Load`.

As caixas de diálogo padrão produzidas por funções do Visual Basic como, por exemplo **MsgBox** e **InputBox** não precisam ser carregadas, exibidas ou descarregadas, mas podem simplesmente ser acionadas de maneira direta.

## Exemplo da instrução Load

Este exemplo usa a instrução **Load** para carregar um objeto **Form**. Para experimentar este exemplo, cole o código na seção `Declarations` de um objeto **Form** e, em seguida, execute o exemplo e clique no objeto **Form**.

---

```

Private Sub Form_Clic ()
    Dim Answer, Msg as String ' Declarar variável.
    Unload Form1 ' Descarregar o formulário.
    Msg = "Form1 foi descarregado. Escolha Yes para carregar e "
    Msg = Msg & "exibir o formulário. Escolha No para carregar o formulário "
    Msg = Msg & "e deixá-lo invisível."
    Answer = MsgBox(Msg, vbYesNo) ' Obter a resposta do usuário.
    If Answer = vbYes Then ' Avaliar a resposta.
        Show ' Se for Yes, exibir o formulário.
    Else
        Load Form1 ' Se for No, apenas carregue-o.
        Msg = "Form1 está agora carregado. Escolha O para exibí-lo."
        MsgBox Msg ' Exibir mensagem.
        Show ' Exibir o formulário.
    End If
End Sub

```

## Instrução Unload

Descarrega um formulário ou controle da memória.

### Sintaxe

**Unload** *object*

O espaço reservado *object* é o nome de um objeto **Form** ou elemento de matriz de controle a ser descarregado.

### Comentários

A descarga de um formulário ou controle pode ser necessária ou conveniente em alguns casos em que a memória usada é necessária para alguma outra coisa ou quando você precisa reconfigurar propriedades como seus valores originais.

Antes que um formulário seja descarregado, ocorrerá o procedimento de evento Query\_Unload, seguido do procedimento de evento Form\_Unload. Configurar o argumento *cancel* como **True** em qualquer um destes eventos impede que o formulário seja descarregado. Para objetos **MDIForm** o procedimento de evento Query\_Unload do objeto **MDIForm** ocorre seguido do procedimento de evento Query\_Unload e procedimento de evento Form\_Unload para cada formulário MDI filhote finalmente o procedimento de evento Form\_Unload do objeto **MDIForm**.

Quando um formulário é descarregado, todos os controles colocados no formulário durante o tempo de execução não são mais acessíveis. Os controles colocados no formulário durante o tempo de criação permanecem intactos; qualquer alteração durante o tempo de execução para aqueles controles e suas propriedades são perdidas quando o formulário é recarregado. Todas as alterações de propriedades de formulário também se perdem. O acesso a qualquer controle no formulário provoca sua recarga.

**Observação:** Quando um formulário é descarregado, somente o componente exibido é descarregado. O código associado ao módulo de formulário permanece na memória.

Somente elementos de matriz de controle adicionados a um formulário durante o tempo de execução podem ser descarregados com a instrução **Unload**. As propriedades de controles descarregados são reinicializadas quando os controles são recarregados.

## Exemplo da instrução Unload

Este exemplo utiliza a instrução **Unload** para descarregar um objeto **Form**. Para experimentar este exemplo, cole o código na seção Declarations de um objeto **Form** e, em seguida, execute o exemplo e clique no objeto **Form**.

```

Private Sub Form_Clic ()
    Dim Answer, Msg ' Declarar variável.
    Unload Form1 ' Descarregar o formulário.
    Msg = "Form1 foi descarregado. Escolha Yes para carregar e "
    Msg = Msg & "exibir o formulário. Escolha No para carregar o formulário "
    Msg = Msg & "e deixá-lo invisível."
    Answer = MsgBox(Msg, vbYesNo) ' Obter a resposta do usuário.
    If Answer = vbYes Then ' Avaliar a resposta.

```

---

```

        Show ' Se Yes, exibir o formulário.
Else
    Load Form1 ' Se No, apenas carregue-o.
    Msg = "Form1 está agora carregado. Escolha O para exibi-lo."
    MsgBox Msg ' Exibir a mensagem.
    Show ' Exibir o formulário.
End If
End Sub

```

## Função LoadPicture

Carrega um elemento gráfico na propriedade **Picture** de um formulário, controle **PictureBox** ou controle **Image**.

### Sintaxe

**LoadPicture**([*stringexpression*])

O argumento *stringexpression* é o nome de um arquivo de elementos gráficos a ser carregado.

### Comentários

Os formatos gráficos reconhecidos pelo Visual Basic incluem arquivos de bitmap (.bmp), arquivos de ícones (.ico), arquivos codificados run-length (.rle), metarquivos (.wmf), metarquivos aprimorados (.emf), arquivos GIF e arquivos JPEG (.jpg).

Os elementos gráficos são apagados de formulários, caixas de figura e controles de imagem atribuindo **LoadPicture** sem argumento.

Para carregar elementos gráficos para exibição em um controle **PictureBox**, controle **Image** ou como segundo plano de um formulário, o valor de retorno de **LoadPicture** deve ser atribuído à propriedade **Picture** do objeto onde a figura é exibida. Por exemplo:

```

Set Picture = LoadPicture("PARTY.BMP")
Set Picture1.Picture = LoadPicture("PARTY.BMP")

```

Para designar um ícone a um formulário, configure o valor de retorno da função **LoadPicture** como a propriedade **Icon** do objeto **Form**:

```

Set Form1.Icon = LoadPicture("MYICON.ICO")

```

Os ícones também podem ser atribuídos à propriedade **DragIcon** de todos os controles, exceto controles **Timer** e controles **Menu**. Por exemplo:

```

Set Command1.DragIcon = LoadPicture("MYICON.ICO")

```

Carregue um arquivo de elementos gráficos na **Área de transferência** usando **LoadPicture** como a seguir:

```

Clipboard.SetData LoadPicture("PARTY.BMP")

```

## Exemplo da função LoadPicture

Este exemplo usa a função **LoadPicture** para carregar uma figura na propriedade **Picture** de um formulário e apagar a figura do objeto **Form**. Para experimentar este exemplo, cole o código na seção **Declarations** de um objeto **Form** e, em seguida, execute o exemplo e clique no objeto **Form**.

```

Private Sub Form_Click ()
    Dim Msg As String ' Declarar variáveis.
    On Error Resume Next ' Configurar o tratamento de erro.
    Height = 3990
    Width = 4890 ' Configurar a altura e largura.
    Set Picture = LoadPicture("PAPER.BMP")
    ' Carregar o bitmap.
    If Err Then
        Msg = "Não foi possível encontrar o arquivo .BMP."
        MsgBox Msg ' Exibir a mensagem de erro.
        Exit Sub ' Sair se ocorrer erro.
    End If
    Msg = "Escolher O para limpar o bitmap do formulário."
    MsgBox Msg
    Set Picture = LoadPicture() ' Limpar o formulário.

```



## Instrução SavePicture

Salva um elemento gráfico da propriedade **Picture** ou **Image** de um objeto ou controle (se algum estiver associado a ele) em um arquivo.

### Sintaxe

**SavePicture** *picture*, *stringexpression*

A sintaxe da instrução **SavePicture** tem estas partes:

Parte	Descrição
<i>picture</i>	Controle <b>Picture</b> ou <b>Image</b> de onde o arquivo de elementos gráficos deve ser criado.
<i>Stringexpression</i>	Nome de arquivo do arquivo de elementos gráficos a ser salvo.

### Comentários

Se um elemento gráfico foi carregado de um arquivo para a propriedade **Picture** de um objeto durante o tempo de criação ou durante o tempo de execução e ele é um bitmap, ícone, metarquivo ou metarquivo aprimorado, ele é então salvo usando o mesmo formato que o arquivo original. Se ele for um arquivo GIF ou JPEG, ele é salvo como um arquivo de bitmap.

Elementos gráficos em uma propriedade **Image** são sempre salvos como arquivos de bitmap (.bmp), independente dos formatos originais.

## Exemplo da instrução SavePicture

Este exemplo usa a instrução **SavePicture** para salvar um elemento gráfico desenhado na propriedade **Picture** de um objeto **Form**. Para experimentar este exemplo, cole o código na seção Declarations de um objeto **Form** e, em seguida, execute o exemplo e clique no objeto **Form**.

```
Private Sub Form_Click ()
    ' Declarar variáveis.
    Dim CX, CY, Limit, Radius as Integer, Msg as String
    ScaleMode = vbPixels ' Configurar a escala como pixels.
    AutoRedraw = True ' Ativar o AutoRedraw.
    Width = Height ' Alterar largura para corresponder altura.
    CX = ScaleWidth / 2 ' Configurar a posição X.
    CY = ScaleHeight / 2 ' Configurar a posição Y.
    Limit = CX ' Limitar o tamanho de círculos.
    For Radius = 0 To Limit ' configurar o raio.
        Circle (CX, CY), Radius, RGB(Rnd * 255, Rnd * 255, Rnd * 255)
        DoEvents ' Passar a outro processamento.
    Next Radius
    Msg = "Escolha O para salvar os elementos gráficos deste formulário "
    Msg = Msg & "em um arquivo de bitmap."
    MsgBox Msg
    SavePicture Image, "TEST.BMP" ' Salvar a figura em um arquivo.
End Sub
```

## Método Circle

Desenha um círculo, elipse ou arco em um objeto.

### Sintaxe

*object.Circle [Step] (x, y), radius, [color, start, end, aspect]*

A sintaxe do método **Circle** tem os seguintes qualificadores de objeto e partes.

Parte	Descrição
<i>object</i>	Opcional. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To. Se <i>object</i> for omitido, pressupõe-se que o <b>Form</b> com o <u>foco</u> seja <i>object</i> .

<b>Step</b>	Opcional. <u>Palavra-chave</u> especificando que o centro do círculo, elipse ou arco é relativo às coordenadas atuais fornecidas pelas propriedades <b>CurrentX</b> e <b>CurrentY</b> de <i>object</i> .
<b>(x, y)</b>	Obrigatório. Valores <b>Single</b> indicando as coordenadas do ponto central do círculo, elipse ou arco. A propriedade <b>ScaleMode</b> de <i>object</i> determina a unidade de medida a ser usada.
<b>radius</b>	Obrigatório. Valor <b>Single</b> indicando o raio do círculo, elipse ou arco. A propriedade <b>ScaleMode</b> de <i>object</i> determina a unidade de medida usada.
<b>color</b>	Opcional. Valor inteiro <b>Long</b> indicando a cor RGB do contorno do círculo. Se omitido, o valor da propriedade <b>ForeColor</b> é usado. Você pode usar a função <b>RGB</b> ou a função <b>QBColor</b> para especificar a cor.
<b>start, end</b>	Opcional. Valores de precisão simples. Quando um arco, círculo parcial ou elipse é desenhado, <i>start</i> e <i>end</i> especificam (em radianos) as posições de início e fim do arco. O intervalo para ambos é -2 pi radianos a 2 pi radianos. O valor padrão para <i>start</i> é 0 radianos; o padrão para <i>end</i> é 2 * pi radianos.
<b>aspect</b>	Opcional. Valor de precisão simples indicando a relação de aspecto do círculo. O valor padrão é 1.0, que produz um círculo perfeito (não-elíptico) em qualquer tela.

### Comentários

Para preencher um círculo, configure as propriedades **FillColor** e **FillStyle** do objeto no qual é desenhado o círculo ou elipse. Somente figuras fechadas podem ser preenchidas. Figuras fechadas incluem círculos, elipses ou pedaços de pizza (arcos com linhas de raio em ambas as extremidades).

Ao desenhar um círculo parcial ou elipse, se *start* for negativo, **Circle** desenha um raio até *start*, e trata o ângulo como positivo; se *end* for negativo, **Circle** desenha um raio até *end* e trata o ângulo como positivo. O método **Circle** sempre desenha em sentido anti-horário (positivo).

A espessura da linha usada para desenhar o círculo, elipse ou arco depende da configuração da propriedade **DrawWidth**. A maneira como o círculo é desenhado em segundo plano depende da configuração das propriedades **DrawMode** e **DrawStyle**.

Ao desenhar pedaços de pizza, para traçar um raio com ângulo 0 (produzindo um segmento de linha horizontal à direita) especifique um número negativo muito pequeno para *start*, ao invés de zero.

Você pode omitir um argumento no meio da sintaxe, mas precisa incluir a vírgula do argumento antes de incluir o próximo argumento. Se omitir um argumento opcional, omita a vírgula que acompanha o último argumento especificado.

Quando **Circle** é executado, as propriedades **CurrentX** e **CurrentY** são configuradas como ponto central especificado pelos argumentos.

Este método não pode ser usado em um bloco **With...End With**.

## Exemplo do método Circle

Este exemplo usa o método **Circle** para desenhar alguns círculos concêntricos no centro de um formulário. Para experimentar este exemplo, cole o código na seção General de um formulário. Em seguida, pressione F5 e clique no formulário.

```
Sub Form_Click ()
    Dim CX, CY, Radius, Limit ' Declarar variável.
    ScaleMode = 3 ' Configurar a escala como pixels.
    CX = ScaleWidth / 2 ' Configurar a posição X.
    CY = ScaleHeight / 2 ' Configurar a posição Y.
    If CX > CY Then Limit = CY Else Limit = CX
    For Radius = 0 To Limit ' configurar o raio.
        Circle (CX, CY), Radius, RGB(Rnd * 255, Rnd * 255, Rnd * 255)
    Next Radius
End Sub
```

## Método Line

Desenha linhas e retângulos em um objeto.

### Sintaxe

*object*.Line [**Step**] (*x1*, 1) [**Step**] (*x2*, *y2*), [*color*], [**B**][**F**]

A sintaxe do método **Line** tem os seguintes qualificadores de objeto e partes:

Parte	Descrição
<i>object</i>	Opcional. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To. Se o objeto for omitido, presume-se que o <b>Form</b> com o <u>foco</u> seja <i>object</i> .
<b>Step</b>	Opcional. <u>Palavra-chave</u> especificando que as coordenadas de ponto de partida são relativas à posição gráfica atual fornecida pelas propriedades <b>CurrentX</b> e <b>CurrentY</b> .
( <i>x1</i> , <i>y1</i> )	Opcional. Valores <b>Single</b> indicando as coordenadas do ponto inicial da linha ou retângulo. A propriedade <b>ScaleMode</b> determina a unidade de medida usada. Se omitida, a linha se inicia na posição indicada por <b>CurrentX</b> e <b>CurrentY</b> .
<b>Step</b>	Opcional. Palavra-chave especificando que as coordenadas de ponto final são relativas ao ponto inicial da linha.
( <i>x2</i> , <i>y2</i> )	Obrigatória. Valores <b>Single</b> indicando as coordenadas do ponto final para a linha que está sendo desenhada.
<i>color</i>	Opcional. Valor inteiro <b>Long</b> indicando o valor RGB usado para desenhar a linha. Se for omitida, será usada a configuração da propriedade <b>ForeColor</b> . Você pode usar a função <b>RGB</b> ou a função <b>QBColor</b> para especificar a cor.
<b>B</b>	Opcional. Se incluído faz com que a caixa seja desenhada usando as coordenadas para especificar cantos opostos da caixa.
<b>F</b>	Opcional. Se for usada a opção <b>B</b> , a opção <b>F</b> especifica que a caixa é preenchida com a mesma cor usada para desenhar a caixa. Você não pode usar <b>F</b> sem <b>B</b> . Se <b>B</b> for usado sem <b>F</b> , a caixa é preenchida com a <b>FillColor</b> e <b>FillStyle</b> atuais. O valor padrão para <b>FillStyle</b> é transparente.

### Comentários

Para desenhar linhas interligadas, comece uma linha subsequente no ponto final da linha anterior. A espessura da linha desenhada depende da configuração da propriedade **DrawWidth**. A maneira como uma linha ou caixa é desenhada em segundo plano depende da configuração das propriedades **DrawMode** e **DrawStyle**.

Quando **Line** é executado, as propriedades **CurrentX** e **CurrentY** são configuradas como o ponto final especificado pelos argumentos.

Este método não pode ser usado em um bloco **With...End With**.

## Exemplo do método Line

Este exemplo usa o método **Line** para desenhar caixas concêntricas em um formulário. Para experimentar este exemplo, cole o código na seção General de um formulário. Em seguida, pressione F5 e clique no formulário.

```
Sub Form_Click ()
    Dim CX, CY, F, F1, F2, I ' Declarar variáveis
    ScaleMode = 3 ' Configurar ScaleMode como pixels.
    CX = ScaleWidth / 2 ' Obter o centro horizontal.
    CY = ScaleHeight / 2 ' Obter o centro vertical.
    DrawWidth = 8 ' Configurar DrawWidth.
```

```

For I = 50 To 0 Step -2
    F = I / 50 ' Executar cálculos
    F1 = 1 - F: F2 = 1 + F ' provisórios.
    Forecolor = QBColor(I Mod 15) ' Definir a cor de primeiro
plano.
    Line (CX * F1, CY * F1)-(CX * F2, CY * F2), , BF
Next I
DoEvents ' Passar a outro processamento.
If CY > CX Then ' Configurar DrawWidth.
    DrawWidth = ScaleWidth / 25
Else
    DrawWidth = ScaleHeight / 25
End If
For I = 0 To 50 Step 2 ' Montar loop.
    F = I / 50 ' Executar cálculos
    F1 = 1 - F: F2 = 1 + F ' provisórios.
    Line (CX * F1, CY)-(CX, CY * F1) ' Desenhar superior esquerda.
    Line -(CX * F2, CY) ' Desenhar superior direita.
    Line -(CX, CY * F2) ' Desenhar inferior direita.
    Line -(CX * F1, CY) ' Desenhar inferior esquerda.
    Forecolor = QBColor(I Mod 15) ' Alterar a cor a cada vez.
Next I
DoEvents ' Passar a outro processamento.
End Sub

```

## Método PSet

Configura um ponto em um objeto como uma cor especificada.

### Sintaxe

*object.PSet [Step] (x, y), [color]*

A sintaxe do método **PSet** tem os seguintes qualificadores de objeto e partes:

Parte	Descrição
<i>object</i>	Opcional. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To. Se <i>object</i> for omitido, pressupõe-se que o <b>Form</b> com o <u>foco</u> seja <i>object</i> .
<b>Step</b>	Opcional. <u>Palavra-chave</u> especificando que as coordenadas são relativas à posição atual dos elementos gráficos dadas pelas propriedades <b>CurrentX</b> e <b>CurrentY</b> .
(x, y)	Obrigatório. Valores <b>Single</b> indicando as coordenadas horizontal (eixo de x) e vertical (eixo de y) do ponto a ser configurado.
<i>color</i>	Opcional. Valor inteiro <b>Long</b> indicando a cor RGB especificada para o ponto. Se omitido será usada a configuração atual da propriedade <b>ForeColor</b> . Você pode usar a função <b>RGB</b> ou a função <b>QBColor</b> para especificar a cor.

### Comentários

O tamanho do ponto desenhado depende da configuração da propriedade **DrawWidth**. Quando **DrawWidth** é 1, **PSet** configura a cor especificada para um único pixel. Quando **DrawWidth** é maior que 1, o ponto é centrado nas coordenadas especificadas.

A maneira como o ponto é desenhado depende da configuração das propriedades **DrawMode** e **DrawStyle**.

Quando **PSet** é executado, as propriedades **CurrentX** e **CurrentY** são configuradas como o ponto especificado pelos argumentos.

Para apagar um único pixel com o método **PSet**, especifique as coordenadas do pixel e use a configuração da propriedade **BackColor** como argumento *color*.

Este método não pode ser usado em um bloco **With...End With**.

## Exemplo do método PSet

Este exemplo usa o método **PSet** para desenhar confete em um formulário. Para experimentar este exemplo, cole o código na seção General de um formulário. Em seguida, pressione F5 e clique no formulário.

```
Sub Form_Click ()
    Dim CX, CY, Msg, XPos, YPos      ' Declarar variáveis.
    ScaleMode = 3                    ' Configurar ScaleMode como
        ' pixels.
    DrawWidth = 5                     ' Configurar DrawWidth.
    ForeColor = QBColor(4)           ' Configurar o primeiro plano como vermelho.
    FontSize = 24                     ' Configurar o tamanho do ponto.
    CX = ScaleWidth / 2               ' Obter o centro horizontal.
    CY = ScaleHeight / 2              ' Obter o centro vertical.
    Cls                               ' Limpar o formulário.
    Msg = "Feliz Ano Novo!"
    CurrentX = CX - TextWidth(Msg) / 2 ' Posição Horizontal.
    CurrentY = CY - TextHeight(Msg)    ' Posição vertical.
    Print Msg                          ' Imprimir mensagem.
    Do
        XPos = Rnd * ScaleWidth        ' Obter posição horizontal.
        YPos = Rnd * ScaleHeight        ' Obter posição vertical.
        PSet (XPos, YPos), QBColor(Rnd * 15) ' Desenhar confete.
        DoEvents ' Passar a outro
    Loop ' processamento.
End Sub
```

## Método AddItem

Adiciona um item a um controle **ListBox** ou **ComboBox**, ou adiciona uma linha a um controle **MS Flex Grid**. Não suporta argumentos nomeados.

### Sintaxe

*object.AddItem item, index*

A sintaxe do método **AddItem** tem estas partes:

Parte	Descrição
<i>object</i>	Obrigatório. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>item</i>	Obrigatório. <u>Expressão de sequência de caracteres</u> especificando o item a ser adicionado ao objeto. Somente para o controle <b>MS Flex Grid</b> , use o caractere de tabulação (código de caractere 90) para separar seqüências de caracteres múltiplas que você deseja inserir em cada coluna de uma linha recém-adicionada.
<i>index</i>	Opcional. Número inteiro especificando a posição dentro do objeto onde o novo item ou linha está colocado. Para o primeiro item em um controle <b>ListBox</b> ou <b>ComboBox</b> ou para a primeira linha em um controle <b>MS Flex Grid</b> , <i>index</i> é 0.

### Comentários

Se você fornece um valor válido para *index*, *item* é colocado naquela posição dentro de *object*. Se *index* é omitido, *item* é adicionado na posição classificada adequada (se a propriedade **Sorted** estiver configurada como **True**) ou ao final da lista (se **Sorted** for configurado como **False**). Um controle **ListBox** ou **ComboBox** que seja acoplado a um controle **Data** não suporta o método **AddItem**.

## Método Arrange

Organiza as janelas ou ícones em um objeto **MDIForm**. Não suporta argumentos nomeados.

### Sintaxe

*object.Arrange arrangement*

A sintaxe do método **Arrange** tem estas partes:

Parte	Descrição
<i>object</i>	Obrigatório. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>arrangement</i>	Obrigatório. Um valor ou constante que especifica como organizar janelas ou ícones em um objeto <b>MDIForm</b> , conforme descrito em Configurações.

### Configurações

As configurações para *arrangement* são:

Constante	Valor	Descrição
<b>vbCascade</b>	0	Coloca em cascata todos os formulários <u>MDI filho</u> não minimizados
<b>vbTileHorizontal</b>	1	Coloca lado a lado horizontalmente todos os MDI filho não minimizados
<b>vbTileVertical</b>	2	Coloca lado a lado verticalmente todos os formulários MDI filho não minimizados
<b>vbArrangeIcons</b>	3	Organiza ícones para formulários MDI filho minimizados

### Comentários

Estas constantes estão listadas na biblioteca de objetos Visual Basic (VB) no Object Browser. Janelas ou ícones são organizados, mesmo se o objeto **MDIForm** está minimizado. Os resultados são visíveis quando **MDIForm** é maximizado.

## Método Clear (Clipboard, ComboBox, ListBox)

Limpa o conteúdo de uma **ListBox**, **ComboBox** ou a Área de transferência do sistema.

### Sintaxe

*object*.Clear

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Um controle **ListBox** ou **ComboBox** acoplado a um controle **Data** não suporta o método **Clear**.

## Método Cls

Limpa elementos gráficos e texto gerado durante o tempo de execução de um **Form** ou **PictureBox**.

### Sintaxe

*object*.Cls

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To. Se *object* é omitido, pressupõe-se que o **Form** com o foco é *object*.

### Comentários

**Cls** limpa texto e elementos gráficos gerados durante o tempo de execução por instruções gráficas ou de impressão. Bitmaps de segundo plano configurados usando a propriedade **Picture** e os controles colocados em um **Form** durante o tempo de criação não são afetados por **Cls**. Elementos gráficos e texto colocado em um **Form** ou **PictureBox** enquanto a propriedade **AutoRedraw** é configurada como **True** não são afetados se **AutoRedraw** for configurado como **False** antes que **Cls** seja acionado. Isto é, você pode manter texto e elementos gráficos em um **Form** ou **PictureBox** manipulando a propriedade **AutoRedraw** do objeto no qual você está trabalhando.

Após **Cls** ter sido acionado, as propriedades **CurrentX** e **CurrentY** do objeto são reconfiguradas como 0.

---

## Método Drag

Inicia, encerra ou cancela uma operação de arraste de qualquer controle, exceto os controles **Line**, **Menu**, **Shape**, **Timer** ou **CommonDialog**. Não suporta argumentos nomeados.

### Sintaxe

*object.Drag action*

A sintaxe do método **Drag** tem estas partes:

Parte	Descrição
<i>object</i>	Obrigatório. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To. Se <i>object</i> for omitido, o objeto cujo procedimento de evento contenha o método <b>Drag</b> será assumido.
<i>action</i>	Opcional. Uma constante ou valor que especifica a ação a ser executada, conforme descrito em Configurações. Se <i>action</i> for omitido, o padrão é iniciar o arraste do objeto.

### Configurações

As configurações para *action* são:

Constante	Valor	Descrição
<b>vbCancel</b>	0	Cancela a operação de arraste
<b>vbBeginDrag</b>	1	Inicia o arraste de <i>object</i>
<b>vbEndDrag</b>	2	Encerra o arrastar-e-soltar o <i>object</i>

### Comentários

Estas constantes estão listadas na biblioteca de objetos Visual Basic (VB) no Object Browser.

Somente se exige o uso do método **Drag** para controlar uma operação de arrastar-e-soltar quando a propriedade **DragMode** do objeto é configurada como Manual (0). Entretanto, você pode usar **Drag** em um objeto cuja propriedade **DragMode** está configurada como Automatic (1 ou **vbAutomatic**).

Se você deseja que o ponteiro do mouse altere sua forma, enquanto o objeto está sendo arrastado, use qualquer uma das propriedades **DragIcon** ou **MousePointer**. A propriedade **MousePointer** somente é usada se nenhum **DragIcon** for especificado.

Em versões anteriores do Visual Basic, **Drag** era um método assíncrono onde as instruções subsequentes eram acionadas, apesar da ação Drag não ter sido encerrada.

## Método EndDoc

Encerra uma operação de impressão enviada ao objeto **Printer**, liberando o documento para o spooler ou dispositivo de impressão.

### Sintaxe

*object.EndDoc*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Se **EndDoc** for imediatamente acionada após o método **NewPage**, nenhuma página em branco adicional é impressa.

## Função GetAutoServerSettings

Retorna informações sobre o estado do registro de um componente ActiveX.

### Sintaxe

*object.GetAutoServerSettings([progid], [clsid])*

A sintaxe da função **GetAutoServerSettings** tem estas partes:

Parte	Descrição
<i>object</i>	Obrigatório. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.

<i>progid</i>	Opcional. Uma <u>expressão de variante</u> especificando a ProgID do componente.
<i>clsid</i>	Opcional. Uma <u>expressão de variante</u> especificando a CLSID do componente.

### Valores de retorno

A função **GetAutoServerSettings** retorna uma Variant que contém uma matriz de valores sobre o componente ActiveX em particular. Os valores de índice e descrições são:

Valor	Descrição
1	Verdadeiro se o componente ActiveX for remotamente registrado.
2	Nome da máquina remota.
3	Nome do protocolo RPC.
4	Nível de autenticação RPC.

### Comentários

Se um valor estiver faltando ou não estiver disponível, ele será uma sequência de caracteres vazia. Se ocorrer um erro durante o método, o valor de retorno será uma Variant do tipo Empty.

## Método GetData

Retorna um elemento gráfico do objeto **Clipboard**. Não suporta argumentos nomeados.

### Sintaxe

*object*.**GetData** (*format*)

A sintaxe do método **GetData** tem estas partes:

Parte	Descrição
<i>object</i>	Obrigatório. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>format</i>	Opcional. Uma constante ou valor que especifica o formato de elementos gráficos do <b>Clipboard</b> , conforme descrito em Configurações. A constante ou valor deve estar entre parênteses. Se <i>format</i> for 0 ou omitido, <b>GetData</b> usa automaticamente o formato adequado.

### Configurações

As Configurações de *format* são:

Constante	Valor	Descrição
<b>vbCFBitmap</b>	2	<u>Bitmap</u> (arquivos .bmp)
<b>vbCFMetafile</b>	3	<u>Metarquivo</u> (arquivos .wmf)
<b>vbCFDIB</b>	8	Bitmap independente de dispositivo (DIB)
<b>vbCFPalette</b>	9	Paleta de cores

### Comentários

Estas constantes estão listadas na biblioteca de objetos Visual Basic (VB) no Object Browser. Se nenhum elemento gráfico no objeto **Clipboard** corresponde ao formato esperado, nada é retornado. Se somente uma paleta de cores estiver presente no objeto **Clipboard**, um DIB de tamanho mínimo (1 x 1) será criado.

## Método GetFormat

Retorna um número inteiro indicando se um item no objeto **Clipboard** corresponde a um formato especificado. Não suporta argumentos nomeados.

### Sintaxe

*object*.**GetFormat** (*format*)

A sintaxe do método **GetFormat** tem estas partes:

Parte	Descrição
-------	-----------



<i>object</i>	Obrigatório. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>format</i>	Obrigatório. Um valor ou objeto que especifica o formato do objeto <b>Clipboard</b> conforme descrito em Configurações. O valor ou constante deve estar entre parênteses.

### Configurações

As configurações para *format* são:

Constante	Valor	Descrição
<b>vbCFLin</b>	&HBF00	Informações de conversação DDE
<b>vbCFText</b>	1	Texto
<b>vbCFBitmap</b>	2	<u>Bitmap</u> (arquivos .bmp)
<b>vbCFMetafile</b>	3	<u>Metarquivo</u> (arquivos .wmf)
<b>vbCFDIB</b>	8	Bitmap independente de dispositivo (DIB)
<b>vbCFPalette</b>	9	Paleta de cores

### Comentários

Estas constantes estão listadas na biblioteca de objetos Visual Basic (VB) no Object Browser. O método **GetFormat** retorna **True** se um item no objeto **Clipboard** corresponde ao formato especificado. Caso contrário, ele retorna **False**. Para os formatos **vbCFDIB** e **vbCFBitmap**, não importa a paleta de cores usada no **Clipboard** quando o elemento gráfico é exibido.

## Método GetText

Retorna uma seqüência de caracteres de texto a partir do objeto **Clipboard**. Não suporta argumentos nomeados.

### Sintaxe

*object*.**GetText** (*format*)

A sintaxe do método **GetText** tem estas partes:

Parte	Descrição
<i>object</i>	Obrigatório. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>format</i>	Opcional. Um valor ou constante que especifica o formato do objeto <b>Clipboard</b> , conforme descrito em Configurações. O valor ou constante deve estar entre parênteses.

### Configurações

As configurações para *format* são:

Constante	Valor	Descrição
<b>vbCFLin</b>	&HBF00	Informações de conversação DDE
<b>vbCFText</b>	1	(Padrão) texto
<b>vbCFRTF</b>	&HBF01	Rich Text Format (arquivo .rtf)

### Comentários

Estas constantes estão listadas na biblioteca de objetos Visual Basic (VB) no Object Browser. Se nenhuma seqüência de caracteres de texto no objeto **Clipboard** corresponde ao formato especificado, será retornada uma seqüência de caracteres de comprimento zero ("").

## Método Hide

Oculta um objeto **MDIForm** ou **Form** mas não o descarrega.

### Sintaxe

*object*.**Hide**

---

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To. Se *object* for omitido, pressupõe-se que o formulário com o foco seja *object*.

### Comentários

Quando um formulário está oculto, ele é removido da tela e sua propriedade **Visible** é configurada como **False**. Os controles de um formulário oculto não estão acessíveis ao usuário, mas eles estão disponíveis para o aplicativo do Visual Basic em execução, para outros processos que possam estar se comunicando com o aplicativo através de DDE e para eventos do controle **Timer**.

Quando um formulário está oculto, o usuário pode interagir com o aplicativo até que todo o código no procedimento de evento que provocou a ocultação do formulário tenha encerrado sua execução.

Se o formulário não está carregado quando o método **Hide** é acionado, este método carrega o formulário, mas não o exibe.

## Método illDoc

Encerra imediatamente o trabalho de impressão atual.

### Sintaxe

*object.illDoc*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Se o gerenciador de impressão do sistema operacional está manipulando um trabalho de impressão (o gerenciador de impressão está sendo executado e tem impressão em segundo plano ativada), **illDoc** exclui o trabalho de impressão atual, e a impressora nada recebe.

Se o gerenciador de impressão não estiver manipulando o trabalho de impressão (impressão em segundo plano não está ativada) alguns, ou todos os dados, podem ser enviados à impressora antes que **illDoc** possa estar efetivo. Neste caso, o driver de impressora reconfigura a impressora assim que possível, e encerra o trabalho de impressão.

## Método LinExecute

Envia uma sequência de caracteres de comando ao aplicativo de origem em uma conversa DDE. Não suporta argumentos nomeados.

### Sintaxe

*object.LinExecute string*

A sintaxe do método **LinExecute** tem estas partes:

Parte	Descrição
<i>object</i>	Obrigatório. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>string</i>	Obrigatório. <u>Expressão de sequência de caracteres</u> contendo um comando reconhecido pelo aplicativo de origem.

### Comentários

O valor efetivo de *string* varia dependendo do aplicativo de origem. Por exemplo, o Microsoft Excel e Microsoft Word for Windows aceitam sequências de caracteres de comando que consistem em seus comandos de macro colocados entre colchetes ([ ]). Para visualizar sequências de caracteres de comando aceitas por um aplicativo de origem, consulte a documentação do aplicativo.

## Método LinPoe

Transfere o conteúdo de um controle **Label**, **PictureBox** ou **TextBox** para o aplicativo de origem em uma conversa DDE.

### Sintaxe

*object.LinPoe*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na

---

lista Applies To.

### Comentários

O *object* é o nome de um **Label**, **PictureBox** ou **TextBox** envolvido em uma conversação DDE como um destino. Se *object* for um **Label**, **LinPoe** transfere o conteúdo da propriedade **Caption** para a origem. Se *object* for uma **PictureBox**, **LinPoe** transfere o conteúdo da propriedade **Picture** para a origem. Se *object* for uma **TextBox**, **LinPoe** transfere o conteúdo da propriedade **Text** para a origem.

Tipicamente, as informações em uma conversação DDE fluem da origem para o destino. Entretanto, **LinPoe** permite que um objeto de destino forneça dados à origem. Nem todos os aplicativos de origem aceitam informações fornecidas desta forma; se o aplicativo de origem não aceita os dados, ocorre um erro.

## Método LinRequest

Pede ao aplicativo de origem em uma conversação DDE que atualize o conteúdo de um controle **Label**, **PictureBox** ou **TextBox**.

### Sintaxe

*object*.**LinRequest**

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

O *object* é o nome de um **Label**, **PictureBox** ou **TextBox** envolvido em uma conversação DDE como um destino. **LinRequest** faz com que o aplicativo de origem envie os dados mais atuais a *object*, atualizando a configuração da propriedade **Caption** se *object* for um **Label**, a configuração da propriedade **Picture** se *object* for uma **PictureBox**, ou a configuração da propriedade **Text** se *object* for uma **TextBox**.

Se a propriedade **LinMode** de *object* for configurada como Automatic (1 ou **vbLinAutomatic**), o aplicativo de origem atualiza automaticamente *object* e **LinRequest** não é necessário. Se a propriedade **LinMode** de *object* for configurada como Manual (2 ou **vbLinManual**), o aplicativo de origem somente atualiza *object* quando **LinRequest** for usado. Se a propriedade **LinMode** de *object* for configurada como Notify (3 ou **vbLinNotify**), a origem notifica o destino que os dados se alteraram acionando o evento LinNotify. O destino deve, então, usar **LinRequest** para atualizar os dados.

## Método LinSend

Transfere o conteúdo de um controle **PictureBox** para o aplicativo de destino em uma conversação DDE.

### Sintaxe

*object*.**LinSend**

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

O *object* deve ser um objeto **PictureBox** em um objeto **Form** que seja uma origem em uma conversação DDE.

Quando outros aplicativos estabelecem vínculos automáticos com um **Form** em seu aplicativo, o Visual Basic os notifica quando o conteúdo de uma **TextBox** ou um **Label** no **Form** se altera. Entretanto, o Visual Basic não notifica automaticamente um aplicativo de destino DDE quando a configuração da propriedade **Picture** de uma **PictureBox** em um **Form** de origem se altera. Como a quantidade de dados em um elemento gráfico pode ser muito grande, e porque raramente faz sentido atualizar um aplicativo de destino, visto que cada pixel na figura se altera, o Visual Basic exige que você use o método **LinSend** para notificar explicitamente os aplicativos DDE de destino quando o conteúdo de uma **PictureBox** se altera.

## Função LoadResData

Carrega dados de diversos tipos possíveis de um arquivo de recursos (.res) e retorna uma matriz

Byte.

### Sintaxe

**LoadResData**(*index*, *format*)

A sintaxe da função **LoadResData** tem estas partes:

Parte	Descrição
<i>index</i>	Obrigatório. Número inteiro ou sequência de caracteres especificando o identificador (ID) dos dados contidos no arquivo de recursos. O recurso cuja ID é 1 é reservado para o ícone do aplicativo.
<i>format</i>	Obrigatório. O valor que especifica o formato original dos dados que estão sendo retornados, conforme descrito em Configurações. O valor também pode ser o nome da sequência de caracteres de um recurso definido pelo usuário.

### Configurações

As configurações para *format* são:

Configuração	Descrição
1	Recurso de cursor
2	Recurso de <u>bitmap</u>
3	Recurso de <u>Ícone</u>
4	Recurso de menu
5	<u>Caixa de diálogo</u>
6	Recurso de sequência de caracteres
7	Recurso de diretório de fonte
8	Recurso de fonte
9	Tabela de aceleradores
10	Recurso definido pelo usuário
12	Cursor de grupo
14	Ícone de grupo

### Comentários

Os dados que **LoadResData** carrega do arquivo de recursos pode ter até 64.

Usar **LoadResData** com um tipo de recurso bitmap, ícone ou cursor retorna uma sequência de caracteres contendo os bits efetivos no recurso. Se você deseja usar o verdadeiro bitmap, ícone ou recurso, use a função **LoadResPicture**.

Usar **LoadResData** é útil para localizar um aplicativo do Visual Basic porque os recursos que precisam ser traduzidos estão isolados em um arquivo de recursos e não há necessidade de acessar o código fonte ou recompilar o aplicativo.

## Função LoadResPicture

Carrega um bitmap, ícone, ou cursor de um arquivo de recursos (.res).

### Sintaxe

**LoadResPicture**(*index*, *format*)

A sintaxe da função **LoadResPicture** tem estas partes:

Parte	Descrição
<i>index</i>	Obrigatório. Número inteiro ou sequência de caracteres especificando o identificador (ID) dos dados no arquivo de recursos. O recurso cuja ID é 1 está reservado para o ícone do aplicativo.
<i>format</i>	Obrigatório. Valor ou constante que especifica o formato dos dados que estão sendo retornados, conforme descrito em Configurações.

### Configurações

As configurações para *format* são:

Constante	Valor	Descrição
<b>VbResBitmap</b>	0	Recurso de bitmap

<b>vbResIcon</b>	1	Recurso de ícone
<b>vbResCursor</b>	2	Recurso de cursor

### Comentários

Você pode usar a função **LoadResPicture** ao invés de referir-se a elementos gráficos armazenados na propriedade **Picture** de um **Form** ou controles.

Armazenar bitmaps, ícones ou cursores e acessá-los individualmente na medida das necessidades no arquivo de recurso, ao invés de todos de uma vez quando um **Form** é carregado.

Usar **LoadResPicture** é útil para localizar um aplicativo do Visual Basic, pois os recursos que se precisa traduzir estão isolados em um arquivo de recursos e não há necessidade de se acessar código ou recompilar o aplicativo.

## Função LoadResString

Carrega uma seqüência de caracteres a partir de um arquivo de recursos (.res).

### Sintaxe

**LoadResString**(*index*)

A sintaxe da função **LoadResString** tem estas partes:

<b>Parte</b>	<b>Descrição</b>
<i>index</i>	Obrigatório. Número inteiro especificando o identificador (ID) dos dados contidos no arquivo de recursos. O recurso cuja ID é 1 é reservado para o ícone do aplicativo.

### Comentários

Você pode usar a função **LoadResString** ao invés de literais de seqüência de caracteres em seu código. Armazenar longas seqüências de caracteres de dados, e acessá-las individualmente no arquivo de recursos à medida que forem necessárias, melhora o tempo de carga, pois você pode carregá-las individualmente à medida que forem necessários a partir do arquivo de recursos, ao invés de todos de uma só vez quando o formulário é carregado.

Usar **LoadResString** é útil para localizar um aplicativo do Visual Basic, pois os recursos que precisam ser traduzidos estão isolados em um arquivo de recursos e não há necessidade de acessar código fonte ou recompilar o aplicativo.

## Método Move

Move um **MDIForm**, **Form** ou controle. Não suporta argumentos nomeados.

### Sintaxe

*object*.**Move** *left, top, width, height*

A sintaxe do método **Move** tem estas partes:

<b>Parte</b>	<b>Descrição</b>
<i>object</i>	Opcional. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To. Se <i>object</i> for omitido pressupõe-se que o formulário com o <u>foco</u> seja <i>object</i> .
<i>Left</i>	Obrigatório. Valor de precisão simples indicando a coordenada horizontal da borda esquerda de <i>object</i> .
<i>Top</i>	Opcional. Valor de precisão simples indicando a coordenada vertical (eixo de y) para a borda superior de <i>object</i> .
<i>width</i>	Opcional. Valor de precisão simples indicando a nova largura de <i>object</i> .
<i>height</i>	Opcional. Valor de precisão simples indicando a nova altura de <i>object</i> .

### Comentários

Somente o argumento *left* é obrigatório. Entretanto, para especificar quaisquer outros argumentos, você deve especificar todos os argumentos que aparecem na sintaxe antes do argumento

que você deseja especificar. Por exemplo, você não pode especificar *width* sem especificar *left* e *top*. Qualquer argumento antes dele que não seja especificado permanece inalterado. Para formulários e controles em um controle **Frame**, o sistema de coordenadas é sempre em *twips*. Mover um formulário na tela ou mover um controle em um **Frame** é sempre relativo à origem (0,0) que é o canto superior esquerdo. Ao mover um controle em um objeto **Form** ou em uma **PictureBox** (ou um formulário **MDI** filho em um objeto **MDIForm**), o sistema de coordenadas do objeto recipiente é usado. O sistema de coordenadas ou unidade de medida é configurado com a propriedade **ScaleMode** durante o tempo de criação. Você pode alterar o sistema de coordenadas durante o tempo de execução com o método **Scale**.

## Método NewPage

Fecha a página atual e avança até a página seguinte no objeto **Printer**.

### Sintaxe

*object.NewPage*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

**NewPage** avança uma nova página impressa e reconfigura a posição de impressão como o canto superior esquerdo da nova página. Ao ser acionado, **NewPage** incrementa em 1 a propriedade **Page** do objeto **Printer**.

## Método PaintPicture

Desenha o conteúdo de um arquivo de elementos gráficos (.bmp, .wmf, .emf, .ico ou .dib) em um **Form**, **PictureBox** ou **Printer**. Não suporta argumentos nomeados.

### Sintaxe

*object.PaintPicture picture, x1, y1, width1, height1, x2, y2, width2, height2, opcode*

A sintaxe do método **PaintPicture** tem estas partes:

Parte	Descrição
<i>object</i>	Opcional. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To. Se <i>object</i> for omitido, pressupõe-se que o objeto <b>Form</b> com o <u>foco</u> seja <i>object</i> .
<i>Picture</i>	Obrigatório. A origem do elemento gráfico a ser desenhado em <i>object</i> . Deve ser a propriedade <b>Picture</b> de um <b>Form</b> ou <b>PictureBox</b> .
<i>x1, y1</i>	Obrigatório. Valores de precisão simples indicando as coordenadas de destino (eixo de x e eixo de y) em <i>object</i> para a <i>picture</i> a ser desenhada. A propriedade <b>ScaleMode</b> de <i>object</i> determina a unidade de medida usada.
<i>Width1</i>	Opcional. Valor de precisão simples indicando a largura de destino de <i>picture</i> . A propriedade <b>ScaleMode</b> de <i>object</i> determina a unidade de medida usada. Se a largura de destino for maior ou menor que a largura de origem ( <i>width2</i> ), <i>picture</i> é esticada ou comprimida para ajustar-se. Se omitida, será usada a largura de origem.
<i>Height1</i>	Opcional. Valor de precisão simples indicando a altura de destino de <i>picture</i> . A propriedade <b>ScaleMode</b> de <i>object</i> determina a unidade de medida usada. Se a altura de destino for maior ou menor que a altura de origem ( <i>height2</i> ), <i>picture</i> é esticada ou comprimida para ajustar-se. Se for omitida, será utilizada a altura de origem.
<i>x2, y2</i>	Opcional. Valores de precisão simples indicando as coordenadas (eixo de x e eixo de y) de uma área de recorte dentro de <i>picture</i> . A propriedade <b>ScaleMode</b> de <i>object</i> determina a unidade de medida usada. Se for omitida, pressupõe-se que seja 0.
<i>Width2</i>	Opcional. Valor de precisão simples indicando a largura de origem de uma área de recorte em <i>picture</i> . A proprie-

---

	dade <b>ScaleMode</b> de <i>object</i> determina a unidade de medida usada. Se for omitida, a largura total de origem será usada.
<i>Height2</i>	Opcional. Valor de precisão simples indicando a altura de origem de uma área de recorte em <i>picture</i> . A propriedade <b>ScaleMode</b> de <i>object</i> determina a unidade de medida usada. Se for omitida, será usada a altura de origem completa.
<i>Opcode</i>	Opcional. Valor ou código Long que somente é usado com <i>bitmaps</i> . Ele define uma operação voltada para bits (por exemplo, <b>vbMergeCopy</b> ou <b>vbSrcAnd</b> ) que é executada sobre <i>picture</i> ao ser desenhada em <i>object</i> . Para uma lista completa de constantes de operador voltada para bits, consulte o tópico RasterOp Constants no Visual Basic Help.

### Comentários

Você pode girar horizontalmente ou verticalmente um bitmap usando valores negativos para a altura de destino (*height1*) e/ou a largura de destino (*width1*).

Você pode omitir tantos argumentos de preenchimento quantos queira. Se você omitir um argumento de preenchimento ou argumentos opcionais não use qualquer vírgula após o último que você especifica. Se deseja especificar um argumento opcional, você deve especificar todos os argumentos opcionais que aparecem na sintaxe antes dele.

## Método Point

Retorna, como número inteiro longo, a cor RGB (vermelho-verde-azul) do ponto especificado em um **Form** ou **PictureBox**. Não suporta argumentos nomeados.

### Sintaxe

*object*.**Point**(*x*, *y*)

A sintaxe do método **Point** tem estas partes:

Parte	Descrição
<i>object</i>	Opcional. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To. Se <i>object</i> for omitido, pressupõe-se que o objeto <b>Form</b> com o <u>foco</u> seja <i>object</i> .
<i>x</i> , <i>y</i>	Obrigatório. Valores de precisão simples indicando as coordenadas horizontal (eixo de x) e vertical (eixo de y) do ponto na propriedade <b>ScaleMode</b> do <b>Form</b> ou <b>PictureBox</b> . Os valores devem ser colocados entre parênteses.

### Comentários

Se o ponto referido pelas coordenadas *x* e *y* estiver fora de *object*, o método **Point** retorna -1.

## Método PopupMenu

Exibe um menu pop-up em um objeto **MDIForm** ou **Form** no local atual do mouse, ou em coordenadas específicas. Não suporta argumentos nomeados.

### Sintaxe

*object*.**PopupMenu** *menuname*, *flags*, *x*, *y*, *boldcommand*

A sintaxe do método **PopupMenu** tem estas partes:

Parte	Descrição
<i>object</i>	Opcional. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To. Se <i>object</i> for omitido, pressupõe-se que o formulário com o <u>foco</u> seja <i>object</i> .
<i>Menuname</i>	Obrigatório. O nome do menu pop-up a ser exibido. O menu especificado deve ter pelo menos um <u>sub-</u>

	<u>menu</u> .
<i>Flags</i>	Opcional. Um valor ou constante que especifica a localização e comportamento de um menu pop-up, conforme descrito em Configurações.
<i>X</i>	Opcional. Especifica a coordenada de x onde o menu pop-up é exibido. Se for omitido, será usada a coordenada do mouse.
<i>Y</i>	Opcional. Especifica a coordenada de y onde o menu pop-up é exibido. Se for omitido, será usada a coordenada do mouse.
<i>boldcommand</i>	Opcional. Especifica o nome de um controle de menu no menu pop-up para exibir sua legenda em texto negrito. Se for omitido, nenhum controle do menu pop-up aparecerá em negrito.

### Configurações

As configurações para *flags* são:

Constante (localização)	Valor	Descrição
<b>vbPopupMenuLeftAlign</b>	0	(Padrão) O lado esquerdo do menu pop-up está localizado em x.
<b>vbPopupMenuCenterAlign</b>	4	O menu pop-up está centralizado em x.
<b>vbPopupMenuRightAlign</b>	8	O lado direito do menu pop-up está localizado em x.
Constante (comportamento)	Valor	Descrição
<b>vbPopupMenuLeftButton</b>	0	(Padrão) Um item no menu pop-up reage a um clique de mouse somente quando você usa o botão esquerdo do mouse.
<b>vbPopupMenuRightButton</b>	2	Um item no menu pop-up reage a um clique de mouse quando você usa o botão direito ou o botão esquerdo do mouse.

**Observação:** O parâmetro *flags* não tem efeito algum sobre aplicativos que são executados sob o Microsoft Windows versão 3.0 ou anterior. Para especificar dois *flags*, combine uma constante para cada grupo usando o operador **Or**.

### Comentários

Estas constantes estão listadas na biblioteca de objetos Visual Basic (VB) no Object Browser. Você especifica a unidade de medida para as coordenadas x e y usando a propriedade **ScaleMode**. As coordenadas x e y definem onde a pop-up é exibida em relação ao formulário especificado. Se as coordenadas x e y não estão incluídas, o menu pop-up é exibido na localização atual do ponteiro do mouse.

Quando você exibe um menu pop-up, o código que aparece após a chamada do método **PopupMenu** não é executado até que o usuário escolha um comando no menu (caso em que o código para aquele evento Clic do comando é executado antes do código posterior à instrução **PopupMenu**) ou cancela o menu. Além disso, somente um menu pop-up pode ser exibido em um determinado momento; portanto, chamadas a este método são ignoradas se um menu pop-up já estiver sendo exibido, ou se um menu pull-down estiver aberto.

## Método PrintForm

Envia uma imagem bit-por-bit de um objeto **Form** à impressora.

### Sintaxe



---

## *object*.PrintForm

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To. Se *object* for omitido, pressupõe-se que o **Form** com o foco seja *object*.

### Comentários

**PrintForm** imprime todos os objetos visíveis e bitmaps do objeto **Form**. **PrintForm** também imprime elementos gráficos adicionados a um objeto **Form** ou controle **PictureBox** durante o tempo de execução se a propriedade **AutoRedraw** é **True** quando os elementos gráficos são desenhados.

A impressora usada por **PrintForm** é determinada pelas configurações do **Painel de controle** do sistema operacional.

## Método RemoveItem

Remove um item de uma **ListBox** ou controle **ComboBox** ou uma linha de e um controle **MS Flex Grid**. Não suporta argumentos nomeados.

### Sintaxe

*object*.RemoveItem *index*

A sintaxe do método **RemoveItem** tem estas partes:

Parte	Descrição
<i>object</i>	Obrigatório. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	Obrigatório. Número inteiro representando a posição dentro do objeto do item ou linha a ser removida. Para o primeiro item em uma <b>ListBox</b> ou <b>ComboBox</b> ou para a primeira linha em um controle <b>MS Flex Grid</b> , <i>index</i> = 0.

### Comentários

Uma **ListBox** ou **ComboBox** que seja acoplada a um controle **Data** não suporta o método **RemoveItem**.

## Método Scale

Define o sistema de coordenadas para um **Form**, **PictureBox** ou **Printer**. Não suporta argumentos nomeados.

### Sintaxe

*object*.Scale (*x1*, *y1*) - (*x2*, *y2*)

A sintaxe do método **Scale** tem estas partes:

Parte	Descrição
<i>object</i>	Opcional. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To. Se <i>object</i> for omitido, pressupõe-se que o objeto <b>Form</b> com o <u>foco</u> seja <i>object</i> .
<i>x1</i> , <i>y1</i>	Opcional. Valores de precisão simples indicando as coordenadas horizontal (eixo de x) e vertical (eixo de y) que definem o canto superior esquerdo de <i>object</i> . Os valores devem ser colocados entre parênteses. Se for omitido, o segundo conjunto de coordenadas também deve ser omitido.
<i>x2</i> , <i>y2</i>	Opcional. Valores de precisão simples indicando as coordenadas horizontal e vertical que definem o canto inferior direito de <i>object</i> . Os valores devem ser colocados entre parênteses. Se omitido, o primeiro conjunto de coordenadas também deve ser omitido.

### Comentários

O método **Scale** permite reconfigurar o sistema de coordenadas como qualquer escala escolhida. **Scale** afeta o sistema de coordenadas tanto para as instruções de elementos gráficos durante o

tempo de execução quanto a colocação de controles.

Se você usa **Scale** sem argumentos (ambos os conjuntos de coordenadas omitidos), ele reconfigura o sistema de coordenadas para twips.

## Métodos ScaleX, ScaleY

Converte o valor da largura e altura de um **Form**, **PictureBox** ou **Printer** de uma unidade de medida da propriedade **ScaleMode** para outra. Não suporta argumentos nomeados.

### Sintaxe

*object*.**ScaleX** (*width*, *fromscale*, *toscale*)

*object*.**ScaleY** (*height*, *fromscale*, *toscale*)

As sintaxes de método **ScaleX** e **ScaleY** têm estas partes:

Parte	Descrição
<i>object</i>	Opcional. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To. Se <i>object</i> for omitido pressupõe-se que o objeto <b>Form</b> com o <u>foco</u> seja <i>object</i> .
<i>width</i>	Obrigatório. Especifica o número de unidades de medida de <i>object</i> a serem convertidas.
<i>height</i>	Obrigatório. Especifica o número de unidades de medida de <i>object</i> a serem convertidas.
<i>fromscale</i>	Opcional. Uma constante ou valor especificando o sistema de coordenadas de onde <i>width</i> ou <i>height</i> de <i>object</i> deve ser convertidos, conforme descrito em Configurações. Os valores possíveis de <i>fromscale</i> são os mesmos que a propriedade <b>ScaleMode</b> , mais um novo valor de HiMetric.
<i>toscale</i>	Opcional. Uma constante ou valor especificando o sistema de coordenadas para o qual <i>width</i> ou <i>height</i> de <i>object</i> devem ser convertidos, conforme descrito em Configurações. Os valores possíveis de <i>toscale</i> são os mesmos que para a propriedade <b>ScaleMode</b> , mais o novo valor de HiMetric.

### Configurações

As configurações para *fromscale* e *toscale* são:

Constante	Valor	Descrição
<b>vbUser</b>	0	Definida pelo usuário: indica que a largura e altura de <i>object</i> estão definidas como um valor personalizado.
<b>vbTwips</b>	1	<u>Twip</u> (1440 twips por polegada lógica; 567 twips por centímetro lógico).
<b>vbPoints</b>	2	<u>Ponto</u> (72 pontos por polegada lógica).
<b>vbPixels</b>	3	<u>Pixel</u> (menor unidade de monitor ou resolução de impressora).
<b>vbCharacters</b>	4	Caractere (horizontal = 120 twips por unidade; vertical = 240 twips por unidade).
<b>vbInches</b>	5	Polegada.
<b>vbMillimeters</b>	6	Milímetro.
<b>vbCentimeters</b>	7	Centímetro.
<b>vbHiMetric</b>	8	HiMetric. Se <i>fromscale</i> for omitido, pressupõe-se que HiMetric seja o padrão.
<b>vbContainerPosition</b>	9	Determina a posição do controle.
<b>vbContainerSize</b>	10	Determina o tamanho do controle.

### Comentários

Os métodos **ScaleX** e **ScaleY** tomam um valor (*width* ou *height*), com sua unidade de medida especificada por *fromscale*, e o convertem para o valor correspondente na unidade de medida especificada por *toscale*.

---

Você também pode usar **ScaleX** e **ScaleY** com o método **PaintPicture**.

## Método SetAutoServerSettings

Define os valores de registro de Remote Automation para atender os requisitos de ActiveX e Remote Automation, incluindo definições de configuração e acesso a servidor remoto.

### Sintaxe

*object*. **SetAutoServerSettings**(*remote*, [*progid*], [*clsid*], [*servername*], [*protocol*], [*authentication*])

A sintaxe do método **SetAutoServerSettings** tem estas partes:

Parte	Descrição
<i>object</i>	Obrigatório. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>remote</i>	Obrigatório. Boolean. <b>True</b> se o servidor for remoto, <b>False</b> se for local.
<i>progid</i>	Opcional. Uma <u>expressão de variante</u> especificando a ProgID do servidor.
<i>clsid</i>	Opcional. Uma <u>expressão de variante</u> especificando o CLSID do servidor.
<i>servername</i>	Opcional. Uma <u>expressão de variante</u> especificando o nome da máquina servidor.
<i>protocol</i>	Opcional. Uma <u>expressão de variante</u> especificando o nome RPC do protocolo a ser utilizado.
<i>authentication</i>	Opcional. A <u>expressão de variante</u> especificando o nível de autenticação RPC.

### Valores de retorno

O método **SetAutoServerSettings** retorna os seguintes códigos de erro:

Valor	Descrição
0	Nenhum erro.
1	Ocorreu erro durante o tempo de execução desconhecido.
2	Nenhum protocolo foi especificado.
3	Nenhum nome de máquina servidora foi especificado.
4	Ocorreu um erro ao ler no registro.
5	Ocorreu um erro ao gravar no registro.
6	Ambos os parâmetros ProgID e CLSID faltavam.
7	Não existe servidor local (tanto em processo quanto entre processos, 16 bits ou 32 bits).
8	Ocorreu um erro ao procurar as DLLs Proxy, verifique se elas foram corretamente instaladas.

### Comentários

O método **SetAutoServerSettings** toma a CLSID ou a ProgID e configura as informações do registro como local ou remoto, dependendo do valor do parâmetro *remote*. Se tanto uma CLSID quanto uma ProgID são passadas ao método, o CLSID tem precedência.

## Método SetData

Coloca uma figura no objeto **Clipboard** usando o formato gráfico especificado. Não suporta argumentos nomeados.

### Sintaxe

*object*.**SetData** *data*, *format*

A sintaxe do método **SetData** tem estas partes:

Parte	Descrição
<i>object</i>	Obrigatório. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>data</i>	Obrigatório. Um elemento gráfico a ser colocado no objeto <b>Clipboard</b> .
<i>format</i>	Opcional. Uma constante ou valor que especifica um

dos formatos de objeto **Clipboard** reconhecidos pelo Visual Basic, conforme descrito em Configurações. Se *format* for omitido, **SetData** determina automaticamente o formato.

### Configurações

As configurações para *format* são:

Constante	Valor	Descrição
<b>vbCFBitmap</b>	2	<u>Bitmap</u> (arquivos .bmp)
<b>vbCFMetafile</b>	3	<u>Metarquivos</u> (arquivos .wmf)
<b>vbCFDIB</b>	8	Bitmap independente de dispositivo (DIB)
<b>vbCFPalette</b>	9	Paleta de cores

### Comentários

Estas constantes estão listadas na biblioteca de objetos Visual Basic (VB) no Object Browser.

Você define o elemento gráfico que deve ser colocado no objeto **Clipboard** com a função **LoadPicture** ou a propriedade **Picture** de um **Form**, **Image** ou **PictureBox**.

## Método SetText

Coloca uma sequência de caracteres de texto no objeto **Clipboard** usando o formato de objeto **Clipboard** especificado. Não suporta argumentos nomeados.

### Sintaxe

*object*.**SetText** *data*, *format*

A sintaxe do método **SetText** tem estas partes:

Parte	Descrição
<i>object</i>	Obrigatório. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>data</i>	Obrigatório. Dados de sequência de caracteres a serem colocados na <b>Área de transferência</b> .
<i>Format</i>	Opcional. Uma constante ou valor que especifica um dos formatos reconhecidos pelo Visual Basic, conforme descrito em Configurações.

### Configurações

As configurações para *format* são:

Constante	Valor	Descrição
<b>vbCFLin</b>	&HBF00	Informações de conversação DDE
<b>vbCFRTF</b>	&HBF01	RichText Format
<b>vbCFText</b>	1	(Padrão) texto

### Comentários

Estas constantes estão listadas na biblioteca de objetos Visual Basic (VB) no Object Browser.

## Método Show

Exibe o objeto **MDIForm** ou **Form**. Não suporta argumentos nomeados.

### Sintaxe

*object*.**Show** *style*, *ownerform*

A sintaxe do método **Show** tem estas partes:

Parte	Descrição
<i>object</i>	Opcional. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To. Se <i>object</i> for omitido, pressupõe-se que o formulário associado ao <u>módulo</u> do formulário ativo seja <i>object</i> .
<i>style</i>	Opcional. Número inteiro que determina se o formulário tem <u>janela restrita</u> ou <u>não</u> . Se <i>style</i> for 0, o formulário não tem janela restrita; se <i>style</i> for 1, o

*ownerform*

formulário tem janela restrita.

Opcional. Uma expressão de seqüência de caracteres que especifica o componente que é "proprietário" do formulário em exibição. Para formulário padrão do Visual Basic, use a palavra-chave **Me**

### Comentários

Se o formulário especificado não estiver carregado quando o método **Show** é acionado, o Visual Basic o carrega automaticamente.

Quando **Show** exibe um formulário sem janela restrita, o código subsequente é executado como ele é encontrado. Quando **Show** exibe um formulário de janela restrita, nenhum código subsequente é executado até que o formulário seja ocultado ou descarregado.

Quando **Show** exibe um formulário de janela restrita, nenhuma entrada (teclado ou clique de mouse) pode ocorrer, exceto em objetos no formulário de janela restrita. O programa deve ocultar ou descarregar um formulário de janela restrita (normalmente em resposta a alguma ação de usuário) antes que entrada em outro formulário possa ocorrer. Um **MDIForm** não pode ser de janela restrita.

Embora outros formulários em seu aplicativo estejam desativados quando um formulário de janela restrita é exibido, os outros aplicativos não estão.

O formulário de inicialização de um aplicativo é automaticamente exibido após seu evento Load ter sido acionado.

Aqui está um exemplo de como o argumento *ownerform* é usado com o método **Show**:

```
Private Sub cmdShowResults_Click()  
    ' Exibir um formulário de janela restrita chamado frmResults.  
    frmResults.Show vbModal, Me  
End Sub
```

## Método TextHeight

Retorna a altura de uma seqüência de caracteres de texto como ela seria impressa na fonte atual de um **Form**, **PictureBox** ou **Printer**. Não suporta argumentos nomeados.

### Sintaxe

*object.TextHeight(string)*

A sintaxe do método **TextHeight** tem estas partes:

Parte	Descrição
<i>object</i>	Opcional. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To. Se <i>object</i> for omitido, pressupõe-se que o objeto <b>Form</b> com o <u>foco</u> seja <i>object</i> .
<i>String</i>	Obrigatório. Uma <u>expressão de seqüência de caracteres</u> que avalia para uma seqüência de caracteres cuja altura de texto é determinada. A expressão de seqüência de caracteres deve estar entre parênteses.

### Comentários

A altura é expressa em termos da configuração da propriedade **ScaleMode** ou sistema de coordenadas do método **Scale** válidas para *object*. Use **TextHeight** para determinar a quantidade de espaço vertical exigido para exibir o texto. A altura retornada inclui o espaço normal que precede o texto, acima e abaixo, de modo que você pode usar a altura para calcular e posicionar linhas múltiplas de texto dentro de *object*.

Se *string* contém retornos de linha, **TextHeight** retorna a altura acumulada das linhas, incluindo o espaço que precede cada linha, acima e abaixo.

## Método TextWidth

Retorna a largura de uma seqüência de caracteres de texto, como ela seria impressa na fonte atual de um **Form**, **PictureBox** ou **Printer**. Não suporta argumentos nomeados.

### Sintaxe

*object.TextWidth(string)*

---

A sintaxe do método **TextWidth** tem estas partes:

Parte	Descrição
<i>object</i>	Opcional. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To. Se <i>object</i> for omitido, pressupõe-se que o <b>Form</b> com o <u>foco</u> seja <i>object</i> .
<i>String</i>	Obrigatório. Uma <u>expressão de sequência de caracteres</u> que avalia para uma sequência de caracteres cuja largura é determinada. A expressão de sequência de caracteres deve estar entre parênteses.

#### Comentários

A largura é expressa em termos da configuração da propriedades **ScaleMode** ou do sistema de coordenadas do método **Scale** válidos para *object*. Use **TextWidth** para determinar a quantidade de espaço horizontal exigido para exibir o texto. Se *string* contém retornos de linha incorporados, **TextWidth** retorna a largura da linha mais longa.

## Método ZOrder

Coloca um **MDIForm**, **Form** ou controle especificado à frente ou atrás na ordem-z dentro de seu nível gráfico. Não suporta argumentos nomeados.

#### Sintaxe

*object.ZOrder position*

A sintaxe do método **ZOrder** tem estas partes:

Parte	Descrição
<i>object</i>	Opcional. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To. Se <i>object</i> for omitido, pressupõe-se que o formulário com o <u>foco</u> seja <i>object</i> .
<i>Position</i>	Opcional. Número inteiro indicando a posição de <i>object</i> em relação a outras ocorrências do mesmo <i>object</i> . Se a posição for 0 ou omitida, <i>object</i> é posiciona na frente, na ordem-z. Se a posição for 1, <i>object</i> é posicionado atrás na ordem-z.

#### Comentários

A ordem-z de objetos pode ser definida durante o tempo de criação escolhendo o comando de menu **Bring To Front** ou **Send To Bac** no menu **Edit**.

Dentro de um objeto **MDIForm**, **ZOrder** envia formulários **MDI filho** para a frente ou para trás da área MDI cliente, dependendo do valor de *position*. Para um **MDIForm** ou objeto **Form**, **ZOrder** envia o formulário para a frente ou para trás da tela, dependendo do valor de *position*. Como resultado, os formulários podem ser exibidos à frente ou atrás de outros aplicativos em execução.

Três camadas gráficas são associadas a formulários e recipientes. A camada de trás é o espaço de desenho onde os resultados dos métodos gráficos são exibidos. A seguir está a camada média onde os objetos gráficos e os controles **Label** são exibidos. A camada frontal é onde todos os controles não-gráficos como **CommandButton**, **CheckBox** ou **ListBox** são exibidos. Qualquer coisa contida em uma camada mais próxima do primeiro plano cobre tudo o que está contido na(s) camada(s) atrás dele. **ZOrder** dispõe os objetos somente dentro da camada onde o objeto é exibido.

## Exemplo do método AddItem

Este exemplo utiliza o método **AddItem** para adicionar 100 itens a uma caixa de listagem. Para experimentar este exemplo, cole o código na seção Declarations de um formulário contendo um controle **ListBox** chamado List1 e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Click ()  
    Dim Entry, I, Msg ' Declarar variáveis.  
    Msg = "Choose 0 to add 100 items to your list box."  
    MsgBox Msg ' Exibir mensagem.  
    For I = 1 To 100 ' Contar de 1 a 100.  
        Entry = "Entry " & I ' Criar entrada.
```

```

        List1.AddItem Entry ' Adicionar a entrada.
    Next I
    Msg = "Choose 0 to remove every other entry."
    MsgBox Msg ' Exibir mensagem.
    For I = 1 To 50 ' Determinar como
        List1.RemoveItem I ' remover um em cada dois
    Next I ' itens.
    Msg = "Choose 0 to remove all items from the list box."
    MsgBox Msg ' Exibir mensagem.
    List1.Clear ' Limpar a caixa de listagem.
End Sub

```

## Exemplo do método Arrange

Este exemplo usa o método **Arrange** para organizar janelas e ícones em um formulário MDI. Para experimentar este exemplo, cole o código na seção Declarations de um formulário MDI chamado MDIForm1 que tenha um formulário MDI filho (chamado Form1, com sua propriedade **MDIChild** definida como **True**) e uma caixa de figura no formulário MDI (chamado Picture1). Pressione F5 e clique em qualquer lugar na caixa de figura para ver os efeitos do método **Arrange**.

```

Const FORMCOUNT = 5
Dim F(1 To FORMCOUNT) As New Form1
Private Sub MDIForm_Load ()
    Dim I ' Declarar a variável local.
    Load Form1 ' Carregar o Form1 original.
    For I = 1 To FORMCOUNT
        F(I).Caption = "Form" & I + 1 ' Alterar a legenda em cópias.
    Next I
End Sub

Private Sub Picture1_Clic ()
    Static ClicCount ' Declarar as variáveis.
    Dim I, PrevWidth, Start
    ClicCount = ClicCount + 1 ' Incrementar o contador de cliques.
    Select Case ClicCount
        Case 1
            MDIForm1.Arrange 1 ' Organizar lado a lado horizontalmente.
        Case 2
            MDIForm1.Arrange 2 ' Organizar lado a lado verticalmente.
        Case 3 ' Minimizar cada formulário.
            PrevWidth = MDIForm1.Width ' Obter a largura do formulário
MDI.
            MDIForm1.Width = PrevWidth / 2 ' Dividi-lo na metade.
            Form1.WindowState = 1 ' Minimizar o original.
            For I = 1 To FORMCOUNT ' Olhar cada ocorrência de F.
                F(I).WindowState = 1 ' Minimizar cada cópia de F.
            Next I
            Start = Timer
            Do
                Loop Until Timer = Start + 5
            MDIForm1.Width = PrevWidth ' Redimensionar ao tamanho ori-
ginal.
            MDIForm1.Arrange 3 ' Organizar os ícones.
        End Select
    End Sub

```

## Exemplo do método Clear

Este exemplo usa o método **Clear** para limpar todos os itens de uma caixa de listagem. Para experimentar este exemplo, cole o código na seção Declarations de um formulário com um controle **ListBox** chamado List1 e, em seguida, pressione F5 e clique no formulário.

```

Private Sub Form_Click ()
    Dim Entry, I, Msg ' Declarar as variáveis.
    Msg = "Choose 0 to remove every other entry."
    MsgBox Msg ' Exibir mensagem.

```

```

For I = 1 To 100 ' Contar de 1 a 100.
    Entry = "Entry " & I ' Criar entrada.
    List1.AddItem Entry ' Adicionar a entrada.
Next I
Msg = "Choose 0 to remove every other entry."
MsgBox Msg ' Exibir mensagem.
For I = 1 To 50 ' Determinar como
    List1.RemoveItem I ' remover um em cada dois
Next I ' itens.
Msg = "Choose 0 to remove all items from the list box."
MsgBox Msg ' Exibir mensagem.
List1.Clear ' Limpar a caixa de listagem.
End Sub

```

Este exemplo usa o método **Clear** para limpar o objeto **Clipboard**. Para experimentar este exemplo, cole o código na seção Declarations de um formulário e, em seguida, pressione F5 e clique no formulário.

```

Private Sub Form_Click ()
    Const CF_BITMAP = 2 ' Definir o formato de bitmap.
    Dim Msg ' Declarar a variável.
    On Error Resume Next ' Configurar o tratamento de erro.
    Msg = "Choose 0 to load a bitmap onto the Clipboard."
    MsgBox Msg ' Exibir mensagem.
    Clipboard.Clear ' Limpar Clipboard.
    Clipboard.SetData LoadPicture("PAPER.BMP") ' Obter bitmap.
    If Err Then
        Msg = "Can't find the .BMP file."
        MsgBox Msg ' Exibir mensagem de erro.
        Exit Sub
    End If
    Msg = "A bitmap is now on the Clipboard. Choose 0 to copy "
    Msg = Msg & "the bitmap from the Clipboard to the form."
    MsgBox Msg ' Exibir mensagem.
    Picture = Clipboard.GetData() ' Copiar da Área de transferência.
    Msg = "Choose 0 to clear the picture."
    MsgBox Msg ' Exibir mensagem.
    Picture = LoadPicture() ' Limpar a figura.
End Sub

```

## Exemplo do método Cls

Este exemplo usa o método **Cls** para excluir informações impressas de um formulário. Para experimentar este exemplo, cole o código na seção Declarations de um formulário e, em seguida, pressione F5 e clique no formulário.

```

Private Sub Form_Click ()
    Dim Msg ' Declarar variável.
    AutoRedraw = -1 ' Ativa AutoRedraw.
    ForeColor = QBColor(15) ' Configurar o primeiro plano como branco.
    BackColor = QBColor(1) ' Configurar o segundo plano como azul.
    FillStyle = 7 ' Configurar hachura diagonal.
    Line (0, 0)-(ScaleWidth, ScaleHeight), , B ' Colocar uma caixa no
    formulário.
    Msg = "This is information printed on the form background."
    CurrentX = ScaleWidth / 2 - TextWidth(Msg) / 2 ' Configura a posição
    X.
    CurrentY = 2 * TextHeight(Msg) ' Configurar a posição Y.
    Print Msg ' Imprimir mensagem em formulário.
    Msg = "Choose 0 to clear the information and background "
    Msg = Msg & "pattern just displayed on the form."
    MsgBox Msg ' Exibir mensagem.
    Cls ' Limpar o segundo plano do formulário.
End Sub

```



---

## Exemplo do método Drag

Este exemplo utiliza o método **Drag** para arrastar o nome do arquivo de um arquivo bitmap (.bmp) para uma caixa de figura onde o bitmap é exibido. Para experimentar este exemplo, cole todo o código na seção Declarations de um formulário que contenha controles **DriveListBox**, **DirListBox**, **FileListBox**, **PictureBox** e **Label**. Use os nomes padrão para todos os controles. Dimensione e posicione todos os controles de modo que eles possam ser facilmente vistos e utilizados. O tamanho e posição do rótulo não é importante pois ele é alterado durante o tempo de execução. Quando o programa se inicia, você pode pesquisar seu sistema de arquivo e carregar quaisquer bitmaps. Uma vez que tenha localizado um bitmap que queira exibir, clique no nome do arquivo daquele bitmap e arraste-o para a caixa de figura.

```
Private Sub Form_Load ()
    Picture1.AutoSize = -1 ' Ativar o AutoSize.
    Label1.Visible = 0 ' Tornar o rótulo invisível .
    File1.Pattern = "*.BMP; *.ICO; *.WMF" ' Definir padrões de arquivo.
End Sub

Private Sub Dir1_Change () ' Qualquer alteração em Dir1
    File1.Path = Dir1.Path ' é refletida em File1.
End Sub

Private Sub Drive1_Change () ' Qualquer alteração em Drive1
    Dir1.Path = Drive1.Drive ' é refletida em Dir1.
End Sub

Private Sub File1_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
    Dim DY ' Declarar variável.
    DY = TextHeight("A") ' Obter a altura de uma linha.
    Label1.Move File1.Left, File1.Top + Y - DY / 2, File1.Width, DY
    Label1.Drag ' Arrastar o contorno do rótulo.
End Sub

Private Sub Dir1_DragOver (Source As Control, X As Single, Y As Single, State As Integer)
    ' Alterar o ponteiro para não-soltar.
    If State = 0 Then Source.MousePointer = 12
    ' Usar o ponteiro de mouse padrão.
    If State = 1 Then Source.MousePointer = 0
End Sub

Private Sub Drive1_DragOver (Source As Control, X As Single, Y As Single, State As Integer)
    ' Alterar ponteiro para não-soltar.
    If State = 0 Then Source.MousePointer = 12
    ' Usar o ponteiro de mouse padrão.
    If State = 1 Then Source.MousePointer = 0
End Sub

Private Sub Form_DragOver (Source As Control, X As Single, Y As Single, State As Integer)
    ' Alterar o ponteiro para não-soltar.
    If State = 0 Then Source.MousePointer = 12
    ' Usar o ponteiro de mouse padrão.
    If State = 1 Then Source.MousePointer = 0
End Sub

Private Sub File1_DragOver (Source As Control, X As Single, Y As Single, State As Integer)
    On Error Resume Next
    If State = 0 And Right$(File1.FileName, 4) = ".ICO" Then
        Label1.DragIcon = LoadPicture(File1.Path + "\" + File1.FileName)
    If Err Then MsgBox "The icon file can't be loaded."
```

```

ElseIf State = 1 Then
    Label1.DragIcon = LoadPicture () ' Usar o ícone não-arrastar.
End If
End Sub

Private Sub Picture1_DragDrop (Source As Control, X As Single, Y As Single)
    On Error Resume Next
    Picture1.Picture = LoadPicture(File1.Path + "\" + File1.FileName)
    If Err Then MsgBox "The picture file can't be loaded."
End Sub

```

## Exemplo do método EndDoc

Este exemplo usa o método **EndDoc** para encerrar um documento após imprimir duas páginas, cada uma delas com uma linha centralizada de texto indicando o número da página. Para experimentar este exemplo, cole o código na seção Declarations de um formulário e, em seguida, pressione F5 e clique no formulário.

```

Private Sub Form_Click ()
    Dim HWidth, HHeight, I, Msg ' Declarar as variáveis.
    On Error GoTo ErrorHandler ' Configurar manipulador de erro.
    Msg = "This is printed on page"
    For I = 1 To 2 ' Configura duas iterações.
        HWidth = Printer.TextWidth(Msg) / 2 ' Obter meia largura.
        HHeight = Printer.TextHeight(Msg) / 2 ' Obter meia altura.
        Printer.CurrentX = Printer.ScaleWidth / 2 - HWidth
        Printer.CurrentY = Printer.ScaleHeight / 2 - HHeight
        Printer.Print Msg & Printer.Page & "." ' Imprimir.
        Printer.NewPage ' Enviar nova página.
    Next I
    Printer.EndDoc ' Impressão está concluída.
    Msg = "Two pages, each with a single, centered line of text, "
    Msg = Msg & "have been sent to your printer."
    MsgBox Msg ' Exibir mensagem.
    Exit Sub
ErrorHandler:
    MsgBox "There was a problem printing to your printer."
    Exit Sub
End Sub

```

## Exemplo do método GetData

Este exemplo usa o método **GetData** para copiar um bitmap do objeto **Clipboard** para um formulário. Para experimentar este exemplo, cole o código na seção Declarations de um formulário, pressione F5 e clique no formulário.

```

Private Sub Form_Click ()
    Const CF_BITMAP = 2 ' Definir o formato bitmap.
    Dim Msg ' Declara variável.
    On Error Resume Next ' Configurar o tratamento de erro.
    Msg = "Choose O to load a bitmap onto the Clipboard."
    MsgBox Msg ' Exibir mensagem.
    Clipboard.Clear ' Limpar a Área de transferência.
    Clipboard.SetData LoadPicture("PAPER.BMP") ' Obter o bitmap.
    If Err Then
        Msg = "Can't find the .bmp file."
        MsgBox Msg ' Exibir mensagem de erro.
        Exit Sub
    End If
    Msg = "A bitmap is now on the Clipboard. Choose O to copy "
    Msg = Msg & "the bitmap from the Clipboard to the form "
    MsgBox Msg ' Exibir mensagem.
    Picture = Clipboard.GetData() ' Copiar da Área de transferência.
    Msg = "Choose O to clear the form."
    MsgBox Msg ' Exibir mensagem.
    Picture = LoadPicture() ' Limpar o formulário.

```

---

End Sub

## Exemplo do método GetFormat

Este exemplo usa o método **GetFormat** para determinar o formato dos dados no objeto **Clipboard**. Para experimentar este exemplo, cole o código na seção Declarations de um formulário e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Click ()
    ' Definir formatos de bitmap.
    Dim ClpFmt, Msg ' Declarar variáveis.
    On Error Resume Next ' Configurar tratamento de erro.
    If Clipboard.GetFormat(vbCFText) Then ClpFmt = ClpFmt + 1
    If Clipboard.GetFormat(vbCFBitmap) Then ClpFmt = ClpFmt + 2
    If Clipboard.GetFormat(vbCFDIB) Then ClpFmt = ClpFmt + 4
    If Clipboard.GetFormat(vbCFRTF) Then ClpFmt = ClpFmt + 8
    Select Case ClpFmt
        Case 1
            Msg = "The Clipboard contains only text."
        Case 2, 4, 6
            Msg = "The Clipboard contains only a bitmap."
        Case 3, 5, 7
            Msg = "The Clipboard contains text and a bitmap."
        Case 8, 9
            Msg = "The Clipboard contains only rich text."
        Case Else
            Msg = "There is nothing on the Clipboard."
    End Select
    MsgBox Msg ' Exibir mensagem.
End Sub
```

---

## Exemplo do método GetText

Este exemplo usa o método **GetText** para copiar uma sequência de caracteres de texto do objeto **Clipboard** para uma variável de sequência de caracteres. Para experimentar este exemplo, cole o código na seção Declarations de um formulário com um controle **TextBox** chamado Text1 e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Click ()
    Dim I, Msg, Temp ' Declarar variáveis.
    On Error Resume Next ' Configurar tratamento de erro.
    Msg = "Type anything you lie into the text box below."
    Text1.Text = InputBox(Msg) ' Obter o texto do usuário.
    Msg = "Choose O to copy the contents of the text box "
    Msg = Msg & "to the Clipboard."
    MsgBox Msg ' Exibir mensagem.
    Clipboard.Clear ' Limpar a Área de transferência.
    Clipboard.SetText Text1.Text ' Colocar texto na Área de transferência.
    If Clipboard.GetFormat(vbCFText) Then
        Text1.Text = "" ' Limpar a caixa de texto.
        Msg = "The text is now on the Clipboard. Choose O "
        Msg = Msg & "to copy the text from the Clipboard bac "
        Msg = Msg & "to the text box."
        MsgBox Msg ' Exibir mensagem.
        Temp = Clipboard.GetText(vbCFText) ' Obter o texto da Área de transferência.
        For I = Len(Temp) To 1 Step -1 ' Inverter o texto.
            Text1.Text = Text1.Text & Mid(Temp, I, 1)
        Next I
    Else
        Msg = "There is no text on the Clipboard."
        MsgBox Msg ' Exibir mensagem de erro.
    End If
End Sub
```

## Exemplo do método Hide

Este exemplo usa o método **Hide** para ocultar um formulário. Para experimentar este exemplo, cole o código na seção Declarations de um formulário não-MDI e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Click ()
    Dim Msg ' Declarar variável.
    Hide ' Ocultar formulário.
    Msg = "Choose O to make the form reappear."
    MsgBox Msg ' Exibir mensagem.
    Show ' Mostrar o formulário novamente.
End Sub
```

## Exemplo do método illDoc

Este exemplo usa o método **illDoc** para finalizar o trabalho atual de impressão. Para experimentar este exemplo, cole o código na seção Declarations de um formulário e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Click()
    For i = 1 To 40
        Printer.CurrentX = 1440 ' Configurar a margem esquerda.
        Printer.CurrentY = (i * 300) ' Avançar a página para a próxima linha.
        Printer.Print "This is line" & Str$(i) & " of text."
        On Error Resume Next ' Capturar qualquer erro de impressora.
        If i = 26 Then
            Printer.illDoc ' Finalizar abruptamente o trabalho de impressão.
        End If
    Next i
    Printer.EndDoc
End Sub
```

```
Next i
End Sub
```

## Exemplo do método LinExecute

Este exemplo estabelece um vínculo DDE com o Microsoft Excel, coloca alguns valores em células na primeira linha de uma nova planilha e transforma os valores em gráficos. **LinExecute** envia ao Microsoft Excel o comando para ativar uma planilha, selecionar alguns valores e transformá-los em gráfico. Para experimentar este exemplo, o Microsoft Excel deve estar instalado em seu computador e na instrução de caminho de seu arquivo Autoexec.bat. Cole o código na seção **Declarations** de um formulário que contenha um controle **TextBox** com o nome padrão Text1 e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Click ()
    Dim Cmd, I, Q, Row, Z ' Declarar variáveis.
    Q = Chr(34) ' Definir aspas.
    ' Criar uma sequência de caracteres contendo comandos de macro do Microsoft Excel.
    Cmd = "[ACTIVATE(" & Q & "SHEET1" & Q & ")]"
    Cmd = Cmd & "[SELECT(" & Q & "R1C1:R5C2" & Q & ")]"
    Cmd = Cmd & "[NEW(2,1)][ARRANGE.ALL()]"
    If Text1.LinMode = vbNone Then
        Z = Shell("Excel", 4) ' Iniciar o Microsoft Excel.
        Text1.LinTopic = "Excel|Sheet1" ' Definir tópico de vinculação.
        Text1.LinItem = "R1C1" ' Definir o item de vinculação.
        Text1.LinMode = vbLinManual ' Definir o modo de vinculação.
    End If
    For I = 1 To 5
        Row = I ' Definir o número da linha.
        Text1.LinItem = "R" & Row & "C1" ' Definir o item de vinculação.
        Text1.Text = Chr(64 + I) ' Colocar valor em Text.
        Text1.LinPoe ' Enviar o valor à célula.
        Text1.LinItem = "R" & Row & "C2" ' Definir o item de vinculação.
        Text1.Text = Row ' Colocar o valor em Text.
        Text1.LinPoe ' Enviar o valor à célula.
    Next I
    On Error Resume Next
    Text1.LinExecute Cmd ' Executar os comandos do Microsoft Excel.
    MsgBox "LinExecute DDE demo with Microsoft Excel finished.", 64
End Sub
```

## Exemplo do método LinPoe

Este exemplo estabelece um vínculo DDE com o Microsoft Excel, coloca alguns valores em células na primeira linha de uma nova planilha e transforma os valores em gráfico. **LinPoe** envia os valores a serem transformados em gráfico à planilha do Microsoft Excel. Para experimentar este exemplo, o Microsoft Excel deve estar instalado e na instrução de caminho de seu arquivo Autoexec.bat. Cole o código na seção **Declarations** de um formulário que contenha uma caixa **TextBox** com o nome padrão Text1 e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Click ()
    Dim Cmd, I, Q, Row, Z ' Declarar variáveis.
    Q = Chr(34) ' Definir aspas.
    ' Criar uma sequência de caracteres contendo comandos de macro do Microsoft Excel.
    Cmd = "[ACTIVATE(" & Q & "SHEET1" & Q & ")]"
    Cmd = Cmd & "[SELECT(" & Q & "R1C1:R5C2" & Q & ")]"
    Cmd = Cmd & "[NEW(2,1)][ARRANGE.ALL()]"
    If Text1.LinMode = vbNone Then
        Z = Shell("Excel", 4) ' Iniciar o Microsoft Excel.
        Text1.LinTopic = "Excel|Sheet1" ' Definir tópico de vinculação.
        Text1.LinItem = "R1C1" ' Definir o item de vinculação.
        Text1.LinMode = vbLinManual ' Definir o modo de vinculação.
    End If
    For I = 1 To 5
        Row = I ' Definir número de linha.
```

```

Text1.LinItem = "R" & Row & "C1" ' Definir o item de vinculação.
Text1.Text = Chr(64 + I) ' Colocar valor em Text.
Text1.LinPoe ' Enviar o valor à célula.
Text1.LinItem = "R" & Row & "C2" ' Definir o item de vinculação.
Text1.Text = Row ' Colocar valor em Text.
Text1.LinPoe ' Enviar valor à célula.
Next I
Text1.LinExecute Cmd ' Executar os comandos do Microsoft Excel.
On Error Resume Next
MsgBox "LinPoe DDE demo with Microsoft Excel finished.", 64
End
End Sub

```

## Exemplo do método LinRequest

Este exemplo utiliza **LinRequest** para atualizar o conteúdo de uma caixa de texto com os valores contidos em uma planilha do Microsoft Excel. Para experimentar este exemplo, você deve ter o Microsoft Excel em execução em seu computador. Coloque alguns dados nas primeiras células da primeira coluna na planilha padrão (Sheet1.xls). Cole o código na seção Declarations de um formulário que tenha um controle **TextBox** chamado Text1 e, em seguida, pressione F5 e clique no formulário.

```

Private Sub Form_Click ()
    If Text1.LinMode = vbNone Then ' Testar o modo de vinculação.
        Text1.LinTopic = "Excel|Sheet1" ' Definir o tópico de vinculação.

        Text1.LinItem = "R1C1" ' Definir o item de vinculação.
        Text1.LinMode = vbLinManual ' Definir o modo de vinculação.
        Text1.LinRequest ' Atualizar a caixa de texto.
    Else
        If Text1.LinItem = "R1C1" Then
            Text1.LinItem = "R2C1"
            Text1.LinRequest ' Atualizar a caixa de texto.
        Else
            Text1.LinItem = "R1C1"
            Text1.LinRequest ' Atualizar a caixa de texto.
        End If
    End If
End Sub

```

## Exemplo do método Move

Este exemplo usa o método **Move** para mover um formulário pela tela. Para experimentar este exemplo, cole o código na seção Declarations de um formulário e, em seguida, pressione F5 e clique no formulário.

```

Private Sub Form_Click ()
    Dim Inch, Msg ' Declarar variáveis.
    Msg = "Choose O to resize and move this form by "
    Msg = Msg & "changing the value of properties."
    MsgBox Msg ' Exibir mensagem.
    Inch = 1440 ' Definir polegadas em twips.
    Width = 4 * Inch ' Definir a largura.
    Height = 2 * Inch ' Definir a altura.
    Left = 0 ' Definir a esquerda como origem.
    Top = 0 ' Definir a parte superior como origem.
    Msg = "Now choose O to resize and move this form "
    Msg = Msg & "using the Move method."
    MsgBox Msg ' Exibir mensagem.
    Move Screen.Width - 2 * Inch, Screen.Height - Inch, 2 * Inch, Inch
End Sub

```

## Exemplo do método NewPage

Este exemplo usa o método **NewPage** para iniciar uma nova página de impressão após imprimir uma linha de texto única, centralizada em uma página. Para experimentar este exemplo, cole o código na seção Declarations de um formulário e, em seguida, pressione F5 e clique no formulário.

---

```

Private Sub Form_Clic ()
    Dim HWidth, HHeight, I, Msg      ' Declarar variáveis.
    On Error GoTo ErrorHandler      ' Definir manipulador de erro.
    Msg = "This is printed on page"
    For I = 1 To 2 ' Configurar duas iterações.
        HWidth = Printer.TextWidth(Msg) / 2      ' Obter uma metade de
largura.
        HHeight = Printer.TextHeight(Msg) / 2      ' Obter uma metade de al-
tura.
        Printer.CurrentX = Printer.ScaleWidth / 2 - HWidth
        Printer.CurrentY = Printer.ScaleHeight / 2 - HHeight
        Printer.Print Msg & Printer.Page & "."      ' Imprime.
        Printer.NewPage      ' Enviar nova página.
    Next I
    Printer.EndDoc ' Impressão terminada.
    Msg = "Two pages, each with a single, centered line of text, "
    Msg = Msg & "have been sent to your printer."
    MsgBox Msg ' Exibir mensagem.
Exit Sub
ErrorHandler:
    MsgBox "There was a problem printing to your printer."
    Exit Sub
End Sub

```

## Exemplo do método Point

Este exemplo usa o método **Point** para determinar a cor de um ponto específico em um formulário. Para experimentar este exemplo, cole o código na seção Declarations de um formulário e, em seguida, pressione F5 e clique no formulário.

```

Private Sub Form_Clic ()
    Dim LeftColor, MidColor, Msg, RightColor      ' Declarar variáveis.
    AutoRedraw = -1      ' Ativar AutoRedraw.
    Height = 3 * 1440 ' Definir a altura para 3 polegadas.
    Width = 5 * 1440 ' Definir a largura para 5 polegadas.
    BacColor = QBColor(1)      ' Definir o segundo plano para azul.
    ForeColor = QBColor(4)      ' Definir o primeiro plano para vermelho.
    Line (0, 0)-(Width / 3, Height), , BF ' Caixa vermelha.
    ForeColor = QBColor(15) ' Definir o primeiro plano para branco.
    Line (Width / 3, 0)-((Width / 3) * 2, Height), , BF
    LeftColor = Point(0, 0) ' Localizar a cor da caixa da esquerda,
    MidColor = Point(Width / 2, Height / 2)      ' caixa do centro e
    RightColor = Point(Width, Height)      ' caixa da direita.
    Msg = "The color number for the red box on the left side of "
    Msg = Msg & "the form is " & LeftColor & ". A "
    Msg = Msg & "color of the white box in the center is "
    Msg = Msg & MidColor & ". The color of the blue "
    Msg = Msg & "box on the right is " & RightColor & "."
    MsgBox Msg ' Exibir mensagem.
End Sub

```

---

## Exemplo do método PopupMenu

Este exemplo exibe um menu pop-up no local do cursor quando o usuário clica o botão direito do mouse em um formulário. Para experimentar este exemplo, crie um formulário que inclua um controle **Menu** chamado mnuFile (mnuFile deve ter pelo menos um submenu). Copie o código na seção Declarations do formulário e pressione F5.

```
Private Sub Form_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = 2 Then
        PopupMenu mnuFile
    End If
End Sub
```

## Exemplo do método PrintForm

Este exemplo usa o método **PrintForm** para imprimir o formulário atual. Para experimentar este exemplo, cole o código na seção Declarations de um formulário. Coloque no formulário todos os controles que deseja ver no formulário impresso e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Clic ()
    Dim Msg ' Declarar variável.
    On Error GoTo ErrorHandler ' Configurar manipulador de erro.
    PrintForm ' Imprimir formulário.
    Exit Sub
ErrorHandler:
    Msg = "The form can't be printed."
    MsgBox Msg ' Exibir mensagem.
    Resume Next
End Sub
```

## Exemplo do método RemoveItem

Este exemplo usa o método **RemoveItem** para remover entradas de uma caixa de listagem. Para experimentar este exemplo, cole o código na seção Declarations de um formulário com um controle **ListBox** chamado List1 e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Clic ()
    Dim Entry, I, Msg ' Declarar variáveis.
    Msg = "Choose 0 to add 100 items to your list box."
    MsgBox Msg ' Exibir mensagem.
    For I = 1 To 100 ' Contar de 1 a 100.
        Entry = "Entry " & I ' Criar entrada.
        List1.AddItem Entry ' Adicionar a entrada.
    Next I
    Msg = "Choose 0 to remove every other entry."
    MsgBox Msg ' Exibir mensagem.
    For I = 1 To 50 ' Determinar como
        List1.RemoveItem I ' remover um a cada dois
    Next I ' itens.
    Msg = "Choose 0 to remove all items from the list box."
    MsgBox Msg ' Exibir mensagem.
    List1.Clear ' Limpar a caixa de listagem.
End Sub
```

## Exemplo do método Scale

Este exemplo usa o método **Scale** para configurar um sistema de coordenadas personalizado, de modo que um gráfico de barras possa ser desenhado em um formulário. Para experimentar este exemplo, cole o código na seção Declarations de um formulário e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Clic ()
    Dim I, OldFontSize ' Declarar variáveis.
    Width = 8640: Height = 5760 ' Definir o tamanho do formulário em twips.
    Move 100,100 ' Mover a origem do formulário.
    AutoRedraw = -1 ' Ativar AutoRedraw.
    OldFontSize = FontSize ' Salvar o tamanho de fonte antigo.
```



```

BacColor = QBColor(7) ' Definir o segundo plano como cinza.
Scale (0, 110)-(130, 0) ' Definir o sistema de coordenadas personali-
zado.
For I = 100 To 10 Step -10
    Line (0, I)-(2, I) ' Desenhar marcas de escala a cada 10 unida-
des.
    CurrentY = CurrentY + 1.5 ' Mover a posição do cursor.
    Print I ' Imprimir o valor da marca de escala à esquerda.
    Line (ScaleWidth - 2, I)-(ScaleWidth, I)
    CurrentY = CurrentY + 1.5 ' Mover a posição do cursor.
    CurrentX = ScaleWidth - 9
    Print I ' Imprimir o valor da marca de escala à direita.
Next I
' Desenhar gráfico de barras.
Line (10, 0)-(20, 45), RGB(0, 0, 255), BF ' Primeira barra azul.
Line (20, 0)-(30, 55), RGB(255, 0, 0), BF ' Primeira barra verme-
lha.
Line (40, 0)-(50, 40), RGB(0, 0, 255), BF
Line (50, 0)-(60, 25), RGB(255, 0, 0), BF
Line (70, 0)-(80, 35), RGB(0, 0, 255), BF
Line (80, 0)-(90, 60), RGB(255, 0, 0), BF
Line (100, 0)-(110, 75), RGB(0, 0, 255), BF
Line (110, 0)-(120, 90), RGB(255, 0, 0), BF
CurrentX = 18: CurrentY = 100 ' Mover a posição do cursor.
FontSize = 14 ' Ampliar a fonte para o título.
Print "Widget Quarterly Sales" ' Imprimir o título.
FontSize = OldFontSize ' Restaurar o tamanho de fonte.
CurrentX = 27: CurrentY = 93 ' Mover a posição do cursor.
Print "Planned Vs. Actual" ' Imprimir subtítulo.
Line (29, 86)-(34, 88), RGB(0, 0, 255), BF ' Imprimir a legenda.
Line (43, 86)-(49, 88), RGB(255, 0, 0), BF
End Sub

```

## Exemplo do método SetText

Este exemplo usa o método **SetText** para copiar texto de uma caixa de texto para a **Área de transferência**. Para experimentar este exemplo, cole o código na seção Declarations de um formulário com uma caixa de texto chamada Text1 e, em seguida, pressione F5 e clique no formulário.

```

Private Sub Form_Clic ()
    Const CF_TEXT = 1 ' Definir o formato de bitmap.
    Dim I, Msg, Temp ' Declarar variáveis.
    On Error Resume Next ' Configurar tratamento de erro.
    Msg = "Type anything you lie into the text box below."
    Text1.Text = InputBox(Msg) ' Obter o texto do usuário.
    Msg = "Choose O to copy the contents of the text box "
    Msg = Msg & "to the Clipboard."
    MsgBox Msg ' Exibir mensagem.
    Clipboard.Clear ' Limpar a Área de transferência.
    Clipboard.SetText Text1.Text ' Colocar texto na Área de transfe-
rência.
    If Clipboard.GetFormat(CF_TEXT) Then
        Text1.Text = "" ' Limpar a caixa de texto.
        Msg = "The text is now on the Clipboard. Choose O "
        Msg = Msg & "to copy the text from the Clipboard"
        Msg = Msg & "to the text box."
        MsgBox Msg ' Exibir mensagem.
        Temp = Clipboard.GetText(CF_TEXT) ' Obter o texto da Área de
transferência .
        For I = Len(Temp) To 1 Step -1 ' Inverter o texto.
            Text1.Text = Text1.Text & Mid(Temp, I, 1)
        Next I
    Else
        Msg = "There is no text on the Clipboard."
        MsgBox Msg ' Exibir mensagem de erro.
    End If
End Sub

```

---

```
End If
End Sub
```

## Exemplo do método Show

Este exemplo usa o método **Show** para mostrar um formulário oculto. Para experimentar este exemplo, cole o código na seção Declarations de um formulário não-MDI e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Click ()
    Dim Msg ' Declarar variável.
    Hide ' Ocultar formulário.
    Msg = "Choose O to make the form reappear."
    MsgBox Msg ' Exibir mensagem.
    Show ' Mostrar o formulário novamente.
End Sub
```

## Exemplo do método TextHeight

O método **TextHeight** é usado para centralizar uma linha de texto verticalmente em um formulário. Para experimentar este exemplo, cole o código na seção Declarations de um formulário e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Click ()
    Dim HalfWidth, HalfHeight, Msg ' Declarar variável.
    AutoRedraw = -1 ' Ativa AutoRedraw.
    BackColor = QBColor(4) ' Definir a cor de segundo plano.
    ForeColor = QBColor(15) ' Definir a cor de primeiro plano.
    Msg = "Visual Basic" ' Criar mensagem.
    FontSize = 48 ' Definir a cor da fonte.
    HalfWidth = TextWidth(Msg) / 2 ' Calcular metade da largura.
    HalfHeight = TextHeight(Msg) / 2 ' Calcular metade da altura.
    CurrentX = ScaleWidth / 2 - HalfWidth ' Definir X.
    CurrentY = ScaleHeight / 2 - HalfHeight ' Definir Y.
    Print Msg ' Imprimir a mensagem.
End Sub
```

## Exemplo do método TextWidth

O método **TextWidth** é utilizado para centralizar uma linha de texto horizontalmente em um formulário. Para experimentar este exemplo, cole o código na seção Declarations de um formulário e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Click ()
    Dim HalfHeight, HalfWidth, Msg ' Declarar variáveis.
    AutoRedraw = -1 ' Ativa AutoRedraw.
    BackColor = QBColor(4) ' Definir a cor de segundo plano.
    ForeColor = QBColor(15) ' Definir a cor de primeiro plano.
    Msg = "Visual Basic" ' Criar mensagem.
    FontSize = 48 ' Definir tamanho de fonte.
    HalfWidth = TextWidth(Msg) / 2 ' Calcular metade da largura.
    HalfHeight = TextHeight(Msg) / 2 ' Calcular metade da altura.
    CurrentX = ScaleWidth / 2 - HalfWidth ' Definir X.
    CurrentY = ScaleHeight / 2 - HalfHeight ' Definir Y.
    Print Msg ' Imprimir a mensagem.
End Sub
```

---

## Exemplo da função GetAutoServerSettings

Este exemplo recupera informações sobre um objeto remotamente registrado chamado "Hello":

```
Sub ViewHello()  
    Dim oRegClass As New RegClass  
    Dim vRC As Variant  
    vRC = oRegClass.GetAutoServerSettings _  
        ("HelloProj.HelloClass")  
    If Not(IsEmpty(vRC)) Then  
        If vRC(1) Then  
            MsgBox "Hello is registered remotely on a " _  
                & "server named: " & vRC(1)  
        Else  
            MsgBox "Hello is registered locally."  
        End If  
    End If  
End Sub
```

## Exemplo do método SetAutoServerSettings

Este exemplo alterna um servidor chamado "Hello" de registro local para remoto e, em seguida, retorna à situação anterior:

```
Sub SwitchHello()  
    Dim oRegClass As New RegClass  
    ' Registrar Hello para ser executado remotamente  
    ' em uma máquina chamada Server1.  
    oRegClass.SetAutoServerSettings True, _  
        "HelloProj.HelloClass", 1 _  
        ServerName:="Server1", Protocol:="ncacn_ip_tcp"  
    ' Registrar Hello para ser novamente executado localmente.  
    oRegClass.SetAutoServerSettings False, _  
        "HelloProj.HelloClass"  
End Sub
```

## Objeto App

O objeto **App** é um objeto global acessado com a palavra-chave **App**. Ele determina ou especifica informações sobre o título do aplicativo, informações de versão, o caminho e nome de seu arquivo executável e arquivos de **Ajuda**, e se uma ocorrência anterior do aplicativo está ou não sendo executada.

### Sintaxe

**App**

## Controle CheckBox

Um controle **CheckBox** exibe um X ao ser selecionado; o X desaparece quando a seleção de **CheckBox** é anulado. Use este controle para oferecer ao usuário uma opção **True/False** ou **Yes/No**. Você pode usar controles **CheckBox** em grupos para exibir múltiplas escolhas onde o usuário pode selecionar um ou mais. Você também pode configurar o valor de um **CheckBox** utilizando programa com a propriedade **Value**.

### Sintaxe

**CheckBox**

### Comentários

Os controles **CheckBox** e **OptionButton** funcionam de maneira semelhante, mas com uma diferença importante: qualquer número de controles **CheckBox** em um formulário pode ser selecionado ao mesmo tempo. Por outro lado, somente um **OptionButton** em um grupo pode ser selecionado em um determinado momento.

Para exibir texto junto ao **CheckBox**, configure a propriedade **Caption**. Use a propriedade **Value** para determinar o estado do controle — selecionado, limpo ou indisponível.

## Controle ComboBox

Um controle **ComboBox** combina os recursos de um controle **TextBox** e um controle **ListBox** —

---

os usuários podem inserir informações na parte caixa de texto ou selecionar um item da parte caixa de listagem do controle.

#### Sintaxe

#### ComboBox

#### Comentários

Para adicionar ou excluir itens em um controle **ComboBox**, use o método **AddItem** ou **RemoveItem**. Configure as propriedades **List**, **ListCount** e **ListIndex** para permitir a um usuário acessar itens no **ComboBox**. Como alternativa, você pode adicionar itens à lista usando a propriedade **List** durante o tempo de criação.

**Observação:** Um evento **Scroll** ocorrerá em um controle **ComboBox** somente quando o conteúdo da parte drop-down do **ComboBox** é rolado, não a cada vez que o conteúdo **ComboBox** for alterado. Por exemplo, se a parte drop-down de um **ComboBox** contém cinco itens e o item na parte superior está selecionado, não ocorrerá um evento **Scroll** até que você pressione a seta para baixo seis vezes, (ou a tecla **PgDn** uma vez). Após isto, ocorre um evento **Scroll** para cada pressionamento da tecla de seta para baixo. Entretanto, se em seguida você pressionar a tecla seta para cima, não ocorrerá um evento **Scroll** até que você pressione a tecla de seta para cima seis vezes (ou a tecla **PgUp** uma vez). Após isto, cada pressionamento da tecla seta para cima resultará em um evento **Scroll**.

## Controle CommandButton

Use um controle **CommandButton** para iniciar, interromper ou finalizar um processo. Ao ser escolhido, um **CommandButton** aparece pressionado e, assim, é por vezes chamado de botão de pressionamento.

#### Sintaxe

#### CommandButton

#### Comentários

Para exibir texto em um controle **CommandButton**, configure sua propriedade **Caption** de maneira adequada. Um usuário pode sempre escolher um **CommandButton** clicando nele. Para permitir ao usuário escolhê-lo pressionando **ENTER**, configure a propriedade **Default** como **True**. Para permitir ao usuário escolher o botão pressionando **ESC**, configure a propriedade **Cancel** do **CommandButton** como **True**.

## Coleção Controls

Uma coleção cujos elementos representam cada controle em um formulário, incluindo os elementos da matriz de controle. A coleção **Controls** tem uma única propriedade **Count**, que especifica o número de elementos em uma matriz.

#### Sintaxe

*object*.**Controls**(*index*)

A sintaxe da coleção **Controls** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto <b>Form</b> .
<i>Index</i>	Um número inteiro com um intervalo de 0 a <code>Controls.Count - 1</code> .

#### Comentários

A coleção **Controls** enumera controles carregados em um formulário e é útil para iterar por eles. A coleção **Controls** identifica uma variável intrínseca ao nível de formulário chamado **Controls**. Se você omitir o espaço reservado opcional *object*, você deve incluir a palavra-chave **Controls**. Entretanto, se você incluir *object*, você pode omitir a palavra-chave **Controls**. Por exemplo, as duas linhas de código abaixo têm o mesmo efeito:

```
MyForm.Controls(6).Top = MyForm.Controls(5).Top + increment  
MyForm(6).Top = MyForm(5).Top + increment
```

Você pode passar **Controls**(*index*) a uma função cujo argumento seja especificado com uma classe **Controls**. Você também pode acessar membros usando seu nome. Por exemplo:

```
Controls("Command1").Top
```

---

## Controle DirListBox

Um controle **DirListBox** exibe diretórios e caminhos durante o tempo de execução. Use este controle para exibir uma lista hierárquica de diretórios. Você pode criar caixas de diálogo que, por exemplo, permitem ao usuário abrir um arquivo de uma lista de arquivos em todos os diretórios disponíveis.

### Sintaxe

#### DirListBox

### Comentários

Configura as propriedades **List**, **ListCount** e **ListIndex** para permitir ao usuário acessar itens em uma lista. Se você também exibe os controles **DriveListBox** e **FileListBox**, pode escrever código para sincronizá-los com o controle **DirListBox** e entre si.

## Controle DriveListBox

Um controle **DriveListBox** permite ao usuário selecionar uma unidade de disco válida durante o tempo de execução. Use este controle para exibir uma lista contendo todas as unidades de disco válidas no sistema de um usuário. Você pode criar caixas de diálogo para permitir ao usuário abrir um arquivo a partir de uma lista de arquivos em um disco em qualquer unidade de disco disponível.

### Sintaxe

#### DriveListBox

### Comentários

Configura as propriedades **List**, **ListCount** e **ListIndex** para permitir a um usuário acessar itens na lista. Se você também exibe os controles **DirListBox** e **FileListBox**, você pode escrever código para sincronizá-los com o controle **DriveListBox** e entre si.

## Controle FileListBox

Um controle **FileListBox** localiza a lista de arquivos no diretório especificado pela propriedade **Path** durante o tempo de execução. Use este controle para exibir uma lista de arquivos selecionados por tipo de arquivo. Você pode criar caixas de diálogo em seu aplicativo que, por exemplo, permitem ao usuário selecionar um arquivo ou grupo de arquivos.

### Sintaxe

#### FileListBox

### Comentários

Configure as propriedades **List**, **ListCount** e **ListIndex** para permitir a um usuário acessar itens na lista. Se também exibir os controles **DirListBox** e **DriveListBox**, você pode escrever código para sincronizá-los com o controle **FileListBox** e entre si.

## Objeto Form, coleção Forms

Um objeto **Form** é uma janela ou caixa de diálogo que compõe parte da interface de usuário de um aplicativo.

Uma coleção Forms é uma coleção cujos elementos representam cada formulário carregado em um aplicativo. A coleção inclui o formulário MDI, formulários MDI filho e formulário não-MDI do aplicativo. A coleção **Forms** tem uma única propriedade, **Count**, que especifica o número de elementos na coleção.

### Sintaxe

#### Form

#### Forms(index)

O espaço reservado *index* representa um número inteiro com um intervalo de 0 a `Forms.Count - 1`.

### Comentários

---

Você pode usar a coleção **Forms** para iterar por todos os formulários carregados em um aplicativo. Ele identifica uma variável intrínseca global chamada **Forms**. Você pode passar **Forms(index)** a uma função cujo argumento seja especificado como uma classe **Forms**.

Os formulários têm propriedades que determinam aspectos de suas aparências como, por exemplo, posição, tamanho e cor; e aspectos de seu comportamento como, por exemplo, se eles são ou não redimensionáveis.

Os formulários também podem responder a eventos iniciados por um usuário ou disparados pelo sistema. Por exemplo, você pode escrever código em um procedimento de evento **Clic** de um formulário que permitiria ao usuário alterar a cor de um formulário ao ser clicado.

Além de propriedades e eventos, você pode usar métodos para manipular formulários usando código. Por exemplo, você pode usar o método **Move** para alterar a localização e tamanho de um formulário.

Um tipo especial de formulário, o formulário MDI, pode conter outras formas chamadas de formulários MDI filho. Um formulário MDI é criado com o comando **MDI Form** no menu **Insert**; um formulário MDI filho é criado selecionando-se **New Form** no menu **File** e, em seguida, configurando a propriedade **MDIChild** como **True**.

Você pode criar múltiplas ocorrências de formulários em código, usando a palavra-chave **New** em instruções **Dim**, **Set** e **Static**.

Ao criar formulários, configure a propriedade **BorderStyle** para definir a borda de um formulário e configure a propriedade **Caption** para colocar texto na barra de título. Em código, você pode usar os métodos **Hide** e **Show** para tornar formulários invisíveis ou visíveis durante o tempo de execução.

**Observação:** Configurar **BorderStyle** como 0 remove a borda. Se você deseja que seu formulário tenha uma borda sem barra de título, caixa do menu **Control**, botão **Maximize** e botão **Minimize**, exclua qualquer texto da propriedade **Caption** do formulário e configure as propriedades **ControlBox**, **MaxButton** e **MinButton** como **False**.

**Form** é um tipo de dados Object. Você pode declarar variáveis como tipo **Form** antes de configurá-las como uma ocorrência de um tipo de formulário que foi declarado durante o tempo de criação. De maneira semelhante, você pode passar um argumento a um procedimento como um tipo **Form**.

Os formulários também podem funcionar como origens em uma conversação DDE, com um controle **Label**, **PictureBox** ou **TextBox** fornecendo os dados.

Você pode acessar a coleção de controles em um **Form** usando a coleção **Controls**. Por exemplo, para ocultar todos os controles em um **Form** você pode usar código semelhante a este:

```
For Each Control in Form1.Controls
    Control.Visible = False
Next Control
```

## Controle Frame

Um controle **Frame** fornece um agrupamento identificável para controles. Você também pode usar um **Frame** para subdividir a funcionalidade de um formulário — por exemplo, para separar grupos de controles **OptionButton**.

### Sintaxe

#### Frame

### Comentários

Para agrupar os controles, primeiro desenhe o controle **Frame** e, em seguida, desenhe os controles dentro de **Frame**. Isto permite mover ao mesmo tempo o **Frame** e os controles nele contidos. Se você desenha um controle fora de **Frame** e, em seguida, tenta movê-lo para fora, o controle estará sobre o **Frame** e você terá que mover o **Frame** e os controles separadamente.

Para selecionar controles múltiplos em um **Frame**, mantenha pressionado a tecla **CTRL**, enquanto usa o mouse para desenhar uma caixa ao redor dos controles.

## Controles HScrollBar, VScrollBar

Barras de rolagem oferecem navegação fácil por uma longa lista de itens ou uma grande quantidade de informações. Eles também podem fornecer uma representação análoga da posição atual. Você pode usar uma barra de rolagem como dispositivo de entrada ou indicador de velocidade ou quantidade — por exemplo, para controlar o volume de um jogo de computador, ou visualizar o

---

tempo decorrido em um processo cronometrado.

### **Sintaxe**

**HScrollBar**

**VScrollBar**

### **Comentários**

Quando você está usando uma barra de rolagem como indicador de quantidade ou velocidade, ou como um dispositivo de entrada, use as propriedades **Max** e **Min** para configurar o intervalo adequado para o controle.

Para especificar a quantidade de alteração a ser reportada em uma barra de rolagem, use a propriedade **LargeChange** para clicar na barra de rolagem e a propriedade **SmallChange** para clicar nas setas, ao final da barra de rolagem. A propriedade **Value** da barra de rolagem aumenta ou diminui de acordo com os valores configurados para as propriedades **LargeChange** e **SmallChange**. Você pode posicionar a caixa de rolagem durante o tempo de execução configurando **Value** entre 0 e 32.767, inclusive.

---

## Controle Image

Use o controle **Image** para exibir um gráfico. Um controle Image pode exibir um elemento gráfico a partir de um bitmap, ícone ou metarquivo, assim como arquivos metarquivo aprimorados, JPEG ou GIF.

### Sintaxe Image

#### Comentários

O controle **Image** usa menos recursos de sistema e regenera-se mais rapidamente que um controle **PictureBox**, mas ele suporta apenas um subconjunto de propriedades, eventos e métodos de **PictureBox**. Use a propriedade **Stretch** para determinar se o gráfico está escalonado para ajustar-se ao controle ou vice-versa. Embora você possa colocar um controle **Image** dentro de um recipiente, um controle **Image** não pode funcionar como recipiente.

## Controle Label

Um controle **Label** é um controle gráfico que você pode usar para exibir texto que o usuário não pode alterar diretamente.

### Sintaxe Label

#### Comentários

Você pode escrever código que altera o texto exibido por um controle **Label** em resposta a eventos durante o tempo de execução. Por exemplo, se o seu aplicativo leva alguns minutos para efetivar uma alteração, você pode exibir uma mensagem do status do processamento em um **Label**. Você também pode usar um **Label** para identificar um controle, por exemplo, um controle **TextBox**, que não tem sua própria propriedade **Caption**.

Configure as propriedades **AutoSize** e **WordWrap**, caso queira que **Label** exiba corretamente linhas de comprimento variável ou número variável de linhas.

Um controle **Label** também pode funcionar como um destino em uma conversação DDE. Configure a propriedade **LinTopic** para estabelecer um vínculo, a propriedade **LinItem** para especificar um item para a conversação, e a propriedade **LinMode** para ativar o vínculo. Quando estas propriedades tiverem sido configuradas, o Visual Basic tenta iniciar a conversação e exibe uma mensagem, caso não consiga.

Configure a propriedade **UseMnemonic** como **True** caso queira definir um caractere na propriedade **Caption** de **Label** como uma tecla de acesso. Ao definir uma tecla de acesso em um controle **Label**, o usuário pode pressionar e manter pressionado ALT+ o caractere que você designou para mover o foco até o próximo controle na ordem de tabulação.

## Controle Line

Um controle **Line** é um controle gráfico exibido como uma linha horizontal, vertical ou diagonal.

### Sintaxe Line

#### Comentários

Você pode usar um controle **Line** durante o tempo de criação para desenhar linhas em formulários. Durante o tempo de execução, você pode usar um controle **Line** ao invés de, ou além do método **Line**. As linhas desenhadas com o controle **Line** permanecem no formulário, mesmo que a configuração da propriedade **AutoRedraw** seja **False**. Os controles **Line** podem ser exibidos em formulários, caixas de figura e em molduras. Você não pode usar o método **Move** para mover um controle **Line** durante o tempo de execução, mas você pode mover ou redimensioná-lo alterando suas propriedades **X1**, **X2**, **Y1** e **Y2**. O efeito de configurar a propriedade **BorderStyle** depende da configuração da propriedade **BorderWidth**. Se **BorderWidth** não for 1 e **BorderStyle** não for 0 ou 6, **BorderStyle** será configurado como 1.

## Controle ListBox

Um controle **ListBox** exibe uma lista de itens onde o usuário pode selecionar um ou mais. Se o número de itens excede o número que pode ser exibido, uma barra de rolagem é adicionada automaticamente ao controle **ListBox**.

Se nenhum item for selecionado, o valor da propriedade **ListIndex** é -1. O primeiro item da lista é **ListIndex** 0 e o valor da propriedade **ListCount** é sempre um a mais que o maior valor de **ListIn-**



---

dex.

## Sintaxe

### ListBox

## Comentários

Para adicionar ou excluir itens em um controle **ListBox**, use o método **AddItem** ou o método **RemoveItem**. Configure as propriedades **List**, **ListCount** e **ListIndex** para permitir que um usuário acesse itens na **ListBox**. Como alternativa, você pode adicionar itens à lista usando a propriedade **List** durante o tempo de criação.

## Objeto MDIForm

Um formulário MDI (interface de documento múltiplo) é uma janela que funciona como segundo plano de um aplicativo e é o recipiente de formulários que tenham suas propriedades **MDIChild** configuradas como **True**.

## Sintaxe

### MDIForm

## Comentários

Você cria um objeto **MDIForm** escolhendo **MDI Form** no menu **Insert**.

Um aplicativo somente pode ter um objeto **MDIForm**, mas muitos formulários MDI filho. Se um formulário MDI filho tem menus, a barra de menu do formulário filho substitui automaticamente a barra de menu do objeto **MDIForm** quando o formulário MDI filho está ativo. Um formulário MDI filho minimizado é exibido como um ícone dentro do **MDIForm**.

Um objeto **MDIForm** pode conter apenas controles **Menu** e **PictureBox** e controles personalizados que tenham uma propriedade **Align**. Para colocar outros controles em um **MDIForm**, você pode desenhar uma caixa de figura em um formulário e, em seguida, desenhar outros controles dentro da caixa de imagem. Você pode usar o método **Print** para exibir texto em uma caixa de figura em um **MDIForm**, mas você não pode usar este método para exibir texto no próprio **MDIForm**.

Um objeto **MDIForm** não pode ter janela restrita.

Os formulários MDI filho são criados independentemente do **MDIForm**, mas estão sempre contidos no **MDIForm** durante o tempo de execução.

Você pode acessar a coleção de controles em um **MDIForm** usando a coleção **Controls**. Por exemplo, para ocultar os controles em um MDIForm você pode usar código semelhante a este:

```
For Each Control in MDIForm1.Controls  
    Control.Visible = False  
Next Control
```

A propriedade **Count** do **MDIForm** informa o número de controles na coleção **Controls**

---

## Controle Menu

Um controle **Menu** exibe um menu personalizado para seu aplicativo. Um menu pode incluir comandos, submenus e barras separadoras. Cada menu que você cria pode ter até quatro níveis de submenus.

### Sintaxe Menu

#### Comentários

Para criar um controle **Menu**, use o Menu Editor. Digite o nome do controle **Menu** na caixa **Caption**. Para criar uma barra separadora, digite um único hífen (-) na caixa **Caption**. Para exibir uma marca de seleção à esquerda de um item de menu, selecione a caixa **Checked**.

Embora você possa configurar algumas propriedades do controle **Menu** usando o Menu Editor, todas as propriedades do controle **Menu** são exibidas na janela **Properties**. Para exibir as propriedades de um controle **Menu**, selecione o nome do menu na lista **Objects** na parte superior da janela **Properties**.

Quando você cria um aplicativo MDI, a barra de menu no formulário MDI filho substitui a barra de menu no objeto **MDIForm** quando o formulário filho está ativo.

## Controle OptionButton

Um controle **OptionButton** exibe uma opção que pode ser ligada ou desligada.

### Sintaxe OptionButton

#### Comentários

Normalmente, os controles **OptionButton** são usados em um grupo de opções para exibir opções entre as quais pode-se selecionar uma. Você agrupa os controles **OptionButton** desenhando-os em recipientes como, por exemplo, um controle **Frame**, um controle **PictureBox** ou um formulário. Para agrupar controles **OptionButton** em um **Frame** ou **PictureBox**, desenhe primeiro o **Frame** ou **PictureBox** e, em seguida, desenhe o **OptionButton** dentro dele. Todos os controles **OptionButton** dentro do mesmo recipiente funcionam como um único grupo.

Embora os controles **OptionButton** e **CheckBox** possam parecer ter um funcionamento semelhante, existe uma diferença importante: quando um usuário seleciona um **OptionButton**, os outros controles **OptionButton** no mesmo grupo ficam automaticamente indisponíveis. Ao contrário, pode-se selecionar qualquer número de controles **CheckBox**.

## Controle PictureBox

Um controle **PictureBox** pode exibir um elemento gráfico de um arquivo de bitmap, ícone ou metarquivo, assim como de metarquivos aprimorados, arquivos JPEG ou GIF. Ele recorta o elemento gráfico se o controle não for grande o suficiente para exibir a imagem inteira.

### Sintaxe PictureBox

#### Comentários

Você também pode usar um controle **PictureBox** para agrupar controles **OptionButton** e para exibir saída de exibição de métodos gráficos e texto escrito com o método **Print**.

Para fazer com que um controle **PictureBox** redimensione-se automaticamente para exibir um elemento gráfico inteiro, configure sua propriedade **AutoSize** como **True**.

Para criar animação ou simulação, você pode manipular propriedades e métodos gráficos em código. Propriedades e eventos gráficos são úteis para operações de impressão durante o tempo de execução como, por exemplo, modificar o formato de um formulário na tela, para impressão.

Um controle **PictureBox** também pode funcionar como um vínculo de destino em uma conversação DDE.

Os controles **PictureBox** e **Data** são os únicos controles padrão do Visual Basic que você pode colocar na área interna de um formulário MDI. Você pode usá-lo para controlar grupos na parte superior ou inferior da área interna para criar uma barra de ferramentas ou barra de status.

## Controle Shape

```
{ewc HLP95EN.DLL,DYNALIN,"Consulte também":"vbobjShapeC;vbproBoosOnlineJumpTopic"} {ewc  
HLP95EN.DLL,DYNALIN,"Exemplo":"vbobjShapeX":1} {ewc HLP95EN.DLL,DYNALIN,"Propriedades":"vbobjShapeP"}  
{ewc HLP95EN.DLL,DYNALIN,"Métodos":"vbobjShapeM"} {ewc HLP95EN.DLL,DYNALIN,"Eventos":"vbobjShapeE"}
```

Um controle **Shape** é um controle gráfico exibido como retângulo, quadrado, oval, círculo, retângulo arredondado ou quadrado arredondado.

### Sintaxe

#### Shape

#### Comentários

Use controles **Shape** durante o tempo de criação ao invés de, ou além de acionar métodos **Circle** e **Line** durante o tempo de execução. Você pode desenhar um controle **Shape** em um recipiente, mas ele não pode funcionar como um recipiente. O efeito da configuração da propriedade **BorderStyle** depende da configuração da propriedade **BorderWidth**. Se **BorderWidth** não for 1 e **BorderStyle** não for 0 ou 6, **BorderStyle** é configurada como 1.

## Controle TextBox

Um controle **TextBox**, chamado algumas vezes de campo de edição ou controle de edição, exibe as informações inseridas durante o tempo de criação, digitadas pelo usuário ou atribuídas ao controle em código durante o tempo de execução.

### Sintaxe

#### TextBox

#### Comentários

Para exibir linhas múltiplas de texto em um controle **TextBox**, configure a propriedade **MultiLine** como **True**. Se um **TextBox** de linhas múltiplas não tem uma barra de rolagem horizontal, o texto muda automaticamente de linha, mesmo quando o **TextBox** for redimensionado. Para personalizar a combinação de barra de rolagem, em um **TextBox**, configure a propriedade **ScrollBars**. As barras de rolagem sempre aparecerão no **TextBox** quando sua propriedade **MultiLine** for configurada como **True**, e sua propriedade **ScrollBars** for configurada como qualquer coisa, exceto **None** (0).

Se você configura a propriedade **MultiLine** como **True**, pode usar a propriedade **Alignment** para configurar o alinhamento do texto dentro do **TextBox**. O texto é justificado à esquerda como padrão. Se a propriedade **MultiLine** for **False**, a configuração da propriedade **Alignment** não tem qualquer efeito.

Um controle **TextBox** também pode funcionar como um vínculo de destino em uma conversaç o DDE.

## Controle Timer

Um controle **Timer** pode executar c digo a intervalos regulares provocando a ocorr ncia de um evento **Timer**.

### Sintaxe

#### Timer

#### Coment rios

O controle **Timer**, invis vel para o usu rio,    til para processamento em segundo plano.

Voc  n o pode configurar a propriedade **Enabled** de um **Timer** como uma sele  o m ltipla de controles diferentes de controles **Timer**.

Praticamente n o existe limite ao n mero de controles de cron metro ativos que voc  pode ter em Visual Basic 5.0 executado sob o Windows 95 ou o Windows NT.

## Exemplo de cole  o Forms

Este exemplo preenche uma caixa de listagem com as legendas de todos os formul rios atualmente carregados.

```
Private Sub Form_Activate ()  
    Dim I ' Declarar vari vel.  
    ' Preencher novamente a lista (no caso de uma ocorr ncia ter sido  
    adicionada ou removida).  
    lstForms.Clear ' Limpar a caixa de listagem.
```

```

For I = 0 To Forms.Count - 1
    lstForms.AddItem Forms(I).Caption
Next I
End Sub

```

## Exemplo de coleção Controls

Este exemplo ativa todos os controles atualmente carregados em um formulário (exceto menus).

```

Sub EnableControlsOn (Frm As Form, State As Integer)
    Dim I ' Declarar variável.
    For I = 0 To Frm.Controls.Count - 1
        If Not TypeOf Frm.Controls(I) Is Menu Then
            Frm.Controls(I).Enabled = State
        End If
    Next I
End Sub

```

## Controle OLE Container

O controle OLE Container permite adicionar objetos inseríveis aos formulários no seu aplicativo em Visual Basic. Com o controle OLE Container, você pode:

- Criar um espaço reservado no seu aplicativo para um objeto inserível. Durante o tempo de execução você pode criar o objeto que é exibido dentro do OLE Container ou alterar um objeto que você colocou no controle OLE Container durante o tempo de criação.
- Criar um objeto vinculado em seu aplicativo.
- Acople o controle OLE Container a um banco de dados usando o controle **Data**.

Você cria o objeto durante o tempo de criação usando a caixa de diálogo **Insert Object** (que contém os comandos **Insert Object**, **Paste Special** e assim por diante) ou durante o tempo de execução configurando as propriedades adequadas.

Quando você move um controle OLE Container em um formulário usando o método **ObjectMove**, os valores das propriedades **Height** e **Width** do objeto podem ser ligeiramente diferentes após movido. Isto ocorre porque os parâmetros do método **ObjectMove** são valores de pixel convertidos para o modo de escala do formulário atual. A conversão de pixels em twips e de volta nem sempre resulta em valores idênticos.

### Usando menus pop-up de controle OLE Container

Toda vez que você desenha um controle OLE Container em um formulário, é exibida a caixa de diálogo **Insert Object**. Use esta caixa de diálogo para criar um objeto vinculado ou incorporado. Se você escolher **Cancel**, nenhum objeto será criado.

Durante o tempo de criação, clique no controle OLE Container com o botão direito do mouse para exibir um menu pop-up. O comando exibido neste menu pop-up depende do estado do OLE Container conforme mostrado na tabela abaixo:

Comando	Ativado no menu pop-up quando
Insert Object	Sempre ativado.
Paste Special	O objeto <b>Clipboard</b> contém um objeto válido.
Delete Embedded Object	O controle OLE Container contém um objeto incorporado.
Delete Lined Object	O controle OLE Container contém um objeto vinculado.
Create Lin	A propriedade <b>SourceDoc</b> está configurada.
Create Embedded Object	A propriedade <b>Class</b> ou <b>SourceDoc</b> está configurada.

Um controle OLE Container pode conter apenas um objeto de cada vez. Você pode criar um objeto incorporado ou vinculado de diversas maneiras:

- Use as caixas de diálogo **Insert Object** ou **Paste Special** (tempo de execução ou tempo de criação).
- Configure a propriedade **Class** na janela **Properties**, clique no controle OLE Container com o botão direito do mouse e selecione o comando adequado (somente tempo de criação).
- Use o método adequado do controle OLE Container.

### Localizando classes de nome

Você pode obter uma lista de nomes de classe disponíveis para o seu aplicativo selecionando a propriedade **Class** na janela **Properties** e clicando no botão **Properties**.

**Observação:** A caixa de diálogo **Insert Object** não exibe uma lista de nomes de classe. Esta

---

caixa de diálogo exibe nomes amigáveis para cada classe de objetos, que geralmente são mais longos e mais facilmente compreendidos.

## Propriedade Class

Retorna ou configura o nome de classe de um objeto incorporado.

### Sintaxe

*object.Class* [ = *string*]

A sintaxe da propriedade **Class** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>string</i>	Uma <u>expressão de sequência de caracteres</u> especificando o nome de classe.

### Comentários

Um nome de classe define o tipo de um objeto. Os aplicativos que suportam componentes ActiveX qualificam totalmente os nomes de classe de seus objetos usando qualquer uma das sintaxes abaixo:

*application.objecttype.version*

*objecttype.version*

A sintaxe de nome de classe de componente ActiveX tem as seguintes partes:

Parte	Descrição
<i>application</i>	O nome do aplicativo que fornece o objeto.
<i>objecttype</i>	O nome do objeto conforme definido na biblioteca de objetos.
<i>version</i>	O número de versão do objeto ou aplicativo que fornece o objeto.

Por exemplo, O Microsoft Excel versão 5.0 suporta um grande número de objetos, incluindo planilhas e gráficos. Seus nomes de classe são **Excel.Sheet.5** e **Excel.Chart.5**. O Microsoft WordArt versão 2.0 suporta um único objeto com o nome de classe **MSWordArt.2**.

**Observação:** Alguns documentação de programação de componente ActiveX refere-se à sintaxe de nome de classe como uma identificação programática.

Para visualizar uma lista de nomes de classe disponíveis em seu sistema, selecione o controle OLE Container, selecione a propriedade **Class** na janela **Properties** e clique no botão compilador.

Copiar um objeto da **Área de transferência** do sistema atualiza a propriedade **Class** do controle. Por exemplo, se você colar um gráfico do Microsoft Excel da **Área de transferência** para um controle OLE Container que anteriormente continha uma planilha do Microsoft Excel, sua configuração de propriedade **Class** se altera de **Excel.Sheet.5** para **Excel.Chart.5**. Você pode colar um objeto da **Área de transferência** do sistema durante o tempo de execução com o método **Paste** ou o método **PasteSpecialDlg**.

## Propriedade ObjectVerbFlags

Retorna o estado do menu (por exemplo, ativado ou desativado, selecionado e assim por diante) para cada verbo em uma determinada matriz **ObjectVerbs**.

### Sintaxe

*object.ObjectVerbFlags*(*number*)

A sintaxe da propriedade **ObjectVerbFlags** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Uma <u>expressão numérica</u> indicando o elemento na matriz.

### Valores de retorno

---

A propriedade **ObjectVerbFlags** retorna os seguintes valores:

Constante	Valor	Descrição
<b>vbOLEFlagChecked</b>	&H0008	O item de menu está selecionado.
<b>vbOLEFlagDisabled</b>	&H0002	O item de menu está desativado (mas não esmaecido).
<b>vbOLEFlagEnabled</b>	&H0000	O item de menu está ativado.
<b>vbOLEFlagGrayed</b>	&H0001	O item de menu está esmaecido.
<b>vbOLEFlagSeparator</b>	&H0800	O item de menu é uma barra separadora.

**Observação:** Estas constantes também estão listadas na biblioteca de procedimentos e objetos do Visual Basic no Object Browser.

#### **Comentários**

O primeiro verbo na matriz **ObjectVerbs** é o verbo padrão. A matriz **ObjectVerbFlags** contém informações sobre o estado do menu (por exemplo, esmaecido, selecionado e assim por diante) para cada verbo na matriz **ObjectVerbs**.

Ao exibir um menu contendo os verbos de um objeto, verifique o valor desta propriedade para ver como o item está configurado para ser exibido.

---

## Propriedade SizeMode

Retorna ou configura um valor especificando como o controle OLE Container é dimensionado ou como sua imagem é exibida quando ele contém um objeto.

### Sintaxe

*object.SizeMode* [ = *value*]

A sintaxe da propriedade **SizeMode** tem estas partes:

Parte	Descrição
<i>object</i>	Uma expressão de <u>objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Um número inteiro ou constante especificando como o controle é dimensionado e como sua imagem é exibida, conforme descrito em Configurações.

### Configurações

As configurações de *value* são:

Constante	Valor	Descrição
<b>vbOLESizeClip</b>	0	(Padrão) Clip. O objeto é exibido em tamanho real. Se o objeto for maior que o controle OLE Container, sua imagem será recortada pelas bordas do controle.
<b>vbOLESizeStretch</b>	1	Stretch. A imagem do objeto é dimensionada para preencher o controle OLE Container. A imagem pode não manter as proporções originais do objeto.
<b>vbOLESizeAutoSize</b>	2	Autosize. O controle OLE Container é redimensionado para exibir o objeto número inteiro.
<b>vbOLESizeZoom</b>	3	Zoom. O objeto é redimensionado para preencher o controle OLE Container tanto quanto possível ao mesmo tempo que mantém as proporções originais do objeto.

### Comentários

Quando **SizeMode** for configurado como 2 (Autosize), o controle OLE Container será automaticamente redimensionado quando o tamanho da exibição de um objeto se alterar. Quando isto ocorrer, o evento *Resize* será acionado antes que o controle OLE Container seja automaticamente redimensionado. Os argumentos *heightnew* e *widthnew* no procedimento de evento *Resize* indicam o tamanho ideal para exibição do objeto (este tamanho é determinado pelo aplicativo que criou o objeto). Você pode dimensionar o controle alterando os valores dos argumentos *heightnew* e *widthnew* no procedimento de evento *Resize*.

---

## Propriedade Data

Retorna ou configura um identificador como um objeto de memória ou objeto de interface de dispositivo gráfico (GDI) contendo dados em um formato específico. Não disponível durante o tempo de criação.

### Sintaxe

*object.Data* [ = *number*]

A sintaxe da propriedade **Data** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Um número inteiro Long especificando um identificador.

### Comentários

Configura esta propriedade para enviar dados a um aplicativo que criou um objeto. Antes de usar a propriedade **Data**, configure a propriedade **Format** para especificar o tipo de dados contido no objeto de memória ou objeto GDI.

Você pode obter uma lista de formatos aceitáveis para um objeto usando as propriedades **ObjectAcceptFormats** e **ObjectGetFormats**.

Configurar esta propriedade como 0 libera a memória associada ao identificador.

**Dica** Automation oferece uma solução mais fácil e mais confiável para enviar dados e comandos de um objeto. Se um objeto suportar automação, você pode acessar o objeto através da propriedade **Object** ou usando as funções **CreateObject** e **GetObject**.

## Propriedade DataText

Retorna uma seqüência de caracteres do objeto especificado ou configura uma seqüência de caracteres para este.

### Sintaxe

*object.DataText* [ = *string*]

A sintaxe da propriedade **DataText** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>string</i>	Uma <u>expressão de seqüência de caracteres</u> especificando a seqüência de caracteres.

### Comentários

Para enviar uma seqüência de caracteres a um objeto, configure primeiro a propriedade **Format** como um formato suportado pelo objeto. Use as propriedades **ObjectGetFormats** e **ObjectAcceptFormats** para obter uma lista de formato suportada por um objeto.

Ao obter dados de um objeto, a propriedade **DataText** retorna a seqüência de caracteres do objeto, terminando no primeiro caractere nulo.

A seqüência de caracteres **DataText** pode ter o tamanho que a memória disponível permitir.

**Dica** Automação oferece uma solução mais fácil e mais confiável para enviar dados e comandos de e para um objeto. Se um objeto suportar Automation, você pode acessá-lo através da propriedade **Object** ou usando as funções **CreateObject** e **GetObject**.



---

## Propriedade FileName

Retorna ou configura o número do arquivo a ser usado ao salvar ou carregar um objeto ou retorna o último número de arquivo usado. Não disponível durante o tempo de criação.

**Observação:** A propriedade **FileName** é incluída para compatibilidade com a propriedade **Action** em versões anteriores. Para a funcionalidade atual, use os métodos **SaveToFile** e **ReadFromFile**.

### Sintaxe

*object.FileName* [ = *number*]

A sintaxe da propriedade **FileName** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Uma <u>expressão numérica</u> especificando o número do arquivo.

### Comentários

O número de arquivo deve corresponder a um arquivo binário aberto.

Você pode usar esta propriedade para especificar o número do arquivo a ser aberto com o método **ReadFromFile** ou salvo com os métodos **SaveToFile** ou **SaveToOle1File**.

## Propriedade HostName

Retorna ou configura o nome de hospedeiro legível pelo usuário de seu aplicativo em Visual Basic.

### Sintaxe

*object.HostName* [ = *name*]

A sintaxe da propriedade **HostName** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>name</i>	Uma <u>expressão de sequência de caracteres</u> especificando o nome do hospedeiro.

### Comentários

Ao editar um objeto, a configuração da propriedade **HostName** pode ser exibida no título da janela do objeto. Entretanto, alguns aplicativos que fornecem objetos não exibem **HostName**.

## Propriedade IpOleObject

Retorna o endereço do objeto.

### Sintaxe

*object.IpOleObject*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Muitas chamadas de função nas DLLs ActiveX exigem o endereço de um objeto como argumento. Passe o valor especificado na propriedade **IpOleObject** ao efetuar chamadas API a DLLs ActiveX. O valor é 0 se nenhum objeto estiver atualmente exibido. Se uma chamada for feita a uma API que efetue uma chamada de retorno ao controle OLE Container, o resultado é imprevisível. O endereço retornado por esta propriedade é um ponteiro para a interface **IpOleObject** para o objeto ativo.

## Propriedade ObjectAcceptFormatsCount

Retorna o número de formatos que podem ser aceitos por um objeto.

---

### Sintaxe

*object*.**ObjectAcceptFormatsCount**

O *object* é uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Use esta propriedade para obter o número de elementos na matriz de propriedade **ObjectAcceptFormats**.

## Propriedade DisplayType

Retorna ou configura um valor indicando se um objeto exibir seu conteúdo ou um ícone.

### Sintaxe

*object*.**DisplayType** [ = *value*]

A sintaxe da propriedade **DisplayType** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Um número inteiro ou constante especificando se um objeto exibe o conteúdo ou um ícone, conforme descrito em Configurações.

### Configurações

As configurações de *value* são:

Constante	Valor	Descrição
<b>vbOLEDisplayContent</b>	0	(Padrão) Content. Quando o controle OLE Container contém um objeto, os dados do objeto são exibidos no controle.
<b>vbOLEDisplayIcon</b>	1	Icon. Quando o controle OLE Container contém um objeto, o ícone do objeto é exibido no controle.

### Comentários

Esta propriedade determina a configuração padrão da caixa de seleção **Display As Icon** As Icon nas caixas de diálogo **Insert Object** e **Paste Special**. Quando você exibir estas caixas de diálogo durante o tempo de execução (com os métodos **InsertObjDlg** ou **PasteSpecialDlg**) ou durante o tempo de criação, a caixa de seleção **Display As Icon** será automaticamente selecionada se esta propriedade for configurada como 1 (Icon).

Ao criar um objeto durante o tempo de execução usando os métodos **CreateEmbed** ou **CreateLink**, use a propriedade **DisplayType** para determinar se o objeto será exibido como um ícone (configurar **DisplayType** = 1) ou se os dados do objeto forem exibidos no controle (configure **DisplayType** = 0).

Uma vez criado um objeto, você não pode alterar o tipo de exibição.

## Propriedade ObjectGetFormats

Retorna a lista de formatos que podem ser fornecidos por um objeto.

### Sintaxe

*object*.**ObjectGetFormats**( *number* )

A sintaxe da propriedade **ObjectGetFormats** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Uma <u>expressão numérica</u> indicando o elemento na matriz.

### Comentários

A lista é uma matriz de sequência de caracteres de base zero. Elementos da matriz podem ser usados para configurar a propriedade **Format** ao obter dados de um objeto usando as proprieda-

## Propriedade **ObjectGetFormatsCount**

Retorna o número de formatos que um objeto pode fornecer.

### Sintaxe

*object*.**ObjectGetFormatsCount**

O *object* é uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Use esta propriedade para determinar o número de elementos na matriz de propriedades **ObjectGetFormats**.

## Propriedade **OLEType**

Retorna o status do objeto em um controle OLE Container.

### Sintaxe

*object*.**OLEType**

O *object* é uma expressão de objeto que avalia para um objeto na lista Applies To.

### Valores de retorno

A propriedade **OLEType** retorna os seguintes valores:

Constante	Valor	Descrição
<b>vbOLELined</b>	0	Vinculado. O controle OLE Container contém um <u>objeto vinculado</u> . Todos os dados do objeto são gerenciados pelo aplicativo que o criou. Quando o objeto for salvo usando o método <b>SaveToFile</b> , somente informações de vínculo como, por exemplo <b>SourceDoc</b> , <b>Sourceltem</b> e assim por diante, são salvas no arquivo especificado por seu aplicativo em Visual Basic.
<b>vbOLEEmbedded</b>	1	Incorporado. O controle OLE Container contém um <u>objeto incorporado</u> . Todos os dados do objeto são gerenciados dentro do aplicativo em Visual Basic. Quando o objeto é salvo usando o método <b>SaveToFile</b> , todos os dados associados ao objeto são salvos no arquivo especificado.
<b>vbOLENone</b>	3	Nenhum. O controle OLE Container não contém um objeto.

### Comentários

Use esta propriedade para determinar se o controle OLE Container contém um objeto ou determina o tipo de objeto contido no controle OLE Container.

Use a propriedade **ApplsRunning** para determinar se o aplicativo que criou o objeto está sendo executado.

Ao criar um objeto, use a propriedade **OLETypeAllowed** para determinar o tipo de objeto que pode ser criado.

## Propriedade **ObjectVerbs**

Retorna a lista de verbos suportada por um objeto.

### Sintaxe

*object*.**ObjectVerbs**(*number*)

A sintaxe da propriedade **ObjectVerbs** tem estas partes:

Parte	Descrição
-------	-----------

<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Uma <u>expressão numérica</u> indicando o elemento na matriz.

### Comentários

**ObjectVerbs** é uma matriz de seqüência de caracteres de base zero. Use esta propriedade junto com a propriedade **ObjectVerbsCount** para obter os verbos suportados por um objeto. Estes verbos são usados para determinar uma ação a ser executada quando um objeto for ativado com o método **DoVerb**. A lista de verbos na matriz varia de objeto para objeto e depende das condições atuais.

Cada objeto pode suportar seu próprio conjunto de verbos. Os valores abaixo representam verbos padrão suportados por cada objeto:

Constante	Valor	Descrição
<b>vbOLEPrimary</b>	0	A ação padrão para o objeto.
<b>vbOLEShow</b>	-1	Ativa o objeto para edição. Se o aplicativo que criou o objeto suportar <u>ativação in-loco</u> , o objeto será ativado dentro do controle OLE Container.
<b>vbOLEOpen</b>	-2	Abre o objeto em uma janela de aplicativo separada. Se o aplicativo que criou o objeto suportar a ativação in-loco, o objeto será ativado em sua própria janela.
<b>vbOLEHide</b>	-3	Para objetos incorporados, oculta o aplicativo que criou o objeto.
<b>vbOLEUIInPlaceUIActivate</b>	-4	Se o objeto suportar ativação in-loco, ativa o objeto para ativação in-loco e mostra qualquer ferramenta de interface de usuário. Se o objeto não suportar ativação in-loco, o objeto não será ativado e ocorrerá um erro.
<b>vbOLEInPlaceActivate</b>	-5	Se o usuário move o foco para o controle OLE Container, cria uma janela e prepara o objeto para ser editado. Ocorrerá um erro se o objeto não suportar ativação com um único clique de mouse.
<b>vbOLEDiscardUndoState</b>	-6	Usado quando o objeto é ativado para edição para descartar todos os registros de alterações que o aplicativo do objeto pode desfazer.

**Observação:** Estes verbos podem ser listados na matriz de propriedades **ObjectVerbs**.

O primeiro verbo na matriz **ObjectVerbs**, **ObjectVerbs(0)**, é o verbo padrão. A menos que seja especificado de maneira diferente, este verbo ativa o objeto.

Os verbos restantes na matriz podem ser exibidos em um menu. Se for adequado exibir o verbo padrão em um menu, o verbo padrão tem duas entradas na matriz **ObjectVerbs**.

Os aplicativos que exibem objetos geralmente incluem um comando **Object** no menu **Edit**. Quando o usuário escolhe **Edit Object**, um menu exibe os verbos do objeto. Use as propriedades **ObjectVerbs**, **ObjectVerbsCount** e **ObjectVerbFlags** para criar este tipo de menu durante o tempo de execução.

A lista de verbos suportados por um objeto pode variar, dependendo do estado do objeto. Para atualizar a lista de verbos suportada por um objeto, use o método **FetchVerbs**. Certifique-se de atualizar a lista de verbos antes de apresentá-la ao usuário.

Para exibir automaticamente os verbos na matriz **ObjectVerbs** em um menu pop-up quando o usuário clica em um objeto com o botão direito do mouse, configure a propriedade **AutoVerbMenu** como **True**.

---

## Propriedade ObjectVerbsCount

Retorna o número de verbos suportados por um objeto.

### Sintaxe

*object*.**ObjectVerbsCount**

O *object* é uma expressão de objeto que é avaliada como um controle OLE Container.

### Comentários

Use esta propriedade para determinar o número de elementos na matriz de propriedades **ObjectVerbs**.

A lista de verbos suportada por um objeto pode variar dependendo do estado do objeto. Para atualizar a lista de verbos suportada por um objeto, use o método **FetchVerbs**.

---

## Propriedade SourceDoc

Retorna ou configura o nome de arquivo a ser usado quando você cria um objeto.

**Observação:** Você configura a propriedade **SourceDoc** para compatibilidade com a propriedade de **Action** em versões anteriores. Para funcionalidade atual, use os métodos **CreateEmbed** e **CreateLink**.

### Sintaxe

*object.SourceDoc* [ = *name*]

A sintaxe da propriedade **SourceDoc** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>name</i>	Uma <u>expressão de sequência de caracteres</u> especificando um nome de arquivo.

### Comentários

Use a propriedade **SourceDoc** para especificar o arquivo a ser vinculado ao criar um objeto vinculado usando a propriedade **Action**. Use a propriedade **Sourceltem** para especificar dados dentro de um arquivo a ser vinculado.

Ao criar um objeto incorporado usando a propriedade **Action**, se a propriedade **SourceDoc** estiver configurada como um nome de arquivo válido, um objeto incorporado será criado usando o arquivo especificado como modelo.

Quando um objeto vinculado for criado, a propriedade **Sourceltem** será concatenada à propriedade **SourceDoc**. Durante o tempo de execução, a propriedade **Sourceltem** retorna uma sequência de caracteres de comprimento zero (""), e a propriedade **SourceDoc** retorna o caminho completo até o arquivo vinculado, seguido de um ponto de exclamação (!) ou barra invertida (\), seguido do **Sourceltem**. Por exemplo:

"C:\WOR\QTR1\REVENUE.XLS!R1C1:R30C15"

## Propriedade Sourceltem

Retorna ou configura os dados no arquivo a ser vinculado quando você cria um objeto vinculado.

### Sintaxe

*object.Sourceltem* [ = *string*]

A sintaxe da propriedade **Sourceltem** tem estas partes:

parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>string</i>	Uma <u>expressão de sequência de caracteres</u> especificando os dados a serem vinculados.

### Comentários

**OLETypeAllowed** deve ser configurado como 0 (Lined) ou 2 (Either) ao usar esta propriedade.

Use a propriedade **SourceDoc** para especificar o arquivo a ser vinculado.

Cada objeto usa sua própria sintaxe para descrever unidades de dados. Para configurar esta propriedade, especifique uma unidade de dados reconhecida pelo objeto. Por exemplo, ao vincular ao Microsoft Excel, especifique o **Sourceltem** usando uma referência de célula ou intervalo de células como, por exemplo, R1C1 ou R3C4:R9C22, ou um intervalo nomeado, por exemplo, Receitas.

Para determinar a sintaxe para descrever uma unidade de dados para um objeto, consulte a documentação do aplicativo que criou o objeto.

**Observação:** Você pode determinar esta sintaxe criando um objeto vinculado durante o tempo de criação usando o comando **Paste Special** (clique no controle OLE Container com o botão direito do mouse). Uma vez criado o objeto, selecione a propriedade **SourceDoc** na janela **Properties** e examine a sequência de caracteres na caixa **Configurações**. Para a maioria dos objetos, esta sequência de caracteres contém um caminho até o arquivo vinculado, seguido de um ponto de exclamação (!) ou barra invertida (\), e a sintaxe dos dados vinculados.

Quando um objeto vinculado é criado, a propriedade **Sourceltem** é concatenada à propriedade **SourceDoc**. Durante o tempo de execução, a propriedade **Sourceltem** retorna uma sequência de caracteres de comprimento zero (""), e a propriedade **SourceDoc** retorna o caminho completo até

---

o arquivo vinculado, seguido de um ponto de exclamação (!) ou barra invertida (\), seguido do **SourceItem**. Por exemplo:  
"C:\WOR\QTR1\REVENUE.XLS!R1C1:R30C15"

## Propriedade ApplsRunning

Retorna ou configura um valor que indica se o aplicativo que criou o objeto no controle OLE Container está sendo executado. Não disponível durante o tempo de criação.

### Sintaxe

*object*.ApplsRunning [= *boolean*]

A sintaxe da propriedade **ApplsRunning** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Um <u>expressão Booleana</u> especificando se o aplicativo que produziu o objeto no controle OLE Container está ou não em execução, conforme descrito em Configurações.

### Configurações

As configurações de *boolean* são:

Configuração	Descrição
<b>True</b>	O aplicativo que produziu o objeto no controle OLE Container está sendo executado.
<b>False</b>	O aplicativo que produziu o objeto no controle OLE Container não está sendo executado.

### Comentários

Você pode configurar o valor da propriedade **ApplsRunning** para iniciar o aplicativo que produz o objeto no controle OLE Container. Fazer isto provoca uma ativação mais rápida do objeto. Você também pode configurar esta propriedade como **False** para fechar o aplicativo quando o objeto perde o foco.

## Propriedade UpdateOptions

Retorna ou configura um valor especificando como um objeto é atualizado quando dados vinculados são modificados.

### Sintaxe

*object*.UpdateOptions [= *number*]

A sintaxe da propriedade **UpdateOptions** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Um número inteiro especificando como um objeto é atualizada, conforme descrito em Configurações.

### Configurações

As configurações de *number* são:

Constante	Valor	Descrição
<b>vbOLEAutomatic</b>	0	(Padrão) Automatic. O objeto é atualizado toda vez que os dados vinculados são alterados.
<b>vbOLEFrozen</b>	1	Frozen. O objeto é atualizado sempre que o usuário salva os dados vinculados a partir do aplicativo em que foram criados.
<b>vbOLEManual</b>	2	Manual. O objeto somente é atualizado usando o método <b>Update</b> .

---

## Comentários

Esta propriedade é útil para objetos vinculados em que outros usuários ou aplicativos podem acessar e modificar os dados vinculados.

Quando os dados de um objeto forem alterados, o evento Updates será acionado.

## Propriedade Verb

Retorna ou configura um valor especificando uma operação a ser executada quando um objeto é ativado usando a propriedade **Action**.

**Observação:** A propriedade **Verb** é incluída para compatibilidade com a propriedade **Action** em versões anteriores. Para funcionalidade atual, use o método **DoVerb**.

### Sintaxe

*object.Verb [= number]*

A sintaxe da propriedade **Verb** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Um valor que especifica a operação a ser executada.

### Comentários

Cada objeto pode suportar seu próprio conjunto de verbos. Use as propriedades **ObjectVerbs** e **ObjectVerbsCount** para acessar a lista de verbos suportadas por um objeto. Configure **Verb** = 1 para especificar o primeiro verbo da lista, configure **Verb** = 2 para especificar o segundo verbo da lista e assim por diante.

Configure **AutoActivate** como 2 (Double-Clic) para ativar automaticamente um objeto quando ele é clicado duas vezes pelo usuário.

Configure **AutoVerbMenu** = **True** para exibir um menu pop-up contendo os verbos do objeto quando o usuário clicar no objeto com o botão direito do mouse.



---

## Evento Updated

Ocorre quando os dados de um objeto forem modificados.

### Sintaxe

**Sub** *object\_Updated* (*code* **As Integer**)

O evento **Updated** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>code</i>	Um número inteiro que especifica como o objeto foi atualizado, conforme descrito em Configurações.

### Configurações

As configurações de *code* são:

Constante	Valor	Descrição
<b>vbOLEChanged</b>	0	Os dados do objeto foram alterados.
<b>vbOLESaved</b>	1	Os dados do objeto foram salvos pelo aplicativo que criou o objeto.
<b>vbOLEClosed</b>	2	O arquivo contendo os dados do objeto vinculado foram fechados pelo aplicativo que criou o objeto.
<b>vbOLERenamed</b>	3	O arquivo contendo os dados do objeto vinculado foi renomeado pelo aplicativo que criou o objeto.

### Comentários

Estas constantes estão listadas na biblioteca de objetos do Visual Basic (VB) no Object Browser. Você pode usar este evento para determinar se os dados de um objeto foram alterados desde a última vez em que eles foram salvos. Para fazer isto, configure a variável global no evento Updates indicando que o objeto precisa ser salvo. Após salvar o objeto, reconfigure a variável.

## Propriedade AutoActivate

Retorna ou configura um valor que ativa um objeto clicando duas vezes no controle OLE Container ou movendo o foco para o controle OLE Container.

### Sintaxe

*object*.**AutoActivate** [= *value*]

A sintaxe da propriedade **AutoActivate** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Um número inteiro ou constante especificando a técnica usada para ativar o objeto dentro do controle OLE Container, conforme descrito em Configurações.

### Configurações

As configurações de *value* são:

Constante	Valor	Descrição
<b>vbOLEActivateManual</b>	0	Manual. O objeto não é automaticamente ativado. Você pode ativar um objeto por meio de programa usando o método <b>DoVerb</b> .
<b>VbOLEActivateGetFocus</b>	1	Focus. Se o controle OLE Container tiver um objeto que suporte ativação através de clique único, o aplicativo que oferece o objeto será ativado quando o controle OLE Container recebe o foco.
<b>VbOLEActivateDoubleclick</b>	2	(Padrão) Duplo clique. Se o con-

		trole OLE Container tiver um objeto, o aplicativo que oferece o objeto será ativado quando o usuário clicar duas vezes no controle OLE Container ou pressionar ENTER quando o controle tiver o foco.
<b>VbOLEActivateAuto</b>	3	Automatic. Se o controle OLE Container tiver um objeto, o aplicativo que oferecer o objeto será ativado com base no método normal de ativação do objeto {bmc bm3.BMP}ou quando o controle receber o foco ou quando o usuário clicar duas vezes no controle.

### Comentários

Você pode determinar se o controle OLE Container contém um objeto, verificando a propriedade **OLEType**.

**Observação:** Quando **AutoActivate** for configurado como 2 (Double-Clic), o evento DbIClic não ocorrerá quando o usuário clicar duas vezes em um controle OLE Container.

## Propriedade AutoVerbMenu

Retorna ou configura um valor que determina se um menu pop-up contendo os verbos do objeto será exibido quando o usuário clicar no controle OLE Container com o botão direito do mouse.

### Sintaxe

*object.AutoVerbMenu* [ = *boolean*]

A sintaxe da propriedade **AutoVerbMenu** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> especificando se um menu pop-up é exibido, conforme descrito em Configurações.

### Configurações

As configurações de *boolean* são:

Configuração	Descrição
<b>True</b>	(Padrão) Quando o usuário clica no controle OLE Container com o botão direito do mouse, é exibido um menu pop-up, mostrando os comandos suportados pelo objeto.
<b>False</b>	Nenhum menu pop-up é exibido.

### Comentários

Quando esta propriedade for configurada como **True** e o controle OLE Container for clicado com o botão direito do mouse, eventos Clic e MouseDown não ocorrerão.

Para exibir seus próprios menus, a propriedade **AutoVerbMenu** deve ser configurada como **False**.

## Propriedade Object (OLE Container)

Retorna o objeto e/ou uma configuração do método de um objeto ou propriedade em um controle OLE Container.

### Sintaxe

*object.Object* [.*property*] .*method*]

A sintaxe da propriedade **Object** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>property</i>	Propriedade suportada pelo objeto.
<i>method</i>	Método suportado pelo objeto.

### Comentários

Use esta propriedade para especificar um objeto que você deseja usar em uma tarefa de automação.

Você usa o objeto retornado pela propriedade **Object** em uma tarefa de Automation usando as propriedades e métodos daquele objeto. Para maiores informações sobre quais propriedades e métodos são suportados pelo objeto, consulte a documentação do aplicativo que criou o objeto.

## Propriedade OLETypeAllowed

Retorna ou configura o tipo de objeto que o controle OLE Container pode conter.

### Sintaxe

*object*.OLETypeAllowed [ = *value*]

A sintaxe da propriedade **OLETypeAllowed** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Um número inteiro ou constante que especifica o tipo de objeto, conforme descrito em Configurações.

### Configurações

As configurações de *value* são:

Constante	Valor	Descrição
<b>vbOLELined</b>	0	Vinculado. O controle OLE Container pode conter apenas um <u>objeto vinculado</u> .
<b>vbOLEEmbedded</b>	1	Incorporado. O controle OLE Container pode conter somente um <u>objeto incorporado</u> .
<b>vbOLEEEither</b>	2	(Padrão) Qualquer um dos dois. O controle OLE Container pode conter um objeto vinculado ou um objeto incorporado.

### Comentários

Esta propriedade determina o tipo de objeto que um usuário pode criar:

- Ao usar a caixa de diálogo **Insert Object**, use o método **InsertObjDlg** para exibir esta caixa de diálogo.
- Ao usar a caixa de diálogo **Paste Special**, use o método **PasteSpecialDlg** para exibir esta caixa de diálogo.
- Ao colar um objeto a partir da **Área de transferência**, use o método **Paste** para colar objetos da **Área de transferência** do sistema.

Use a propriedade **OLEType** para determinar o tipo de um objeto (vinculado, incorporado ou ambos).

## Método PasteSpecialDlg

Exibe a caixa de diálogo **Paste Special**.

### Sintaxe

*object*.PasteSpecialDlg

O *object* é uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Durante o tempo de execução, você exibe esta caixa de diálogo para permitir ao usuário colar um objeto da **Área de transferência** do sistema. Esta caixa de diálogo exibe diversas opções ao usuário incluindo a colagem de um objeto vinculado ou incorporado.

---

Use a propriedade **OLETypeAllowed** para determinar o tipo de objeto que pode ser criado (vinculado, incorporado, ou ambos) usando esta caixa de diálogo.

Se a propriedade **PasteO** estiver configurada como **True** e o Visual Basic não puder colar o objeto, o OLE Container exclui qualquer objeto já existente no controle.

## Método Copy

Copia o objeto em um controle OLE Container para a **Área de transferência** do sistema.

### Sintaxe

*object*.Copy

O *object* é uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Quando você copia um objeto para a **Área de transferência** do sistema, todos os dados e informações de vínculo associadas ao objeto são colocadas na **Área de transferência** do sistema. Você pode copiar tanto objetos incorporados, quanto vinculados para a **Área de transferência** do sistema.

Você pode usar este método para suportar um comando **Edit Copy** em um menu.

---

## Método CreateEmbed

Cria um objeto incorporado. Não suporta argumentos nomeados.

### Sintaxe

*object.CreateEmbed sourcedoc, class*

A sintaxe do método **CreateEmbed** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>sourcedoc</i>	Obrigatório. O nome de arquivo de um documento usado como modelo para objeto incorporado. Deve ser uma seqüência de caracteres de comprimento zero (""), se você não especifica um documento de origem.
<i>class</i>	Opcional. O nome da classe do objeto incorporado. Ignorado se você especificar um nome de arquivo para <i>sourcedoc</i> .

### Comentários

Para visualizar uma lista de nomes de classe válidos em seu sistema, selecione um controle, por exemplo, o controle OLE Container, selecione a propriedade **Class** na janela **Properties** e clique no botão compilador.

**Observação:** Você não precisa configurar as propriedades **Class** e **SourceDoc** quando usar o método **CreateEmbed** para criar um objeto incorporado.

Quando você criar um novo objeto, o aplicativo associado ao nome de classe (por exemplo, Excel.exe), deve estar corretamente registrado com o sistema operacional (o programa de instalação do aplicativo deve registrar corretamente o aplicativo).

## Método CreateLin

Cria um objeto vinculado a partir do conteúdo de um arquivo. Não suporta argumentos nomeados.

### Sintaxe

*object.CreateLin sourcedoc, sourceitem*

A sintaxe do método **CreateLin** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>sourcedoc</i>	Obrigatório. O arquivo a partir do qual é criado o objeto.
<i>sourceitem</i>	Opcional. Os dados no arquivo a serem vinculados no objeto vinculado.

### Comentários

Se você especificar valores para os argumentos deste método, estes valores ignoram estas configurações das propriedades **SourceDoc** e **SourceItem**. Estas propriedades são atualizadas para refletir os valores dos argumentos quando o método é chamado.

Quando um objeto é criado com este método, o controle OLE Container exibe uma imagem do arquivo especificado pela propriedade **SourceDoc**. Se o objeto for salvo, somente as referências de vínculo serão salvas, pois o controle OLE Container contém apenas uma imagem metarquivo dos dados e nenhum dado de origem verdadeiro.

Quando você criar um novo objeto, o aplicativo associado ao nome de classe (por exemplo, Excel.exe) deve estar corretamente registrado no sistema operacional (o programa de instalação deve registrar corretamente o aplicativo).

## Método Delete (OLE Container)

Exclui o objeto especificado e libera a memória a ele associada.

### Sintaxe

---

### *object.Delete*

O *object* é uma expressão de objeto que avalia para um objeto na lista Applies To.

#### **Comentários**

Este método lhe permite excluir explicitamente um objeto. Os objetos são automaticamente excluídos quando um formulário é fechado ou quando o objeto é substituído por um novo objeto.

## Método InsertObjDlg

Exibe a caixa de diálogo **Insert Object**.

#### **Sintaxe**

*object.InsertObjDlg*

O objeto é uma expressão de objeto que avalia para um objeto na lista Applies To.

#### **Comentários**

Durante o tempo de execução, você exibe esta caixa de diálogo para permitir ao usuário criar um objeto vinculado ou incorporado escolhendo o tipo de objeto (vinculado ou incorporado) e o aplicativo fornece o objeto.

Use a propriedade **OLETypeAllowed** para determinar o tipo de objeto que pode ser criado (vinculado, incorporado, ou ambos) usando esta caixa de diálogo.

Quando você cria um novo objeto, o aplicativo associado ao nome de classe (por exemplo, Excel.exe) deve estar corretamente registrado no sistema operacional (o programa de instalação do aplicativo deve registrar corretamente o aplicativo).

## Método Paste

Copia os dados da **Área de transferência** do sistema para um controle OLE Container.

#### **Sintaxe**

*object.Paste*

O *object* é uma expressão de objeto que avalia para um objeto na lista Applies To.

#### **Comentários**

Para usar este método, configure a propriedade **PasteO** e verifique os valores da propriedade **PasteO**. Você não conseguirá colar a menos que **PasteO** retorne um valor **True**.

Se o método **Paste** foi executado, a propriedade **OLEType** é configurada como **vbOLELined** (0) ou **vbOLEEmbedded** (1). Se o método **Paste** não foi executado, a propriedade **OLEType** é configurada como **vbOLENone** (3).

Você pode usar este método para suportar um comando **Edit Paste** em um menu.

Se a configuração da propriedade **PasteO** for **True** e o Visual Basic não puder colar o objeto, o controle OLE Container exclui qualquer objeto já existente no controle.

---

## Método ReadFromFile

Carrega um objeto a partir de um arquivo de dados criado usando o método **SaveToFile**. Não suporta argumentos nomeados.

### Sintaxe

*object.ReadFromFile* *filenumber*

A sintaxe do método **ReadFromFile** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>filenumber</i>	Obrigatório. A <u>expressão numérica</u> especificando o número do arquivo usado ao carregar um objeto. Este número deve corresponder a um arquivo binário aberto.

### Comentários

Você pode salvar um objeto em um arquivo de dados usando os métodos **SaveToFile** ou **SaveToOle1File**.

## Método SaveToFile

Salva um objeto para um arquivo de dados. Não suporta argumentos nomeados.

### Sintaxe

*object.SaveToFile* *filenumber*

A sintaxe do método **SaveToFile** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>filenumber</i>	Obrigatório. Uma <u>expressão numérica</u> especificando o número do arquivo usado ao salvar um objeto. Este número deve corresponder a um arquivo binário aberto.

### Comentários

Use este método para salvar componentes ActiveX. Para salvar um componente ActiveX no formato OLE versão 1.0 porém, use o método **SaveToOle1File**.

Se o objeto estiver vinculado (**OLEType** = **vbOLELined**, 0), somente a informação do vínculo e uma imagem dos dados serão salvos no arquivo especificado. Os dados do objeto são mantidos pelo aplicativo que criou o objeto. Se o objeto for incorporado (**OLEType** = **vbOLEEmbedded**, 1), os dados do objeto são mantidos pelo controle OLE Container e podem ser salvos pelo seu aplicativo em Visual Basic.

Você pode carregar um objeto salvo em um arquivo de dados com o método **ReadFromFile**.

---

## Método DoVerb

Abre um objeto para uma operação como, por exemplo, editar. Não suporta argumentos nomeados.

### Sintaxe

*object.DoVerb (verb)*

A sintaxe do método **DoVerb** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>verb</i>	Opcional. O <u>verbo</u> para executar o objeto no controle OLE Container. Se não for especificado, o valor deste argumento pode ser um dos verbos padrão suportados por todos os objetos ou um <u>índice</u> da matriz de propriedades <b>ObjectVerbs</b> .

### Comentários

Se você configurar a propriedade **AutoActivate** como 2 (Clicar duas vezes) e o usuário clicar duas vezes no controle, o controle OLE Container ativará automaticamente o objeto atual.

Cada objeto pode suportar seu próprio conjunto de verbos. Os valores abaixo representam verbos padrão que todo objeto deve suportar:

Constante	Valor	Descrição
<b>vbOLEPrimary</b>	0	A ação padrão para o objeto.
<b>vbOLEShow</b>	-1	Ativa o objeto para edição. Se o aplicativo que criou o objeto suportar <u>ativação in-loco</u> , o objeto é ativado dentro do controle OLE Container.
<b>vbOLEOpen</b>	-2	Abre o objeto em uma janela de aplicativo separada. Se o aplicativo que criou o objeto suporta ativação in-loco, o objeto é ativado em sua própria janela.
<b>vbOLEHide</b>	-3	Para objetos incorporados, oculta o aplicativo que criou o objeto.
<b>vbOLEUIActivate</b>	-4	Se o objeto suporta ativação in-loco, ativa o objeto para ativação in-loco e mostra qualquer ferramenta de interface de usuário. Se o objeto não suporta ativação in-loco, o objeto não é ativado e ocorre um erro.
<b>VbOLEInPlaceActivate</b>	-5	Se o usuário move o foco para o controle OLE Container, cria uma janela para o objeto e prepara o objeto para ser editado. Ocorre um erro se o objeto não suporta ativação com um clique de mouse.
<b>VbOLEDiscardUndoState</b>	-6	Usado quando o objeto é ativado para edição, para descartar todos os registros de alterações que o aplicativo do objeto pode desfazer.

**Observação:** Estes verbos podem não estar listados na matriz de propriedades **ObjectVerbs**.



---

## Método FetchVerbs

Atualiza a lista de verbos suportados por um objeto.

### Sintaxe

*object*.FetchVerbs

O *object* é uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Você pode ler a lista atualizada de verbos usando a propriedade **ObjectVerbs**.

## Método Close (OLE Container)

Fecha um objeto e encerra a conexão com o aplicativo que forneceu o objeto.

### Sintaxe

*object*.Close

O *object* é uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Este método aplica-se somente a métodos incorporados e é equivalente a fechar o objeto. Ele não tem qualquer efeito sobre objetos vinculados.

## Método Update (OLE Container)

Recupera os dados atuais do aplicativo que forneceu o objeto e exibe estes dados em forma de elemento gráfico no controle OLE Container.

### Sintaxe

*object*.Update

O *object* é uma expressão de objeto que avalia para um objeto na lista Applies To.

## Propriedade Action (OLE Container)

Configura um valor que determina uma ação. Não disponível durante o tempo de criação.

**Observação:** A propriedade **Action** está incluída para compatibilidade com versões anteriores. Para funcionalidade atual, use o método listado em Configurações.

### Sintaxe

*object*.Action = *value*

A sintaxe da propriedade **Action** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Uma constante ou número inteiro especificando o tipo de ação, conforme descrito em Configurações.

### Configurações

As configurações de *value* são:

Valor	Descrição	Método atual
0	Cria um <u>objeto incorporado</u> .	<b>CreateEmbed</b>
1	Cria um <u>objeto vinculado</u> a partir do conteúdo de um arquivo.	<b>CreateLin</b>
4	Copia o objeto para a <b>Área de transferência</b> do sistema.	<b>Copy</b>
5	Copia os dados da <b>Área de transferência</b> para um controle OLE Container.	<b>Paste</b>
6	Recupera os dados atuais do aplicativo que forneceu o objeto e exibe estes dados como uma figura no controle OLE Container.	<b>Update</b>
7	Abre um objeto para uma operação, por	<b>DoVerb</b>

9	exemplo, edição. Fecha um objeto e termina a conexão com o aplicativo que forneceu o objeto.	<b>Close</b>
10	Exclui o objeto especificado e libera a memória a ele associada.	<b>Delete</b>
11	Salva um objeto em um arquivo de dados.	<b>SaveToFile</b>
12	Carrega um objeto que foi salvo em um arquivo de dados.	<b>ReadFromFile</b>
14	Exibe a caixa de diálogo <b>Insert Object</b> .	<b>InsertObjDlg</b>
15	Exibe a caixa de diálogo <b>Paste Special</b> .	<b>PasteSpecialDlg</b>
17	Atualiza a lista de verbos suportada por um objeto.	<b>FetchVerbs</b>
18	Salva um objeto no formato de arquivo do OLE versão 1.0.	<b>SaveToOle1File</b>

## Propriedade MiscFlags

Retorna ou configura um valor que determina o acesso a um ou mais recursos do controle OLE Container.

### Sintaxe

*object*.MiscFlags [ = *value*]

A sintaxe da propriedade **MiscFlags** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Um número inteiro ou constante especificando acesso a um recurso adicional, conforme descrito em Configurações.

### Configurações

As configurações de *value* são:

Constante	Valor	Descrição
<b>vbOLEMiscFlagMemStorage</b>	1	Faz com que o controle utilize memória para armazenar o objeto enquanto ele é carregado.
<b>VbOLEMiscFlagDisableInPlace</b>	2	Ignora o comportamento padrão do controle de permitir a <u>ativação in-loco</u> para objetos que a suportam.

### Comentários

A configuração do sinalizador **vbOLEMiscFlagMemStorage** é mais rápida do que a ação padrão do objeto, que é armazená-lo em disco como arquivo temporário. Entretanto esta configuração pode usar uma grande quantidade de memória para objetos cujos dados exigem muito espaço, como um bitmap de um programa de pintura.

Se um objeto suportar ativação in-loco, você pode usar a configuração **vbOLEMiscFlagDisableInPlace** para forçar o objeto a ser ativado em uma janela separada.

Para combinar valores, use o operador **Or**. Por exemplo, para combinar ambos os sinalizadores, você pode usar este código:

```
Ole1.MiscFlags = vbOLEMiscFlagMemStorage Or _ vbOLEMiscFlagDisableInPlace
```

## Método SaveToOle1File

Salva um objeto no formato de arquivo OLE versão 1.0. Não suporta argumentos nomeados.

### Sintaxe

*object*.SaveToOle1File *filenumber*

A sintaxe do método **SaveToOle1File** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>filenumber</i>	Obrigatório. Uma <u>expressão numérica</u> especificando o número de arquivo usado ao salvar ou carregar um objeto. Este número deve corresponder a um arquivo binário aberto.

### Comentários

Se o objeto for vinculado (**OLEType** = **vbOLELinked**, 0), somente a informação sobre o vínculo e uma imagem dos dados serão salvos no arquivo especificado. Os dados do objeto são mantidos pelo aplicativo que criou o objeto. Se o objeto for incorporado (**OLEType** = **vbOLEEmbedded**, 1), os dados do objeto são mantidos pelo controle OLE Container e podem ser salvos pelo seu aplicativo em Visual Basic.

Se você deseja salvar o objeto no formato do componente ActiveX atual porém, use o método **SaveToFile**.

## Propriedade Format

Retorna ou configura o formato ao enviar dados a um aplicativo que criou um objeto. Não disponível durante o tempo de criação.

### Sintaxe

*object.Format* [ = *format*]

A sintaxe da propriedade **Format** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>format</i>	Uma <u>expressão de sequência de caracteres</u> especificando o formato usado com as propriedades <b>Data</b> e <b>DataText</b> .

### Comentários

Use as propriedades **ObjectAcceptFormats**, **ObjectAcceptFormatsCount**, **ObjectGetFormats** e **ObjectGetFormatsCount** para obter uma lista de formatos de dados aceitáveis para uma classe específica de objetos.

Muitos aplicativos que oferecem objetos suportam apenas um ou dois formatos. Por exemplo, o Microsoft Draw aceita apenas o formato CF\_METAFILEPICT. Embora CF\_METAFILEPICT se pareça com a constante intrínseca **vbCFMetafile** (valor numérico 3) definida na biblioteca de objetos do Visual Basic (VB) no Object Browser, na realidade, ela é uma sequência de caracteres literal e é designada como:

```
Ole1.Format = "CF_METAFILEPICT"
```

Em muitos casos, a lista de formatos que podem ser aceitos por um objeto (**ObjectAcceptFormats**) é diferente da lista de formatos que um objeto pode oferecer (**ObjectGetFormats**).

## Propriedade PasteO

Retorna um valor que determina se o conteúdo da **Área de transferência** do sistema pode ser colado no controle OLE Container.

### Sintaxe

*object.PasteO*

O *object* é uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Quando esta configuração de propriedade for **True**, você pode colar o conteúdo da **Área de transferência** do sistema no controle OLE Container.

Use a propriedade **OLETypeAllowed** para especificar o tipo de objeto (vinculado ou incorporado) que você deseja colar no controle OLE Container. Uma vez que você consegue colar um objeto no controle OLE Container, você pode verificar a propriedade **OLEType** para determinar o tipo de objeto que foi criado.

Você pode usar esta propriedade se o seu aplicativo suportar um comando **Paste** em um menu **Edit**. Se **PasteO** for **False**, desative o comando de menu, senão ele pode ser ativado. Ative e desative comandos de menu configurando sua propriedade **Enabled** como **True** ou **False** respectivamente.

Você cola um objeto no controle OLE Container com o método **Paste**.

Para oferecer mais flexibilidade ao usuário, exiba uma caixa de diálogo **Paste Special** quando o este escolher o comando **Edit Paste** (configure **OLETypeAllowed** = 2 e use o método **PasteSpecialDlg**). Quando esta caixa de diálogo for exibida, um objeto será colado na **Área de transferência** do sistema com base nas seleções do usuário na caixa de diálogo.

## Exemplo da propriedade DataText

Este exemplo envia dados ao aplicativo Microsoft Graph, de modo que você precisa ter o **MS Graph** instalado em seu sistema para executar o exemplo. (Ele é instalado pela maioria dos componentes do Microsoft Office.) Crie um formulário com cerca de metade do tamanho da tela com um controle **CommandButton** (Command1) no canto superior esquerdo do formulário e um controle de recipiente **OLE** (OLE1) colocado abaixo do **CommandButton**.

Quando você coloca o controle de recipiente **OLE** no formulário, é exibida a caixa de diálogo **Insert Object**. Escolha **Cancel** e pressione F5 para executar o exemplo.

```
Private Sub Command1_Click ()
Dim Msg, NL, TB ' Declarar variáveis.
    TB = Chr(9) ' Caractere de tabulação.
    NL = Chr(10) ' Caractere de nova linha.
    ' Criar dados para substituir os dados padrão de Graph.
    Msg = TB + "Drew" & TB & "Teresa" & TB & "Bob"
    Msg = Msg + NL & "Eric" & TB & "1" & TB & "2" & TB & "3"
    Msg = Msg + NL & "Ted" & TB & "11" & TB & "22" & TB & "33"
    Msg = Msg + NL & "Arthur" & TB & "21" & TB & "32" & TB & "23"
    ' Enviar os dados usando a propriedade DataText.
    ' Ativar MSGRAPH como oculto.
Ole1.DoVerb - 3
    If Ole1.AppIsRunning Then
        Ole1.DataText = Msg
        ' Atualizar o objeto.
        Ole1.Update
    Else
        MsgBox "Graph isn't active."
    End If
End Sub

Sub Form_Load ()
    Ole1.Format = "CF_TEXT" ' Definir o formato do arquivo para texto.
    Ole1.SizeMode = 2 ' Autodimensionar.
    Ole1.CreateEmbed "", "MSGRAPH"
End Sub
```

## Exemplo de propriedades ObjectAcceptFormats, ObjectAcceptFormatsCount, ObjectGetFormats, ObjectGetFormatsCount, ObjectVerbs, ObjectVerbsCount

Para executar este exemplo, coloque um controle de recipiente **OLE** e três controles **ListBox** em um formulário. Cole o exemplo de código na seção Declarations do formulário e pressione F5. Quando a caixa de diálogo **Insert Object** é exibida, selecione um aplicativo na caixa de listagem **New Object** e escolha **O** para criar um objeto.

```
Private Sub Form_Click ()
    Dim I ' Declarar variável.
    ' Exibir a caixa de diálogo Insert Object.
    Ole1.InsertObjDlg
    ' Atualizar a lista de verbos disponíveis.
    Ole1.FetchVerbs ' Capturar verbos.
    ' Limpar as caixas de listagem.
    List1.Clear
```

---

```
List2.Clear
List3.Clear
' Preencher a caixa de listagem de verbos. Como ObjectVerbs(0) é
' o verbo padrão e é repetido na matriz ObjectVerbs(),
' inicie a contagem em 1.
For I = 1 To Ole1.ObjectVerbsCount - 1
    List1.AddItem Ole1.ObjectVerbs(I)
Next I
'Preencher a caixa de listagem Accept Formats.
For I = 0 To Ole1.ObjectAcceptFormatsCount - 1
    List2.AddItem Ole1.ObjectAcceptFormats(I)
Next I
' Preencher a caixa de listagem Get Formats.
For I = 0 To Ole1.ObjectGetFormatsCount - 1
    List3.AddItem Ole1.ObjectGetFormats(I)
Next I
End Sub
```

---

## Exemplo da propriedade PasteO

Este exemplo cola um objeto no controle de recipiente **OLE** se a definição da propriedade **PasteO** for **True**. Caso contrário, o exemplo exibe uma caixa de mensagem.

```
Private Sub mnuEditPaste_Clic ()  
    ' Verificar o valor de PasteO.  
    If Ole1.PasteO Then  
        Ole1.Paste ' Ativar o comando Paste se for True.  
    Else  
        ' Caso contrário, desativar o comando  
        mnuEditPaste.Enabled = False ' de menu Paste e produzir  
        MsgBox "Can't paste." ' a mensagem adequada.  
    End If  
End Sub
```

## Evento ObjectMove

Ocorre imediatamente após o objeto em um controle de recipiente **OLE** ter sido movido ou redimensionado enquanto o objeto está ativo.

### Sintaxe

**Private Sub** *object\_ObjectMove*(*left As Single*, *top As Single*, *width As Single*, *height As Single*)

A sintaxe do evento ObjectMove tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>left</i>	A coordenada da borda esquerda do controle de recipiente <b>OLE</b> imediatamente após ele ter sido movido ou redimensionado.
<i>top</i>	A coordenada da borda superior do controle de recipiente <b>OLE</b> imediatamente após ele ter sido movido ou redimensionado.
<i>width</i>	A largura do controle de recipiente <b>OLE</b> imediatamente após ele ter sido movido ou redimensionado.
<i>height</i>	A altura do controle de recipiente <b>OLE</b> imediatamente após ele ter sido movido ou redimensionado.

### Comentários

Quando um usuário move ou redimensiona um controle de recipiente **OLE**, seu aplicativo pode usar o evento ObjectMove para determinar se realmente altera o tamanho e posição do controle. Se o procedimento de evento ObjectMove não altera a posição ou tamanho do controle de recipiente **OLE**, o objeto dentro do controle de recipiente **OLE** retorna à sua posição original e é informado de seu novo tamanho. As coordenadas passadas como argumentos a este evento incluem a borda do controle de recipiente **OLE**.

Os eventos ObjectMove e Resize são disparados quando o controle de recipiente **OLE** recebe informações sobre o tamanho do objeto que ele contém. Entretanto, o evento Resize não recebe qualquer informação sobre a posição do controle. Se o controle de recipiente **OLE** é movido para fora do formulário, o argumento tem valores positivos ou negativos que representam a posição do objeto em relação à parte superior e esquerda do formulário.

---

## Propriedade OLEDropAllowed

Retorna ou define um valor que determina se um controle de recipiente **OLE** pode ser um destino para soltar em operações OLE de arrastar-e-soltar.

### Sintaxe

*object.OLEDropAllowed* [= *boolean*]

A sintaxe da propriedade **OLEDropAllowed** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Boolean</u> especificando se o controle de recipiente <b>OLE</b> pode ser um destino para soltar, conforme descrito em Configurações.

### Configurações

As configurações para *boolean* são:

Configuração	Descrição
<b>True</b>	Ao arrastar um objeto que possa ser <u>objeto vinculado</u> ou <u>objeto incorporado</u> , aparece um ícone de soltar quando o ponteiro do mouse move-se sobre o controle de recipiente <b>OLE</b> . Arrastar o objeto no controle de recipiente <b>OLE</b> tem o mesmo efeito que colar o objeto da <b>Área de transferência</b> do sistema usando o método <b>Paste</b> .
<b>False</b>	(Padrão) Nenhum ícone de soltar aparece sobre o controle de recipiente <b>OLE</b> quando arrasta um objeto que pode ser vinculado ou incorporado. Soltar o objeto no controle de recipiente <b>OLE</b> não tem qualquer efeito sobre o controle.

### Comentários

A propriedade **MousePointer** determina a forma do ponteiro do mouse quando a propriedade **OLEDropAllowed** está definida como **True**. Se a configuração da propriedade **MousePointer** for 0 (Padrão), o Visual Basic exibe o ícone padrão de arrastar e soltar para a ação que se está realizando.

A configuração da propriedade **OLETypeAllowed** deve ser 1 (**vbOLEEmbedded**) ou 2 (**vbOLEEither**) para mover ou copiar o objeto que pode ser vinculado ou incorporado, ou 0 (**vbOLELinked**) ou 2 para vincular o objeto. Soltar um objeto quando **OLEDropAllowed** está configurado como **True** tem o mesmo efeito sobre as configurações da propriedade **Class**, **SourceDoc** e **SourceItem** que usar o método **Paste** do controle de recipiente **OLE**.

Se a propriedade **OLEDropAllowed** está definida como **True**, o controle de recipiente **OLE** não recebe eventos **DragDrop** ou **DragOver** ao arrastar um objeto. Além disso, a configuração da propriedade **DragMode** não tem nenhum efeito sobre o comportamento de arrastar e soltar do controle de recipiente **OLE** quando a propriedade **OLEDropAllowed** está configurada como **True**.

## Propriedade Instancing

Configura um valor que especifica se você pode criar ocorrências de uma classe pública fora de um projeto e, se for o caso, como ela se comporta. Não está disponível durante o tempo de execução.

### Configurações

A propriedade **Instancing** tem estas configurações:

Configuração	Descrição
1	(Padrão) Particular. Outros aplicativos não têm acesso permitido a informações da biblioteca de tipos sobre a classe, e não podem criar ocorrências dela. Objeto particulares são somente para uso dentro do componente.
2	PublicNotCreatable. Outros aplicativos podem usar objetos desta classe somente se o seu componente cria primeiro os objetos. Outros aplicativos não podem usar a função <b>CreateObject</b> ou o operador <b>New</b> para criar objetos da classe.
3	SingleUse. Permite que outros aplicativos criem objetos da classe, mas todo objeto desta classe criados por um cliente inicia uma nova ocorrência de seu componente. Não é permitido em projetos ActiveX DLL.
4	GlobalSingleUse. Semelhante a SingleUse, exceto que as propriedades e métodos da classe podem ser chamados como se eles fossem simplesmente funções globais. Não é permitida em projetos ActiveX DLL.
5	MultiUse. Permite que outros aplicativos criem objetos a partir da classe. Uma ocorrência de seu componente pode oferecer qualquer quantidade de objetos criados desta maneira.
6	GlobalMultiUse. Semelhante a MultiUse, com uma adição: as propriedades e métodos da classe podem ser acionados como se eles simplesmente fossem funções globais. Não é necessário primeiro criar explicitamente uma ocorrência da classe, pois uma será automaticamente criada.

Configuração	Aplica-se ao tipo de projeto			
	ActiveX Exe	ActiveX DLL	ActiveX Control	Std. Exe
Private	X	X	X	X
PublicNotCreatable	X	X	X	
SingleUse	X			
GlobalSingleUse	X			
MultiUse	X	X		
GlobalMultiUse	X	X		

### Comentários

A propriedade **Instancing** foi expandida no Visual Basic 5.0 para incorporar a funcionalidade da propriedade **Public** do Visual Basic 4.0.

Quando uma classe é criável, você pode usar qualquer uma das técnicas abaixo para criar ocorrências da classe a partir de outros aplicativos:

- Usar a função **CreateObject** como em:  
`Set MyInstance = CreateObject("MyProject.MyClass")`
- Usar a instrução **Dim** no mesmo projeto (ou fora do projeto se a propriedade **Public** também está configurada como **True**), com em:  
`Dim MyInstance As New MyClass`

A palavra chave **New** indica que MyInstance deve ser declarada como uma nova ocorrência de MyClass.

Se a propriedade **Public** for **False**, a configuração da propriedade **Instancing** é ignorada. Você sempre pode criar ocorrências da classe no projeto que define a classe. Se a propriedade **Public** for **True**, a classe é visível e portanto pode ser controlada por outros aplicativos, desde que exista uma ocorrência da classe.



---

**Observação** As propriedades e métodos de um objeto GlobalMultiUse não são parte do espaço de nome global do componente que fornece o objeto. Por exemplo, no projeto que contém o módulo de classe GlobalUtility, você deve criar explicitamente uma ocorrência de GlobalUtility para usar as propriedades e métodos do objeto. Outras limitações de objetos globais estão listadas em "Global Objects and Code Libraries" em "Building Code Components" em Boos Online.

## Evento Initialize

Ocorre quando um aplicativo cria uma ocorrência de um **Form**, **MDIForm**, controle **User**, **Property Page** ou classe.

### Sintaxe

**Private Sub** *object\_Initialize*( )

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Você dispara o evento Initialize quando você:

- Usa a função **CreateObject** para criar uma ocorrência de uma classe. Por exemplo:  
`Set X = CreateObject("Project1.MyClass")`
- Refere-se a uma propriedade ou evento de uma ocorrência automaticamente criada de um formulário ou classe em seu código. Por exemplo:  
`MyForm.Caption = "Exemplo"`

Use este evento para inicializar qualquer dado usado pela ocorrência do **Form**, **MDIForm** ou classe. Para um **Form** ou **MDIForm**, o evento Initialize ocorre antes do evento Load.

## Evento Terminate

Ocorre quando todas as referências a uma ocorrência de um **Form**, **MDIForm**, controle **User**, **Property Page** ou classe são removidas da memória através da configuração de todas as variáveis que se referem ao objeto como **Nothing** ou quando a última referência ao objeto fica fora de escopo.

### Sintaxe

**Private Sub** *object\_Terminate*( )

O espaço reservado *object* representa uma expressão de objeto que avalia um objeto na lista Applies To.

### Comentários

Para todos os objetos, exceto classes, o evento Terminate ocorre após o evento Unload.

O evento Terminate não é disparado se as ocorrências do formulário ou classe foram removidas da memória porque o aplicativo foi irregularmente encerrado. Por exemplo, se o seu aplicativo aciona a instrução **End** antes de remover todas as ocorrências existentes da classe ou formulário da memória, o evento Terminate não é disparado para aquela classe ou formulário.

---

## Propriedade Negotiate

Define o valor que determina se um controle que pode ser alinhado é exibido quando um objeto ativo do formulário exibe uma ou mais barras de ferramentas. Não está disponível durante o tempo de execução.

### Configurações

A propriedade **Negotiate** tem estas configurações:

Configuração	Descrição
<b>True</b>	Se o controle é alinhado no formulário (a propriedade <b>Align</b> é definida como um valor diferente de zero), o controle permanece visível quando um objeto ativo no formulário exibe uma barra de ferramentas.
<b>False</b>	(Padrão) O controle não é exibido quando um objeto ativo no formulário exibe uma barra de ferramentas. A barra de ferramentas do objeto ativo é exibida no lugar do controle.

### Comentários

A propriedade **Negotiate** existe para todos os controles com uma propriedade **Align**. Você usa a propriedade **Align** para alinhar o controle em um **Form** ou objeto **MDIForm**; entretanto, a negociação da barra de ferramentas somente ocorre no **MDIForm**. O controle alinhado deve estar no **MDIForm**.

Se a propriedade **NegotiateToolbars** for definida como **False**, a configuração da propriedade **Negotiate** não tem qualquer efeito.

## Propriedade NegotiateMenus

Configura um valor que determina se um formulário incorpora ou não os menus de um objeto no formulário na barra de menu do formulário. Não está disponível durante o tempo de execução.

### Configurações

A propriedade **NegotiateMenus** tem estas configurações:

Configuração	Descrição
<b>True</b>	(Padrão) Quando um objeto no formulário está ativo para edição, os menus daquele objeto são exibidos na barra de menu do formulário.
<b>False</b>	Os menus de objetos no formulário não são exibidos na barra de menu do formulário.

### Comentários

Usando a propriedade **NegotiateMenus**, você determina se a barra de menu de um formulário compartilhará (ou negociará) espaço com os menus de um objeto ativo no formulário. Se você não deseja incluir os menus do objeto ativo na barra de menu de seu formulário, configure **NegotiateMenus** como **False**.

Você não pode negociar menus entre um objeto **MDIForm** e um objeto no **MDIForm**.

Se **NegotiateMenus** for configurada como **True**, o formulário deve ter definida uma barra de menu, mesmo se a barra de menu não estiver visível. Se a propriedade **MDIChild** do formulário estiver configurada como **True**, os menus do objeto ativo são exibidos na barra de menu da janela pai MDI (objeto **MDIForm**).

Quando **NegotiateMenus** está configurada como **True**, você pode usar a propriedade **NegotiatePosition** de controles **Menu** individuais para determinar os menus que seu formulário exibe junto com os menus do objeto ativo.

---

## Propriedade NegotiatePosition

Configura um valor que determina se controles **Menu** de nível alto são ou não exibidos na barra de menu enquanto um objeto vinculado ou objeto incorporado em um formulário está ativo e exibindo seus menus. Não está disponível durante o tempo de execução.

### Configurações

A propriedade **NegotiatePosition** tem estas configurações:

Configuração	Descrição
0	(Padrão) Nenhum. O menu não é exibido na barra de menu, quando o objeto está ativo.
1	Esquerda. O menu é exibido no lado esquerdo da barra de menu quando o objeto está ativo.
2	Centro. O menu é exibido no centro da barra de menu quando o objeto está ativo.
3	Direita. O menu é exibido à direita na barra de menu quando o objeto está ativo.

### Comentários

Usando a propriedade **NegotiatePosition**, você determina os menus individuais na barra de menu de seu formulário que compartilham (ou negociam) espaço na barra de menu com os menus de um objeto ativo do formulário. Qualquer menu com **NegotiatePosition** configurada como um valor diferente de zero é exibido na barra de menu do formulário, junto com os menus do objeto ativo.

Se a propriedade **NegotiateMenus** for configurada como **False**, a configuração da propriedade **NegotiatePosition** não tem qualquer efeito.

## Propriedade NegotiateToolbars

Define um valor que determina se as barras de ferramentas de um objeto em um formulário MDI filho são exibidas no **MDIForm** quando o objeto no formulário MDI filho está ativo. Não está disponível durante o tempo de execução.

### Configurações

A propriedade **NegotiateToolbars** tem estas configurações:

Configuração	Descrição
<b>True</b>	(Padrão) O objeto <b>MDIForm</b> exibe as barras de ferramentas do objeto ativo na parte superior ou inferior do <b>MDIForm</b> . O objeto ativo determina se as barras de ferramentas são exibidas na parte superior ou inferior do <b>MDIForm</b> .
<b>False</b>	As barras de ferramentas do objeto ativo não são exibidas em hipótese alguma, ou são exibidas como paletas de ferramentas flutuantes, conforme determinado pelo objeto ativo.

### Comentários

Use a propriedade **NegotiateToolbars** quando criar um aplicativo de interface de documento múltiplo (MDI) que inclui objetos em formulários MDI filho. Com esta propriedade, você determina como o objeto ativo exibe suas barras de ferramentas. Configurando esta propriedade como **True**, o **MDIForm** compartilha (ou negocia) espaço na parte superior ou inferior do formulário para exibir as barras de ferramentas do objeto ativo.

Se o **MDIForm** também contém uma barra de ferramentas, use a propriedade **Negotiate** para determinar como as diversas barras de ferramentas compartilham o espaço disponível.

---

## Método Clear (Objeto DataObject)

Exclui o conteúdo do objeto **DataObject**.

### Sintaxe

*object*.Clear

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Este método somente está disponível para origens de arraste de componente. Se **Clear** for acionado a partir de um destino de soltura de componente, é gerado um erro.

A maioria dos componentes suporta eventos arrastar-e-soltar OLE manual e alguns suportam eventos arrastar-e-soltar OLE automáticos.

## Objeto DataObject

O objeto **DataObject** é um recipiente para dados que são transferidos de uma origem de componente para um destino de componente. Os dados são armazenados no formato definido pelo método usando o objeto **DataObject**.

### Sintaxe

**DataObject**

### Comentários

O **DataObject**, que reflete a interface **IDataObject**, permite que operações arrastar-e-soltar, da **Área de transferência** implementadas.

A maioria dos componentes suportam eventos arrastar-e-soltar OLE e alguns suportam eventos arrastar-e-soltar OLE automáticos.

## Coleção DataObjectFiles

Uma coleção de seqüências de caracteres que é o tipo da propriedade **Files** no objeto **DataObject**.

### Sintaxe

*object*.DataObjectFiles

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

A coleção **DataObjectFiles** é uma coleção de seqüências de caracteres que representa um conjunto de arquivos que foram selecionados através do método **GetData** ou através de seleção em um aplicativo como, por exemplo, o Windows Explorer.

Embora a coleção **DataObjectFiles** tenha métodos e propriedades próprias, você deve usar a propriedade Files do objeto **DataObject** para visualizar e manipular o conteúdo da coleção **DataObjectFiles**.

Aqui estão alguns exemplos de código mostrando o uso da propriedade Files para visualizar e manipular dados contidos na coleção **DataObjectFiles** (onde "Data" representa um objeto do tipo **DataObject**):

```
Debug.Print Data.Files(index)
For Each v in Data.Files
    Debug.Print v
Next v
Data.Files.Add "autoexec.bat"
Data.Files.Remove index
Data.Files.Clear
For i = 0 to Data.Files.Count - 1
    Debug.print Data.Files(i)
Next i
```

**Observação:** Esta coleção é usada pela propriedade **Files** somente quando os dados no objeto **DataObject** estão no formato **vbCFFiles**

## Propriedade Files

Retorna uma coleção **DataObjectFiles**, que por sua vez contém uma lista de todos os nomes de arquivo usados por um objeto **DataObject** (por exemplo, os nomes dos arquivos e um usuário arrasta de e para o Windows Explorer.)

### Sintaxe

*object.Files(index)*

A sintaxe da coleção **Files** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto <b>DataObject</b> .
<i>index</i>	Um número inteiro que é um índice de uma matriz de nomes de arquivo.

### Comentários

A coleção **Files** é preenchida com nomes de arquivo somente quando o objeto **DataObject** contém dados do tipo **vbCFFiles** (O objeto **DataObject** pode conter diversos tipos diferentes de dados.) Você pode iterar pela coleção para recuperar a lista de nomes de arquivo.

A coleção **Files** pode ser preenchida para permitir que aplicativos em Visual Basic funcionem como uma fonte de arraste para uma lista de arquivos.

## Método GetData (Objeto DataObject)

Retorna dados de um objeto **DataObject** em forma de uma variante.

### Sintaxe

*object.GetData(format)*

A sintaxe do método **GetData** tem estas partes:

Parte	Descrição
<i>object</i>	Obrigatório. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>format</i>	Uma constante ou valor que especifica o formato da data, conforme descrito em Configurações. A constante ou valor deve estar entre parênteses. Se <i>format</i> for 0 ou omitido, <b>GetData</b> utiliza automaticamente o formato adequado.

### Configurações

As configurações de *format* são:

Constante	Valor	Descrição
<b>vbCFTText</b>	1	Texto (arquivos .txt)
<b>vbCFBitmap</b>	2	<u>Bitmap</u> (arquivos .bmp)
<b>vbCFMetafile</b>	3	<u>metarquivo</u> (arquivos .wmf)
<b>vbCFEMetafile</b>	14	Metarquivos aprimorado (arquivos .emf)
<b>vbCFDIB</b>	8	Bitmap independente de dispositivo (DIB)
<b>vbCFPalette</b>	9	Paleta de cores
<b>vbCFFiles</b>	15	Lista de arquivos
<b>vbCFRTF</b>	-16639	Formato Rich text (arquivos .rtf)

### Comentários

Estas constantes estão listadas na biblioteca de objetos do Visual Basic (VB) no Object Browser. É possível para os métodos **GetData** e **SetData** usar formatos de dados diferentes daqueles listados em Configurações, incluindo formatos definidos pelo usuário registrados com o Windows através da função API `RegisterClipboardFormat()`. Entretanto, existem alguns cuidados a serem tomados:

- O método **SetData** exige que os dados estejam na forma de uma matriz de bytes quando ele não reconhece o formato de dados especificado.
- O método **GetData** sempre retorna dados em uma matriz de bytes quando eles estão em um

formato que ele não reconhece, embora o Visual Basic possa converter de maneira transparente esta matriz de bytes retornada em outros tipos de dados, como uma seqüências de caracteres.

- A matriz de bytes retornada por **GetData** será maior que os dados efetivos ao ser executada em alguns sistemas operacionais, com bytes arbitrários no final da matriz. A razão para isto é que o Visual Basic não conhece o formato dos dados e somente conhece a quantidade de memória que o sistema operacional alocou para os dados. Esta alocação de memória é geralmente maior que o exigido pelos dados. Portanto, podem existir bytes extras junto ao final do segmento de memória alocado. Como resultado, você deve usar funções adequadas para interpretar os dados retornados de uma maneira significativa (por exemplo, truncando uma seqüência de caracteres em um determinado comprimento com a função **Left**, se os dados estiverem em formato de texto.

**Observação:** Nem todos os aplicativos suportam **vbCFBitmap** ou **vbCFPalette**, portanto recomenda-se que você use **vbCFDIB** sempre que possível.

## Método GetFormat (Objeto DataObject)

Retorna um valor booleano indicando se um item no objeto DataObject corresponde a um formato específico. Não suporta argumentos nomeados.

### Sintaxe

*object*.**GetFormat** *format*

A sintaxe do método **GetFormat** tem estas partes:

Parte	Descrição
<i>object</i>	Obrigatório. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>format</i>	Uma constante ou valor que especifica o formato dos dados, conforme descrito em Configurações.

### Configurações

As configurações de *format* são:

Constante	Valor	Descrição
<b>vbCFTText</b>	1	Texto (arquivos .txt)
<b>vbCFBitmap</b>	2	<u>Bitmap</u> (arquivos .bmp)
<b>vbCFMetafile</b>	3	<u>metarquivos</u> (arquivos .wmf)
<b>vbCFEMetafile</b>	14	Metarquivo aprimorado (arquivos .emf)
<b>vbCFDIB</b>	8	Bitmap independente de dispositivo (DIB)
<b>vbCFPalette</b>	9	Paleta de cores
<b>vbCFFiles</b>	15	Lista de arquivos
<b>vbCFRTF</b>	-16639	Formato Rich Text (arquivos .rtf)

### Comentários

Estas constantes são listadas na biblioteca de objetos do Visual Basic (VB) no Object Browser.

O método **GetFormat** retorna como **True** se um item no objeto **DataObject** corresponder ao formato especificado. Caso contrário, ele retorna como **False**.

## Evento OLECompleteDrag

Ocorre quando um componente de origem é solto sobre um componente de destino informando ao componente de origem que uma ação de arraste foi executada ou cancelada.

### Sintaxe

**Private Sub** *object*\_**CompleteDrag**(*effect* As Long)

A sintaxe do evento **CompleteDrag** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>effect</i>	Um número inteiro longo configurado pelo objeto de origem

---

identificando a ação que foi executada, permitindo assim que a origem execute a ação adequada se o componente foi movido (por exemplo, a origem exclui os dados se eles foram movidos de um componente para outro). Os valores possíveis estão listados em Configurações.

### Configurações

As configurações de *effect* são:

Constante	Valor	Descrição
<b>vbDropEffectNone</b>	0	O destino de soltar não pode aceitar os dados ou a operação soltar foi cancelada.
<b>vbDropEffectCopy</b>	1	Soltar resulta em uma cópia dos dados da origem para o destino. Os dados originais permanecem inalterados pela operação de arraste.
<b>vbDropEffectMove</b>	2	Soltar resulta na criação de um vínculo até os dados originais entre a fonte de arrastar e o destino de soltar.

### Comentários

O evento **OLECompleteDrag** é o evento final a ser chamado em uma operação arrastar-e-soltar OLE. Este evento informa ao componente de origem sobre a ação que foi executada quando o objeto foi solto sobre o componente de destino. O destino configura este valor através do parâmetro *effect* do evento **OLEDragDrop**. Com base nisto, a origem pode determinar a ação adequada necessária a ser realizada. Por exemplo, se o objeto foi movido para o destino (**vbDropEffectMove**), a origem precisa excluir o objeto de si após a movimentação.

Se **OLEDragMode** for configurado como **Automatic**, o Visual Basic trata o comportamento padrão. O evento ainda ocorre, entretanto, permitindo ao usuário adicionar ou alterar o comportamento.

A maioria dos componentes suporta eventos arrastar-e-soltar OLE manuais e alguns suportam eventos arrastar-e-soltar OLE automáticos.

## Método OLEDrag

Faz com que um componente inicie uma operação arrastar-e-soltar OLE.

### Sintaxe

*object*.**OLEDrag**

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Quando o método **OLEDrag** é chamado, ocorre o evento **OLEStartDrag** do componente, permitindo que ele forneça dados a um componente de destino.

## Evento OLEDragDrop

Ocorre quando um componente de origem é solto sobre um componente de destino quando o componente de origem determina que possa ser solto.

**Observação:** Este evento somente ocorre se **OLEDragMode** é configurado como **1 (Manual)**.

### Sintaxe

**Private Sub** *object*.**OLEDragDrop**(*data* As **DataObject**, *effect* As **Long**, *button* As **Integer**, *shift* As **Integer**, *x* As **Single**, *y* As **Single**)

A sintaxe do evento **OLEDragDrop** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>data</i>	Um objeto <b>DataObject</b> contendo formatos que a origem fornecerá e, além disso, possivelmente os dados para estes formatos. Se nenhum dado estiver contido em <b>DataObject</b> , ele é fornecido quando o controle chama o método <b>GetData</b> . Os métodos <b>SetDa-</b>

	<b>ta</b> e <b>Clear</b> não podem ser usados aqui.
<i>Effect</i>	Um número inteiro longo configurado pelo componente de destino identificando a ação que foi executada (se existir), permitindo assim que a origem realize a ação adequada se o componente foi movido (por exemplo, a origem excluiu dados). Os valores possíveis são listados em Configurações.
<i>Button</i>	Um número inteiro que funciona como um campo de bits correspondendo ao estado de um botão de mouse quando este não está pressionado. O botão da esquerda é bit 0, o botão da direita é bit 1, e o botão do meio é bit 2. Estes bits correspondem, respectivamente, aos valores 1, 2 e 4. Isto indica o estado dos botões do mouse; algum, todos ou nenhum destes três bits pode ser configurado, indicando que algum, todos ou nenhum dos botões está liberado.
<i>Shift</i>	Um número inteiro que funciona como um campo de bit correspondente ao estado das teclas SHIFT, CTRL e ALT quando liberadas. A tecla SHIFT é bit 0, a tecla CTRL é bit 1 e a tecla ALT é bit 2. Estes bits correspondem, respectivamente, aos valores 1, 2 e 4. O parâmetro <i>shift</i> indica o estado destas teclas; algum, todos ou nenhum dos bits pode ser configurado, indicando que alguma, todas ou nenhuma das teclas está liberada. Por exemplo, se as teclas CTRL e ALT não estiverem pressionadas, o valor de <i>shift</i> seria 6.
<i>x,y</i>	Um número inteiro que especifica a localização atual do ponteiro do mouse. Os valores <i>x</i> e <i>y</i> são sempre expressos em termos do sistema de coordenadas configurado pelas propriedades <b>ScaleHeight</b> , <b>ScaleWidth</b> , <b>ScaleLeft</b> e <b>ScaleTop</b> do objeto.

### Configurações

As configurações de *effect* são:

Constante	Valor	Descrição
<b>vbDropEffectNone</b>	0	O destino de soltar não pode aceitar os dados.
<b>vbDropEffectCopy</b>	1	Soltar resulta em uma cópia dos dados da origem para o destino. Os dados originais permanecem inalterados pela operação de arraste.
<b>VbDropEffectMove</b>	2	Soltar resulta em dados sendo movidos da origem do arraste para origem de soltar. A origem de arrastar deve remover os dados de si mesma após a movimentação.

### Comentários

O componente ActiveX de origem deve sempre mascarar valores do parâmetro *effect* para assegurar a compatibilidade com implementações futuras de componentes ActiveX. Atualmente, são utilizados somente três dos 32 bits no parâmetro *effect*. Em futuras versões do Visual Basic, entretanto, estes outros bits podem ser usados. Portanto, como precaução quanto a problemas futuros, origens de arraste e destinos de soltar devem mascarar estes valores de maneira adequada antes de efetuar qualquer comparação.

Por exemplo, um componente de origem não deve comparar um *effect* a, digamos, **vbDropEffectCopy**, como desta forma:

```
If Effect = vbDropEffectCopy...
```

Ao invés disso, o componente de origem deve mascarar o valor ou valores que estão sendo procurados, assim:

```
If Effect And vbDropEffectCopy = vbDropEffectCopy...
```

-ou-

```
If (Effect And vbDropEffectCopy)...
```

Isto permite a definição de novos efeitos de soltar em versões futuras do Visual Basic, enquanto preserva a compatibilidade com versões anteriores de seu código existente.

A maioria dos componentes suporta eventos arrastar-e-soltar OLE manual, e alguns suportam



## Propriedade OLEDragMode

Retorna ou configura se o componente ou o programador tratar de uma operação arrastar-e-soltar OLE.

### Sintaxe

*object*.**OLEDragMode** = *mode*

A sintaxe da propriedade **OLEDragMode** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>mode</i>	Um número inteiro que especifica o método com o qual um componente trata de operações arrastar-e-soltar OLE, conforme descrito em Configurações.

### Configurações

As configurações de *mode* são:

Constante	Valor	Descrição
<b>vbOLEDragManual</b>	0	(Padrão) Manual. O programador trata de todas as operações arrastar-e-soltar OLE.
<b>vbOLEDragAutomatic</b>	1	Automatic. O componente trata de todas as operações arrastar-e-soltar OLE.

### Comentários

Quando **OLEDragMode** é configurado como **Manual**, você precisa chamar o método **OLEDrag** para iniciar o arraste, o que dispara então o evento **OLEStartDrag**.

Quando **OLEDragMode** é configurado como **Automatic**, o componente de origem preenche o objeto **DataObject** com os dados nele contidos e configura o parâmetro *effect* antes de iniciar o evento **OLEStartDrag** (assim como **OLESetData** e outros eventos arrastar-e-soltar OLE de nível de origem) quando o usuário tenta arrastar para fora do controle. Isto lhe oferece controle sobre a operação arrastar-e-soltar e lhe permite interceder adicionando outros formatos, ou ignorando ou desativando os dados e formatos automáticos usando os métodos **Clear** ou **SetData**.

Se a propriedade **OLEDragMode** da origem estiver configurada como **Automatic** e nenhum dado estiver carregado no evento **OLEStartDrag** ou *aftereffects* estiver configurado como **0**, não ocorrerá a operação arrastar-e-soltar OLE.

**Observação:** Se a propriedade **DragMode** de um controle for configurada como **Automatic**, a configuração de **OLEDragMode** é ignorada, pois eventos arrastar-e-soltar normais do Visual Basic têm precedência.

## Evento OLEDragOver

Ocorre quando um componente é arrastado sobre outro.

### Sintaxe

**Private Sub** *object*.**OLEDragOver**(*data* As **DataObject**, *effect* As **Long**, *button* As **Integer**, *shift* As **Integer**, *x* As **Single**, *y* As **Single**, *state* As **Integer**)

A sintaxe do evento **OLEDragOver** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>data</i>	Um objeto <b>DataObject</b> contendo formatos que a origem fornecerá e, além disso, possivelmente os dados para estes formatos. Se nenhum dado estiver contido no <b>DataObject</b> , ele será fornecido quando chama o método <b>GetData</b> . Os métodos <b>SetData</b> e <b>Clear</b> não podem ser usados aqui.
<i>effect</i>	Um número inteiro longo inicialmente configurado pelo objeto de origem identificando todos os efeitos que ele suporta. Os

	parâmetros devem ser corretamente configurados pelo componente de destino durante este evento. O valor de <i>effect</i> é determinado pela conexão de todos os efeitos ativos por meio do operador <b>Or</b> (conforme listado em Configurações). O componente de destino deve verificar estes efeitos e outros parâmetros para determinar quais ações são adequadas a ele e, em seguida, configurar este parâmetro como um dos efeitos permitidos (conforme especificado pela origem) para especificar quais ações serão executadas se o usuário soltar a seleção sobre o componente. Os valores possíveis estão listados em Configurações.
<i>button</i>	Um número inteiro que funciona como um campo de bits correspondente ao estado de um botão de mouse quando não está pressionado. O botão esquerdo é bit 0, o botão direito bit 1 e o botão do meio é bit 2. Estes bits correspondem, respectivamente, aos valores 1, 2, e 4. Ele indica o estado dos botões do mouse; algum, todos ou nenhum destes três bits pode ser configurado, indicando que algum, todos ou nenhum dos botões está liberado.
<i>shift</i>	Um número inteiro que funciona como um campo de bit correspondente ao estado das teclas SHIFT, CTRL e ALT quando não estão pressionadas. A tecla SHIFT é bit 0, a tecla CTRL é bit 1 e a tecla ALT é bit 2. Estes bits correspondem, respectivamente, aos valores 1, 2 e 4. O parâmetro <i>shift</i> indica o estado destas teclas; algum, todos ou nenhum dos bits pode ser configurado, indicando que alguma, todas ou nenhuma das teclas está liberada. Por exemplo, se tanto a tecla CTRL quanto a tecla ALT não estiverem pressionadas, o valor de <i>shift</i> seria 6.
<i>x,y</i>	Um número inteiro que especifica a posição horizontal (x) e vertical (y) atual do ponteiro do mouse dentro do formulário ou controle de destino. Os valores x e y são sempre expressos em termos do sistema de coordenadas configurado pelas propriedades <b>ScaleHeight</b> , <b>ScaleWidth</b> , <b>ScaleLeft</b> e <b>ScaleTop</b> do objeto.
<i>state</i>	Um número inteiro que corresponde ao estado de transição do controle que está sendo arrastado, em relação ao formulário ou controle de destino. Os valores possíveis estão listados em Configurações.

### Configurações

As configurações de *effect* são:

Constante	Valor	Descrição
<b>vbDropEffectNone</b>	0	O destino de soltar não pode aceitar os dados.
<b>vbDropEffectCopy</b>	1	Soltar resulta em uma cópia dos dados da origem para o destino. Os dados originais permanecem inalterados pela operação de arraste.
<b>vbDropEffectMove</b>	2	Soltar resulta em dados serem movidos da origem de arraste a origem de soltar. A origem de arraste deve remover os dados de si mesma após a movimentação.
<b>vbDropEffectScroll</b>	-2147483648 (&H80000000)	A rolagem está ocorrendo ou a ponto de acontecer no componente de destino. Este valor é usado em conjunto com os outros valores. <b>Observação:</b> Use somente se você estiver executando sua própria rola-

gem no componente de destino.

As configurações de *state* são:

Constante	Valor	Descrição
<b>vbEnter</b>	0	O componente de origem está sendo arrastado dentro do intervalo de um destino.
<b>vbLeave</b>	1	O componente de origem está sendo arrastado para fora do intervalo de um destino.
<b>vbOver</b>	2	O componente foi movido de uma posição no destino a outra.

### Comentários

**Observação:** Se o parâmetro *state* for **vbLeave**, indicando que o ponteiro do mouse deixou o destino, então os parâmetros *x* e *y* conterão zeros.

O componente de origem deve sempre mascarar valores do parâmetro *effect* para assegurar a compatibilidade com futuras implementações de componentes ActiveX. Atualmente, somente três dos 32 bits do parâmetro *effect* são utilizados. Em futuras versões do Visual Basic, entretanto, estes outros bits podem ser usados. Portanto, como precaução contra problemas futuros, origens de arraste e destinos de soltar devem mascarar estes valores de maneira adequada, antes de executar qualquer comparação.

Por exemplo, um componente de origem não deve comparar um *effect* a, digamos, **vbDropEffectCopy**, desta maneira:

```
If Effect = vbDropEffectCopy...
```

Ao invés disso, o componente de origem deve mascarar o valor ou valores que estão sendo procurados, assim:

```
If Effect And vbDropEffectCopy = vbDropEffectCopy...
```

-ou-

```
If (Effect And vbDropEffectCopy)...
```

Isto permite a definição de novos efeitos de soltar em futuras versões do Visual Basic, enquanto preserva compatibilidade com versões anteriores de seu código existente.

A maioria dos componentes suporta eventos arrastar-e-soltar OLE manual e alguns suportam eventos arrastar-e-soltar OLE automáticos.

## Propriedade OLEDropMode

Retorna ou configura a maneira como um componente de destino trata operações soltar.

### Sintaxe

*object.OLEDropMode* [= *mode*]

A sintaxe da propriedade **OLEDropMode** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>mode</i>	Uma enumeração de números inteiros que especifica o método que um componente trata operações arrastar-e-soltar OLE, conforme descrito em Configurações.

### Configurações

As configurações de *mode* são:

Constante	Valor	Descrição
<b>vbOLEDropNone</b>	0	(Padrão) None. O componente de destino não aceita soltar OLE e exibe o cursor No Drop.
<b>vbOLEDropManual</b>	1	Manual. O componente de destino dispara os eventos soltar OLE, permitindo ao programador manipular a operação soltar OLE em código.
<b>vbOLEDropAutomatic</b>	2	Automatic. O componente de destino aceita automaticamente soltar OLE se o objeto <b>DataObject</b> tiver dados

---

em um formato por ele reconhecido. Nenhum evento de mouse ou arrastar-e-soltar OLE no destino ocorrerá quando **OLEDropMode** estiver configurado como **vbOLEDropAutomatic**.

### Comentários

**Observação:** O componente de destino inspeciona o que está sendo arrastado sobre ele para determinar quais eventos disparar; os eventos arrastar-e-soltar OLE ou eventos arrastar-e-soltar do Visual Basic. Não existe colisão de componentes ou confusão sobre quais eventos são disparados, pois somente um tipo de objeto pode ser arrastado em um determinado momento.

## Evento OLEGiveFeedback

Ocorre após cada evento OLEDragOver. OLEGiveFeedback permite ao componente de origem oferecer indicações visuais ao usuário como, por exemplo, a mudança do cursor do mouse para indicar o que acontecerá se o usuário soltar o objeto, ou oferece indicação visual na seleção (no componente de origem) para indicar o que acontecerá.

### Sintaxe

**Private Sub** *object* **OLEGiveFeedback**(*effect* **As Long**, *defaultcursors* **As Boolean**)

A sintaxe do evento OLEGiveFeedback tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>effect</i>	Um número inteiro longo configurado pelo componente de destino no evento OLEDragOver especificando a ação a ser executada se o usuário soltar a seleção sobre ele. Isto permite que a origem realize a ação adequada (por exemplo, oferecer indicação visual). Os valores possíveis estão listados em Configurações
<i>defaultcursors</i>	Um valor booleano que determina se o Visual Basic usa o cursor de mouse padrão fornecido pelo componente ou usa um cursor de mouse definido pelo usuário. <b>True</b> (Padrão) = usa o cursor de mouse padrão. <b>False</b> = não usa o cursor de mouse padrão. O cursor de mouse deve ser configurado com a propriedade <b>Mouse-Pointer</b> do objeto <b>Screen</b> .

### Configurações

As configurações de *effect* são:

Constante	Valor	Descrição
<b>vbDropEffectNone</b>	0	Destino de soltar não pode aceitar os dados.
<b>vbDropEffectCopy</b>	1	Soltar resulta em uma cópia dos dados da origem para o destino. Os dados originais permanecem inalterados pela operação de arraste.
<b>vbDropEffectMove</b>	2	Soltar resulta em dados sendo movidos da fonte de arraste para a fonte de soltar. A fonte de arraste deve remover os dados de si mesma após a movimentação.
<b>vbDropEffectScroll</b>	-2147483648 (&H80000000)	A rolagem está ocorrendo ou a ponto de ocorrer no componente de destino. Este valor é usado em conjunto com os outros valores. <b>Observação:</b> Use somente se você estiver executando sua própria rola-

---

gem no componente de destino.

### Comentários

Se não houver código no evento `OLEGiveFeedback`, ou se o parâmetro `defaultcursors` estiver configurado como **True**, o Visual Basic configura automaticamente o cursor do mouse como o cursor padrão fornecido pelo componente.

O componente de origem deve sempre mascarar valores do parâmetro *effect* para assegurar a compatibilidade com implementações futuras de componentes. Atualmente, são usados somente três dos 32 bits no parâmetro *effect*. Em versões futuras do Visual Basic, entretanto, estes outros bits podem ser usados. Portanto, como uma precaução contra problemas futuros, origens de arraste e destinos de soltar devem mascarar estes valores de maneira adequada antes de executar qualquer comparação.

Por exemplo, um componente de origem não deve comparar um *effect* a, digamos, **vbDropEffectCopy**, da maneira abaixo:

```
If Effect = vbDropEffectCopy...
```

Ao invés disso, o componente de origem deve mascarar o valor ou valores que estão sendo procurados, desta forma:

```
If Effect And vbDropEffectCopy = vbDropEffectCopy...
```

-ou-

```
If (Effect And vbDropEffectCopy)...
```

Isto permite a definição de novos efeitos de soltar em versões futuras do Visual Basic enquanto preserva a compatibilidade com versões mais antigas de seu código existente.

A maioria dos componentes suporta eventos arrastar-e-soltar OLE manuais e alguns suportam eventos arrastar-e-soltar OLE automáticos.

---

## Evento OLESetData

Ocorre em um componente de origem quando um componente de origem executa o método **GetData** no objeto **DataObject** da origem, mas os dados para o formato especificado ainda não foram carregados.

### Sintaxe

**Private Sub** *object\_***OLESetData**(*data* As **DataObject**, *dataformat* As **Integer**)

A sintaxe do evento **OLESetData** tem estas partes:

Parte	Descrição
<i>Object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>Data</i>	Um objeto <b>DataObject</b> onde colocar os dados solicitados. O componente chama o método <b>SetData</b> para carregar o formato solicitado.
<i>dataformat</i>	Um número inteiro especificando o formato dos dados que o componente de destino está solicitando. O componente de origem usa este valor para determinar o que carregar no objeto <b>DataObject</b> .

### Comentários

Em determinados casos, você pode desejar adiar a carga dos dados no objeto **DataObject** de um componente de origem para economizar tempo, especialmente se o componente de origem suportar muitos formatos. Este evento permite que a fonte responda somente a uma solicitação de um determinado formato de dados. Quando este evento é chamado, a origem deve verificar o parâmetro *format* para determinar o que precisa ser carregado e, em seguida, executa o método **SetData** no objeto **DataObject** para carregar os dados que são, a seguir, passados de volta ao componente de destino.

## Evento OLEStartDrag

Ocorre quando o método **OLEDrag** de um componente é executado, ou quando um componente inicia uma operação arrastar-e-soltar quando a propriedade **OLEDragMode** está configurada como **Automatic**.

Este evento especifica os formatos de dados e efeitos soltar suportados pelo componente de origem. Ele também pode ser usado para inserir dados no objeto **DataObject**.

### Sintaxe

**Private Sub** *object\_***StartDrag**(*data* As **DataObject**, *allowedeffects* As **Long**)

A sintaxe do evento **StartDrag** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>data</i>	Um objeto <b>DataObject</b> contendo formatos que a origem fornecerá e, opcionalmente, os dados para estes formatos. Se nenhum dado estiver contido em <b>DataObject</b> , eles são fornecidos quando o controle chama o método <b>GetData</b> . O programador deve fornecer os valores deste parâmetro neste evento. Os métodos <b>SetData</b> e <b>Clear</b> não podem ser usados aqui.
<i>allowedeffects</i>	Um número inteiro longo contendo os efeitos suportados pela origem. Os valores possíveis são listados em Configurações. O programador deve fornecer os valores para este parâmetro neste evento.

### Configurações

As configurações de *allowedeffects* são:

Constante	Valor	Descrição
<b>vbDropEffectNone</b>	0	O destino de soltar não aceita os dados.
<b>vbDropEffectCopy</b>	1	Soltar resulta em uma cópia dos dados da origem para o destino. Os dados ori-

## vbDropEffectMove

2

ginais permanecem inalterados pela operação de arraste. Soltar resulta em os dados sendo movidos da origem de arrastar à origem de soltar. A origem de soltar deve remover os dados de si mesma após a movimentação.

### Comentários

O componente de origem deve conectar logicamente os valores suportados através do operador Or e colocar os resultados no parâmetro *allowedeffects*. O componente de destino pode usar este valor para determinar a ação adequada (e qual deve ser a indicação visual adequada para o usuário).

O evento StartDrag também ocorre se a propriedade **OLEDragMode** do componente estiver configurada como **Automatic**. Isto permite adicionar formatos e dados ao objeto **DataObject** após o componente ter feito isto. Você também pode ignorar o comportamento padrão do componente limpando o objeto **DataObject** (usando o método **Clear**) e, em seguida, adicionando seus dados e formatos.

Você pode querer adiar a colocação dos dados no objeto **DataObject** até que o componente de destino os solicite. Isto permite ao componente de origem economizar tempo não carregando formatos de dados múltiplos. Quando o destino executa o método **GetData** sobre o **DataObject**, o evento **OLESetData** da origem ocorrerá se os dados solicitados não estiverem contidos no **DataObject**. Neste ponto, os dados podem ser carregados no **DataObject** que fornecerá os dados ao destino.

Se o usuário não carregar qualquer formato no **DataObject** a operação arrastar-e-soltar será cancelada.

## Método SetData (Objeto DataObject)

Insere dados em um objeto **DataObject** usando o formato de dados especificado.

### Sintaxe

*object.SetData* [*data*], [*format*]

A sintaxe do método **SetData** tem estas partes:

Parte	Descrição
<i>object</i>	Obrigatório. Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>data</i>	Opcional Uma variante contendo os dados a serem passados ao objeto <b>DataObject</b> .
<i>format</i>	Opcional. Uma constante ou valor que especifica o formato dos dados que estão sendo passados, conforme descrito em Configurações.

### Configurações

As configurações de *format* são:

Constante	Valor	Descrição
<b>vbCFTText</b>	1	Texto (arquivos .txt)
<b>vbCFBitmap</b>	2	Bitmap (arquivos .bmp)
<b>vbCFMetafile</b>	3	<u>Metarquivos</u> (arquivos .wmf)
<b>vbCFEMetafile</b>	14	Metarquivos aprimorados (arquivos .emf)
<b>vbCFDIB</b>	8	Bitmap independente de dispositivo (DIB)
<b>vbCFPalette</b>	9	Paleta de cores
<b>vbCFFiles</b>	15	Lista de arquivos
<b>vbCFRTF</b>	-16639	Formato Rich Text (arquivos .rtf)

### Comentários

Estas constantes são listadas na biblioteca de objetos do Visual Basic (VB) no Object Browser.

O argumento *data* é opcional. Isto permite configurar diversos formatos diferentes que o componente de origem pode suportar sem ter que carregar os dados de maneira separada para cada formato. Formatos múltiplos são configurados chamando **SetData** diversas vezes, cada vez

usando um formato diferente. Se você desejar um novo início, use o método **Clear** para limpar todos os dados e informações de formato do **DataObject**.

O argumento *format* também é opcional, mas tanto o argumento *data* quanto o argumento *format* devem ser especificados. Se *data* for especificado, mas não *format*, o Visual Basic tentará determinar o formato dos dados. Se não conseguir, um erro será gerado. Quando o destino solicitar os dados, e um formato for especificado mas nenhum dado fornecido, ocorrerá o evento **OLESetData** da origem e ela pode então fornecer o tipo de dados solicitado.

É possível para os métodos **GetData** e **SetData** usar formatos de dados diferentes daqueles listados em Configurações, incluindo formatos definidos pelo usuário registrados no Windows através da função API `RegisterClipboardFormat()`. Entretanto, deve-se atentar para alguns pontos:

- O método **SetData** exige que os dados estejam na forma de uma matriz de bytes quando ele não reconhece o formato de dados especificado.
- O método **GetData** sempre retorna dados em matriz de bytes quando ele está em um formato que ele não reconhece, embora o Visual Basic possa converter de maneira transparente esta matriz de bytes retornada em outros tipos de dados, como seqüências de caracteres.
- A matriz de bytes retornada por **GetData** será maior que os próprios dados ao ser executada em alguns sistemas operacionais, com bytes arbitrários ao final da matriz. A razão é que o Visual Basic não conhece o formato dos dados e conhece apenas a quantidade de memória alocada para os dados pelo sistema operacional. Esta alocação de memória é com freqüência maior que aquela efetivamente exigida pelos dados. Portanto, podem existir bytes extras junto ao final do segmento de memória alocado. Como resultado, você deve usar funções adequadas para interpretar os dados retornados de uma forma significativa (por exemplo, truncando uma seqüência de caracteres em um determinado comprimento com a função **Left**, caso os dados estejam em um formato de texto).

## Propriedades **BacColor**, **ForeColor**

- **BacColor** — retorna ou configura a cor de segundo plano de um objeto.
- **ForeColor** — retorna ou configura a cor de primeiro plano usada para exibir texto e elementos gráficos em um objeto.

### Sintaxe

*object*.**BacColor** [= *color*]

*object*.**ForeColor** [= *color*]

A sintaxe das propriedades **BacColor** e **ForeColor** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>color</i>	Um valor ou <u>constante</u> que determina as cores de segundo e primeiro planos de um objeto, conforme descrito em Configurações.

### Configurações

O Visual Basic usa o esquema de cores RGB (vermelho-verde-azul) do ambiente operacional do Microsoft Windows. As configurações de *color* são:

Configuração	Descrição
Cores RGB normais	As cores especificadas pelo uso da paleta Color ou pelo uso das funções <b>RGB</b> ou <b>QBColor</b> em código.
Cores padrão do sistema	As cores especificadas pelas constantes de cor do sistema listadas na <u>biblioteca de objetos</u> do Visual Basic (VB) no <u>Object Browser</u> . O ambiente operacional do Windows substitui as escolhas do usuário conforme especificado nas configurações do <b>Painel de controle</b> .

Para todos os formulários e controles, as configurações padrão durante o tempo de criação são:

- **BacColor** — configura a cor padrão do sistema especificada pela constante **vbWindowBackground**.
- **ForeColor** — configura a cor padrão do sistema especificada pela constante **vbWindowText**.

### Comentários



---

Nos controles **Label** e **Shape**, a propriedade **BacColor** é ignorada se a configuração da propriedade **BacStyle** for 0 (Transparente).

Se você configurar a propriedade **BacColor** de um objeto **Form** ou controle **PictureBox**, todos os textos e elementos gráficos, incluindo elementos gráficos permanentes, são apagados. A configuração da propriedade **ForeColor** não afeta os elementos gráficos ou resultados impressos já desenhados. Em todos os outros controles, a cor da tela se altera imediatamente.

O intervalo válido para uma cor RGB normal é de 0 a 16.777.215 (&HFFFFFF). O byte alto de um número neste intervalo é igual a 0; os 3 bytes mais baixos, do menos significativo até o mais significativo determinam, respectivamente, a quantidade de vermelho, verde e azul. Os componentes vermelho, verde e azul são cada um deles representados por um número entre 0 e 255 (&HFF). Se o byte alto não for 0, o Visual Basic utiliza as cores do sistema, conforme definido nas configurações do **Painel de controle** do usuário e por constantes listadas na [biblioteca de objetos](#) do Visual Basic (VB) no [Object Browser](#).

Para exibir texto no ambiente operacional do Windows, tanto o texto quando as cores de segundo plano devem ser uniformes. Se o texto ou cores de segundo plano que você selecionou não forem exibidos, uma das cores selecionadas pode ser pontilhada — ou seja, composta de até três pixels de cores diferentes. Se você escolher uma cor pontilhada para o texto ou para o segundo plano, a cor uniforme mais próxima será substituída.

---

## Propriedade BacStyle

Retorna ou configura um valor indicando se um controle **Label** ou o segundo plano de um controle **Shape** é transparente ou opaco.

### Sintaxe

*object.BacStyle* [= *number*]

A sintaxe da propriedade **BacStyle** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Uma <u>expressão numérica</u> especificando a transparência, conforme descrito em Configurações.

### Configurações

As configurações de *number* são:

Configuração	Descrição
0	Transparente — a cor de segundo plano e qualquer elemento gráfico são visíveis atrás do controle.
1	(Padrão) Opaco — a configuração da propriedade <b>BacColor</b> do controle preenche o controle e obscurece qualquer cor ou gráfico atrás dele.

### Comentários

Você pode usar a propriedade **BacStyle** para criar controles transparentes quando estiver usando uma cor de segundo plano em um objeto **Form**, controle **PictureBox** ou quando desejar colocar um controle em um elemento gráfico. Use um controle opaco quando desejar que ele se destaque.

A propriedade **BacColor** de um controle é ignorada se **BacStyle** = 0.

## Propriedade BorderColor

Retorna ou configura a cor da borda de um objeto.

### Sintaxe

*object.BorderColor* [= *color*]

A sintaxe da propriedade **BorderColor** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>color</i>	Um valor ou <u>constante</u> que determina a cor da borda, conforme descrito em Configurações.

### Configurações

O Visual Basic usa o esquema de cores RGB (vermelho-verde-azul) do ambiente operacional do Microsoft Windows. As configurações de *color* são:

Configuração	Descrição
Cores RGB normais	As cores especificadas usando a paleta Color ou usando as funções <b>RGB</b> ou <b>QBColor</b> no código.
Cores padrão do sistema	As cores especificadas por constantes de cor do sistema listadas na <u>biblioteca de objetos</u> do Visual Basic (VB) no <u>Object Browser</u> . A cor padrão do sistema é especificada pela constante <b>vbWindowText</b> . O ambiente operacional do Windows substitui as opções do usuário conforme especificado nas configurações do Painel de controle.

### Comentários

O intervalo válido para uma cor RGB normal é de 0 a 16.777.215 (&HFFFFFF). O byte alto de um número neste intervalo é igual a 0; os 3 bytes mais baixos, do menos significativo até o mais si-

gnificante determinam, respectivamente, a quantidade de vermelho, verde e azul. Os componentes vermelho, verde e azul são cada um deles representados por um número entre 0 e 255 (&HFF). Se o byte alto não for 0, o Visual Basic utiliza as cores do sistema, conforme definido nas configurações do **Painel de controle** do usuário e por constantes listadas na biblioteca de objetos do Visual Basic (VB no Object Browser).

## Propriedade BorderStyle

Retorna ou configura o estilo de borda de um objeto. Para o objeto **Form** e o controle **TextBox**, somente leitura durante o tempo de execução.

### Sintaxe

*object*.BorderStyle = [*value*]

A sintaxe da propriedade **BorderStyle** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Um valor ou <u>constante</u> que determina o estilo de borda, conforme descrito em Configurações.

### Configurações

As configurações da propriedade **BorderStyle** para um objeto **Form** são:

Constante	Configuração	Descrição
<b>vbBSNone</b>	0	None (nenhuma borda ou elementos relacionados a ela).
<b>vbFixedSingle</b>	1	Fixed Single. Pode incluir a caixa de menu <b>Control</b> , <u>barra de título</u> , <u>botão Maximizar</u> e <u>botão Minimizar</u> . Redimensionável somente usando os botões <b>Maximizar</b> e <b>Minimizar</b> .
<b>vbSizable</b>	2	(Padrão) Sizable. Redimensionável usando qualquer um dos elementos de borda opcionais listados para a configuração 1.
<b>vbFixedDouble</b>	3	Fixed Dialog. Pode incluir caixa do menu <b>Control</b> barra de título; não pode incluir botões <b>Maximizar</b> ou <b>Minimizar</b> . Não é redimensionável.
<b>vbFixedToolWindow</b>	4	Fixed ToolWindow. Exibe uma janela não-dimensionável com um botão <b>Fechar</b> e texto de barra de título em um tamanho de fonte reduzido. O formulário não aparece na barra de tarefas do Windows 95.
<b>vbSizableToolWindow</b>	5	Sizable ToolWindow. Exibe uma janela dimensionável com um botão <b>Fechar</b> e texto de barra de título em um tamanho da fonte redu-

zido. O formulário não aparece na barra de tarefas do Windows 95.

As configurações de propriedade **BorderStyle** para os controles **MS Flex Grid**, **Image**, **Label**, recipiente **OLE**, **PictureBox**, **Frame** e **TextBox** são:

Configuração	Descrição
0	(Padrão para os controles <b>Image</b> e <b>Label</b> ) None.
1	(Padrão para os controles <b>MS Flex Grid</b> , <b>PictureBox</b> , <b>TextBox</b> , e recipiente <b>OLE</b> ) Fixed Single.

As configurações de propriedade **BorderStyle** para os controles **Line** e **Shape** são:

Constante	Configuração	Descrição
<b>vbTransparent</b>	0	Transparente
<b>vbBSSolid</b>	1	(Padrão) Uniforme. A borda é centralizada na borda da forma.
<b>vbBSDash</b>	2	Traço
<b>vbBSDot</b>	3	Ponto
<b>vbBSDashDot</b>	4	Traço-ponto
<b>vbBSDashDotDot</b>	5	Traço-ponto-ponto
<b>vbBSInsideSolid</b>	6	Uniforme interno. A face externa da borda é a face externa da forma.

### Comentários

Para um formulário, a propriedade **BorderStyle** determina características fundamentais que identificam visualmente um formulário como sendo uma janela de uso geral ou uma caixa de diálogo. A configuração 3 (Fixed Dialog) é útil para as caixas de diálogo padrão. As configurações 4 (Fixed ToolWindow) e 5 (Sizable ToolWindow) são úteis para se criar janelas no estilo de caixa de ferramentas.

Formulários MDI filho configurados como 2 (Sizable) são exibidos dentro do formulário MDI em um tamanho padrão definido pelo ambiente operacional do Windows durante o tempo de execução. Para qualquer outra configuração, o formulário é exibido no tamanho especificado durante o tempo de criação.

A alteração da configuração da propriedade **BorderStyle** de um objeto **Form** pode alterar as configurações das propriedades **MinButton**, **MaxButton** e **ShowInTaskbar**. Quando **BorderStyle** é configurada como 1 (Fixed Single) ou 2 (Sizable), as propriedades **MinButton**, **MaxButton** e **ShowInTaskbar** são automaticamente configuradas como **True**. Quando **BorderStyle** é configurada como 0 (None), 3 (Fixed Dialog), 4 (Fixed ToolWindows) ou 5 (Sizable ToolWindow), as propriedades **MinButton**, **MaxButton** e **ShowInTaskbar** são automaticamente configuradas como **False**.

**Observação:** Se um formulário com um menu for configurado como 3 (Fixed Dialog), ele será então exibido com uma configuração de borda 1 (Fixed Single).

Durante o tempo de execução, um formulário é de janela restrita ou sem janela restrita, que você especifica usando o método **Show**.

## Propriedade BorderWidth

Retorna ou configura a largura da borda de um controle.

### Sintaxe

*object*.**BorderWidth** [= *number*]

A sintaxe da propriedade **BorderWidth** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Uma <u>expressão numérica</u> de 1 to 8192, inclusive.

### Comentários

Use as propriedades **BorderWidth** e **BorderStyle** para especificar o tipo de borda que você deseja para um controle **Line** ou **Shape**. A tabela abaixo mostra os efeitos da configuração **BorderStyle** sobre a propriedade **BorderWidth**:

<b>BorderStyle</b>	<b>Efeito sobre BorderWidth</b>
0	A configuração <b>BorderWidth</b> é ignorada.
1–5	A largura da borda expande-se do centro para a borda; a altura e a largura do controle são medidas do centro até a face externa da borda.
6	A largura da borda expande-se para dentro do controle, a partir da face externa da borda; a altura e a largura do controle são medidas a partir a face externa da borda.

Se a configuração da propriedade **BorderWidth** for maior que 1, as únicas configurações efetivas de **BorderStyle** são 1 (Solid) e 6 (Inside Solid).

## Propriedade Cancel

Retorna ou configura um valor indicando se um botão de comando é o botão **Cancel** de um formulário. Este botão de comando pode ser o controle **CommandButton** ou qualquer objeto dentro de um controle de recipiente **OLE** que se comporte como um botão de comando.

### Sintaxe

*object.Cancel* [= *boolean*]

A sintaxe da propriedade **Cancel** tem estas partes:

<b>Parte</b>	<b>Descrição</b>
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Boolean</u> especificando se o objeto é o botão <b>Cancel</b> , conforme descrito em Configurações.

### Configurações

As configurações de *boolean* são:

<b>Configuração</b>	<b>Descrição</b>
<b>True</b>	O controle <b>CommandButton</b> é o botão <b>Cancel</b> .
<b>False</b>	(Padrão) O controle <b>CommandButton</b> não é o botão <b>Cancel</b> .

### Comentários

Use a propriedade **Cancel** para dar ao usuário a opção de cancelar alterações não efetivadas e retornar o formulário a seu estado anterior.

Somente um controle **CommandButton** em um formulário pode ser o botão **Cancel**. Quando a propriedade **Cancel** for configurada como **True** para um **CommandButton**, ela será automaticamente configurada como **False** para todos os outros controles **CommandButton** do formulário. Quando a propriedade **Cancel** de um controle **CommandButton** for configurada como **True** e o formulário for o formulário ativo, o usuário pode escolher o **CommandButton** clicando nele, pressionando a tecla ESC ou pressionando ENTER quando o botão tiver o foco.

Para controles de recipiente **OLE**, a propriedade **Cancel** é fornecida somente para aqueles objetos que se comportam especificamente como botões de comando.

**Dica** Para um formulário que suporta operações irreversíveis, como exclusões, é uma boa idéia deixar o botão **Cancel** como botão padrão. Para fazer isto, configure ambas as propriedades **Cancel** e **Default** como **True**.

## Propriedades Col e Row

Retornam ou configuram a célula ativa em um controle **DBGrid**. Não estão disponíveis durante o tempo de criação.

### Sintaxe

*object.Col* [= *number*]

---

*object*.**Row** [= *number*]

A sintaxe das propriedades **Col** e **Row** tem estas partes:

<b>Parte</b>	<b>Descrição</b>
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	O número da coluna ou linha contendo a célula ativa.

### **Comentários**

Use esta propriedade para especificar uma célula em um controle **DBGrid** ou para descobrir qual coluna ou linha contém a célula ativa em uma área selecionada. As colunas e linhas são numeradas a partir de zero, começando na parte superior para as linhas, e à esquerda para as colunas. A configuração destas propriedades durante o tempo de execução não altera células que estejam selecionadas. Use as propriedades **SelEndCol**, **SelStartCol**, **SelEndRow** e **SelStartRow** para especificar uma área selecionada.

**Observação:** As propriedades **Col** e **Row** não são iguais às propriedades **Colse Rows**.

---

## Propriedades Cols e Rows

Retornam ou configuram o número total de colunas ou linhas em um controle **DBGrid**.

### Sintaxe

*object.Cols* [= *number*]  
*object.Rows* [= *number*]

A sintaxe das propriedades **Cols** e **Rows** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	O número de colunas ou linhas em um controle <b>DBGrid</b> . O número mínimo de colunas é 1 e o máximo é 400; o número mínimo de linhas é 1 e o máximo é 2000.

### Comentários

Use estas propriedades para expandir dinamicamente um controle **DBGrid** durante o tempo de execução. Um controle **DBGrid** deve ter pelo menos uma coluna não-fixa e uma linha não-fixa.

**Observação:** As propriedades **Cols** e **Rows** não são iguais às propriedades **Col** e **Row**.

## Propriedade ControlBox

Retorna ou configura um valor indicando se uma caixa do menu **Control** é exibida em um formulário durante o tempo de execução. Somente leitura durante o tempo de execução.

### Sintaxe

*object.ControlBox*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Configurações

As configurações da propriedade **ControlBox** são:

Configuração	Descrição
<b>True</b>	(Padrão) Exibe a caixa do menu <b>Control</b> .
<b>False</b>	Remove a caixa do menu <b>Control</b> .

### Comentários

Para exibir uma caixa do menu **Control**, você também deve configurar a propriedade **BorderStyle** do formulário como 1 (Fixed Single), 2 (Sizable) ou 3 (Fixed Dialog).

Tanto a janela restrita como não-restrita pode incluir uma caixa do menu **Control**.

Os comandos disponíveis durante o tempo de execução dependem das configurações das propriedades relacionadas — por exemplo, configurar **MaxButton** e **MinButton** como **False** desativa os comandos **Maximize** e **Minimize** no menu **Control**, mas os comandos **Move** e **Close** permanecem disponíveis.

**Observação:** As configurações que você especifica para as propriedades **ControlBox**, **BorderStyle**, **MaxButton** e **MinButton** não são refletidas na aparência do formulário até o tempo de execução.

---

## Propriedade DrawWidth

Retorna ou configura a largura de linha para o resultado dos métodos gráficos.

### Sintaxe

*object.DrawWidth* [= *size*]

A propriedade **DrawWidth** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>Size</i>	Uma <u>expressão numérica</u> de 1 a 32.767. Este valor representa a largura de linha em <u>pixels</u> . O padrão é 1; ou seja, 1 pixel de largura.

### Comentários

Aumente o valor desta propriedade para aumentar a largura da linha. Se a configuração da propriedade **DrawWidth** for maior que 1, as configurações de 1 a 4 da propriedade **DrawStyle** produzem uma linha uniforme (o valor da propriedade **DrawStyle** não é alterado). Configurar **DrawWidth** como 1 permite que **DrawStyle** produza os resultados mostrados na tabela da propriedade **DrawStyle**.

## Propriedades FontBold, FontItalic, FontStriethru e FontUnderline

Retornam ou configuram estilos de fonte nos seguintes formatos: **Negrito**, *Itálico*, ~~Tachado~~ e Sublinhado.

**Observação:** As propriedades **FontBold**, **FontItalic**, **FontStriethru** e **FontUnderline** são incluídas para uso com o controle **CommonDialog** e para compatibilidade com versões anteriores do Visual Basic. Para funcionalidade adicional use as novas propriedades do objeto **Font** (não disponíveis para o controle **CommonDialog**).

### Sintaxe

*object.FontBold* [= *boolean*]

*object.FontItalic* [= *boolean*]

*object.FontStriethru* [= *boolean*]

*object.FontUnderline* [= *boolean*]

As sintaxes das propriedades **FontBold**, **FontItalic**, **FontStriethru** e **FontUnderline** têm estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Boolean</u> especificando o estilo de fonte descrito em Configurações.

### Configurações

As configurações de *boolean* são:

Configuração	Descrição
<b>True</b>	(Padrão para <b>FontBold</b> , exceto com o controle <b>CommonDialog</b> ) Ativa a formatação naquele estilo.
<b>False</b>	(Padrão para <b>FontItalic</b> , <b>FontStriethru</b> e <b>FontUnderline</b> , e <b>FontBold</b> com o controle <b>CommonDialog</b> ) Desativa a formatação naquele estilo.

### Comentários

Use estas propriedades de fonte para formatar textos durante o tempo de criação usando a janela Properties ou durante o tempo de execução usando código. Para controles **PictureBox**, **Form** e objetos **Printer**, a configuração destas propriedades não afeta elementos gráficos ou textos já desenhados no controle ou objeto. Para todos os outros controles, as alterações de fonte são imediatamente efetivas na tela.

Para usar estas propriedades com o controle **CommonDialog**, o sinalizador **Effects** deve ser configurado.

**Observação:** As fontes disponíveis no Visual Basic variam dependendo da configuração de seu



---

sistema, dispositivos de exibição e dispositivos de impressão. As propriedades relativas a fonte somente podem ser configuradas como valores para os quais efetivamente existam fontes. Em geral, você deve alterar a propriedade **FontName** antes de configurar os atributos de tamanho e estilo com as propriedades **FontSize**, **FontBold**, **FontItalic**, **FontStrikethru** e **FontUnderline**. Entretanto, quando configura fontes TrueType como menores que 8 pontos, você deve configurar o tamanho do ponto com a propriedade **FontSize**, configurar a propriedade **FontName** e então configurar o tamanho novamente com a propriedade **FontSize**. O ambiente operacional do Microsoft Windows usa uma fonte diferente para as fontes TrueType que sejam menores que 8 pontos.

## Propriedade FontName

Retorna ou configura a fonte usada para exibir texto em um controle ou em uma operação de desenho ou impressão durante o tempo de execução.

**Observação:** A propriedade **FontName** está incluída para uso com o controle **CommonDialog** e para compatibilidade com versões anteriores do Visual Basic. Para funcionalidade adicional, use as propriedades de objeto **Font** (não disponíveis para o controle CommonDialog).

### Sintaxe

*object*.**FontName** [= *font*]

A propriedade **FontName** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>font</i>	Uma <u>expressão de sequência de caracteres</u> especificando o nome da fonte a ser usada.

### Comentários

O padrão para esta propriedade é determinado pelo sistema. As fontes disponíveis com o Visual Basic variam dependendo da configuração de seu sistema, dispositivos de exibição e dispositivos de impressão. As propriedades relacionadas com fontes podem ser configuradas como valores para os quais existam fontes.

Em geral, você deve alterar **FontName** antes de configurar atributos de tamanho e estilo com as propriedades **FontSize**, **FontBold**, **FontItalic**, **FontStrikethru** e **FontUnderline**.

**Observação:** Durante o tempo de execução, você pode obter informações sobre fontes disponíveis para o sistema através das propriedades **FontCount** e **Fonts**.

---

## Propriedade FontSize

Retorna ou configura o tamanho da fonte a ser usada para o texto exibido em um controle ou em uma operação de desenho ou impressão durante o tempo de execução.

**Observação:** A propriedade **FontSize** é incluída para uso com o controle **CommonDialog** e para compatibilidade com versões anteriores do Visual Basic. Para funcionalidade adicional, use as novas propriedades do objeto **Font** (não disponíveis para o controle **CommonDialog**).

### Sintaxe

*object*.FontSize [= *points*]

A propriedade **FontSize** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>points</i>	Uma <u>expressão numérica</u> especificando o tamanho de fonte a ser usado em <u>pontos</u> .

### Comentários

Use esta propriedade para formatar texto no tamanho de fonte desejado. O padrão é determinado pelo sistema. Para alterar o padrão, especifique o tamanho da fonte em pontos.

O valor máximo para **FontSize** é 2160 pontos.

**Observação:** As fontes disponíveis com o Visual Basic variam dependendo da configuração de seu sistema, dispositivos de exibição e dispositivos de impressão. Propriedades relacionadas com fonte somente podem ser configuradas como valores para os quais existam fontes.

Em geral, você deve alterar a propriedade **FontName** antes de configurar os atributos de tamanho e estilo com as propriedades **FontSize**, **FontBold**, **FontItalic**, **FontStrikethru** e **FontUnderline**. Entretanto, quando você configura fontes TrueType como menores que 8 pontos, você deve configurar o tamanho do ponto com a propriedade **FontSize**, configurar a propriedade **FontName** e então configurar novamente o tamanho com a propriedade **FontSize**. O ambiente operacional do Microsoft Windows usa uma fonte diferente para fontes TrueType que sejam menores que 8 pontos.

## Propriedades Height e Width

Retornam ou configuram as dimensões de um objeto ou a largura do objeto **Columns** de um controle **DBGrid**. Para os objetos **Printer** e **Screen**, não estão disponíveis durante o tempo de criação.

### Sintaxe

*object*.Height [= *number*]

*object*.Width [= *number*]

As sintaxes das propriedades **Height** e **Width** têm estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Uma <u>expressão numérica</u> especificando as dimensões de um objeto, conforme descrito em Configurações.

### Configurações

As medidas são calculadas como abaixo:

- **Formulário** — a altura e largura externa do formulário, incluindo as bordas e a barra de título.
- **Controle** — medido a partir do centro da borda do controle, de modo que os controles com espessuras de borda diferentes alinhem-se corretamente. Estas propriedades usam as unidades de escala do recipiente do controle.
- Objeto **Printer** — as dimensões físicas da configuração de papel para o dispositivo de impressão; não disponível durante o tempo de criação. Se configurada durante o tempo de execução, os valores nestas propriedades são usados, ao invés da configuração da propriedade **PaperSize**.
- Objeto **Screen** — a altura e largura da tela; não disponível durante o tempo de criação e somente leitura durante o tempo de execução.

- Objeto **Picture** — a altura e largura da figura em unidades HiMetric.

### Comentários

Para objetos **Form**, **Printer** e **Screen**, estas propriedades são sempre medidas em twips. Para um formulário ou controle, os valores destas propriedades se alteram à medida que o objeto é dimensionado pelo usuário ou pelo código. Os limites máximos destas propriedades para todos os objetos dependem do sistema.

Se você configurar as propriedades **Height** e **Width** para um driver de impressora que não permite que estas propriedades sejam configuradas, não ocorrerá nenhum erro e o tamanho do papel permanecerá inalterado. Se você configurar **Height** e **Width** para um driver de impressora que permita que somente determinados valores sejam especificados, não ocorrerá nenhum erro e a propriedade é configurada para qualquer coisa que seja permitida pelo driver. Por exemplo, você poderia configurar **Height** como 150 e o driver a configuraria como 144.

Use as propriedades **Height**, **Width**, **Left** e **Top** para operações ou cálculos baseados na área total do objeto, como dimensionar ou mover o objeto. Use as propriedades **ScaleLeft**, **ScaleTop**, **ScaleHeight** e **ScaleWidth** para operações ou cálculos baseados em uma área interna do objeto, como desenhar ou mover objetos dentro de outro objeto.

**Observação:** A propriedade **Height** não pode ser alterada para o controle **DriveListBox** ou para o controle **ComboBox**, cuja configuração de propriedade **Style** é 0 (Dropdown Combo) ou 2 (Dropdown List).

Para o objeto **Columns** do controle **DBGrid**, **Width** é especificado na unidade de medida do objeto que contém o **DBGrid**. O valor padrão para **Width** é o valor da propriedade **DefColWidth** de **DBGrid**.

Para o objeto **Picture**, use os métodos **ScaleX** e **ScaleY** para converter unidades HiMetric na escala necessária.

## Propriedade Icon

Retorna o ícone exibido quando um formulário é minimizado durante o tempo de execução.

### Sintaxe

*object.Icon*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Use esta propriedade para especificar um ícone para qualquer formulário que o usuário possa minimizar durante o tempo de execução.

Por exemplo, você pode atribuir um ícone exclusivo a um formulário para indicar a função do formulário. Especifique o ícone carregando-o usando a janela Properties durante o tempo de criação. O arquivo que você carrega deve ter a extensão e formato de nome de arquivo .ico. Se você não especificar um ícone, o ícone padrão do Visual Basic para formulários será utilizado.

Você pode usar a Biblioteca de Ícones do Visual Basic (no subdiretório Icons) como uma origem para ícones. Quando criar um arquivo executável, você poderá atribuir um ícone ao aplicativo usando a propriedade **Icon** de qualquer formulário naquele aplicativo.

**Observação:** Você pode ver um ícone de formulário no Windows 95 no canto superior esquerdo do formulário ou quando o formulário é minimizado, tanto no Windows 95 quanto no Windows NT. Se o formulário for minimizado, a propriedade **BorderStyle** deverá ser configurada tanto como 1 (Fixed Single) quanto como 2 (Sizable) e, a propriedade **MinButton** deverá ser configurada como **True** para que o ícone seja visível.

Durante o tempo de execução, você pode atribuir uma propriedade **Icon** à propriedade **Icon** ou **DragIcon** de outro objeto. Você também pode atribuir um ícone retornado pela função **LoadPicture**. Usar **LoadPicture** sem um argumento atribui um ícone vazio (null) ao formulário, o que lhe permite desenhar no ícone durante o tempo de execução.

## Propriedade Interval

Retorna ou configura o número de milissegundos entre chamadas a um evento Timer de um controle **Timer**.

### Sintaxe

*object.Interval* [= *milliseconds*]

---

A propriedade **Interval** tem estas partes:

<b>Parte</b>	<b>Descrição</b>
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>milliseconds</i>	Uma <u>expressão numérica</u> especificando o número de milissegundos conforme descrito em Configurações.

### Configurações

As configurações para *milliseconds* são:

<b>Configuração</b>	<b>Descrição</b>
0	(Padrão) Desativa o controle <b>Timer</b> .
1 a 65.535	Configura um intervalo (em milissegundos) que se torna efetiva quando uma propriedade <b>Enabled</b> de um controle <b>Timer</b> for configurada como <b>True</b> . Por exemplo, um valor de 10.000 milissegundos é igual a 10 segundos. O máximo, 65.535 milissegundos é equivalente a pouco mais do que 1 minuto.

### Comentários

Você pode configurar a propriedade **Interval** de um controle **Timer** durante o tempo de criação ou durante o tempo de execução. Quando usar a propriedade **Interval**, lembre-se:

- A propriedade **Enabled** do controle **Timer** determina se o controle responde à passagem do tempo. Configure **Enabled** como **False** para desligar um controle **Timer** e como **True** para ativá-lo. Quando um controle **Timer** for ativado, sua contagem regressiva sempre se inicia a partir do valor de sua configuração de propriedade **Interval**.
- Crie um procedimento de evento Timer para informar ao Visual Basic o que fazer a cada vez que **Interval** decorreu.

---

## Propriedades Left e Top

- **Left** — retorna ou configura a distância entre a borda interna esquerda de um objeto e a borda esquerda de seu recipiente.
- **Top** — retorna ou configura a distância entre a borda interna superior e um objeto e a borda superior de seu recipiente.

### Sintaxe

*object.Left* [= *value*]

*object.Top* [= *value*]

As sintaxes das propriedades **Left** e **Top** têm estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Uma <u>expressão numérica</u> especificando a distância.

### Comentários

Para um formulário, as propriedades **Left** e **Top** são sempre expressas em twips; para um controle, elas são medidas em unidades que dependem do sistema de coordenadas de seu recipiente. Os valores para estas propriedades se alteram à medida que o objeto é movido pelo usuário ou por meio de código. Para os controles **CommonDialog** e **Timer**, estas propriedades não estão disponíveis durante o tempo de execução.

Para qualquer uma das propriedades, você pode especificar um número de precisão simples.

Use as propriedades **Left**, **Top**, **Height** e **Width** para operações baseadas em dimensões externas de um objeto, como mover ou redimensionar. Use as propriedades **ScaleLeft**, **ScaleTop**, **ScaleHeight** e **ScaleWidth** para operações baseadas em dimensões internas de um objeto como, por exemplo, desenhar ou mover objetos que estão contidos no objeto. As propriedades relativas a escala aplicam-se somente a controles PictureBox e a objetos **Form** e **Printer**.

## Propriedade List

Retorna ou configura os itens contidos na parte de listagem de um controle. A lista é uma matriz de seqüência de caracteres em que cada elemento é um item da lista. Disponível durante o tempo de criação para controles **ListBox** e **ComboBox** através do localizador de propriedade; somente leitura durante o tempo de execução para controles **DirListBox**, **DriveListBox** e **FileListBox**; para leitura e gravação durante o tempo de execução para controles **ComboBox** e **ListBox**.

### Sintaxe

*object.List(index)* [= *string*]

A propriedade **List** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	O número de um item específico na lista.
<i>String</i>	Uma <u>expressão de seqüência de caracteres</u> especificando o item da lista.

### Comentários

Use esta propriedade para acessar itens de lista.

Para todos os controles, exceto **DirListBox**, o índice do primeiro item é 0 e o índice do último item é **ListCount**-1.

Para um controle **DirListBox**, a seqüência de números de índice é baseada nos diretórios e sub-diretórios atuais quando o controle é criado durante o tempo de execução. O diretório que está atualmente expandido é representado usando o índice-1. Os diretórios acima do diretório atualmente expandido são representados por índices negativos com valores absolutos mais altos. Por exemplo, -2 é o diretório-pai do diretório atualmente expandido, e -3 é o diretório acima daquele. Os diretórios abaixo daquele atualmente expandido permanecem no intervalo 0 a **ListCount**-1.

Inicialmente, os controles **ComboBox** e **ListBox** contêm uma lista vazia. Para os controles do sistema de arquivos, a lista está baseada em condições que existem quando o controle é criado durante o tempo de execução:

- **DirListBox** — contém uma lista de diretórios, usando o intervalo -*n* até **ListCount**-1.

- 
- **DriveListBox** — contém a lista de conexões de unidades de disco válidas.
  - **FileListBox** — contém a lista de arquivos no diretório que está atualmente expandido e que correspondem à propriedade **Pattern**. O caminho não está incluído.

A propriedade **List** funciona em conjunto com as propriedades **ListCount** e **ListIndex**.

Para todos os controles aplicáveis, exceto um **DirListBox**, a enumeração de uma lista de 0 até **ListCount** -1 retorna todos os itens na lista. Para um controle **DirListBox**, a enumeração da lista de  $-n$  até **ListCount**-1 retorna uma lista contendo todos os diretórios e subdiretórios visíveis do diretório que está atualmente expandido. Neste caso  $n$  é o número de níveis de diretório acima do diretório atualmente expandido.

**Observação:** Para especificar os itens que você deseja exibir em um controle **ComboBox** ou **ListBox**, use o método **AddItem**. Para remover itens, use o método **RemoveItem**. Para manter os itens em ordem alfabética, configure a propriedade **Sorted** do controle como **True** antes de adicionar itens à lista.

Usar uma instrução `Option Base = 1` na seção `Declarations` não afeta a enumeração de elementos em controles do Visual Basic. O primeiro elemento é sempre 0.

Quando o índice **List** está fora do intervalo de entradas verdadeiras na caixa de listagem, é retornada uma seqüência de caracteres de comprimento zero (""). Por exemplo, `List(-1)` retorna uma seqüência de caracteres de comprimento zero para um controle **ComboBox** ou **ListBox**.

## Propriedade ListCount

Retorna o número de itens na parte de listagem de um controle.

### Sintaxe

*object*.**ListCount**

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

**ListCount** fornece informações específicas para cada controle:

- Controles **ComboBox** e **ListBox** — o número de itens na lista.
- Controle **DirListBox** — o número de subdiretórios no diretório atual.
- Controle **DriveListBox** — o número de conexões de unidade de disco.
- Controle **FileListBox** — o número de arquivos no diretório atual que correspondem à configuração da propriedade **Pattern**.

Se nenhum item for selecionado, o valor da propriedade **ListIndex** é -1. O primeiro item na lista é **ListIndex** = 0, e **ListCount** é sempre um a mais que o maior valor **ListIndex**.

---

## Propriedade ListIndex

Retorna ou configura o índice do item atualmente selecionado no controle. Não disponível durante o tempo de criação.

### Sintaxe

*object*.ListIndex [= *index*]

A propriedade **ListIndex** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	Uma <u>expressão numérica</u> especificando o índice do item atual, conforme descrito em Configurações.

### Configurações

As configurações de *index* são:

Configuração	Descrição
-1	(Padrão para controles <b>ComboBox</b> , <b>DirListBox</b> e <b>DriveListBox</b> ) Indica que nenhum item está selecionado atualmente. Para um controle <b>ComboBox</b> , indica que o usuário inseriu texto novo na parte de caixa de texto. Para um controle <b>DirListBox</b> , indica o <u>índice do caminho</u> atual. Para um controle <b>DriveListBox</b> , indica o índice da unidade de disco atual quando o controle é criado durante o <u>tempo de execução</u> .
<i>n</i>	(Padrão para controles <b>FileListBox</b> e <b>ListBox</b> ) Um número indicando o índice do item atualmente selecionado.

### Comentários

A expressão `List(List1.ListIndex)` retorna a seqüência de caracteres do item atualmente selecionado.

O primeiro item da lista é **ListIndex** = 0, e **ListCount** é sempre uma a mais que o maior valor **ListIndex**.

Para um controle onde os usuários podem efetuar seleções múltiplas, o comportamento desta propriedade depende do número de itens selecionados. Se somente um item for selecionado, **ListIndex** retorna o índice daquele item. Em uma seleção múltipla, **ListIndex** retorna o índice do item contido no retângulo do foco, estando ou não efetivamente selecionado o item.

## Propriedade MaxButton

Retorna um valor indicando se um formulário tem um botão Maximize.

### Sintaxe

*object*.MaxButton

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Configurações

As configurações da propriedade **MaxButton** são:

Configuração	Descrição
<b>True</b>	(Padrão) O formulário tem um botão <b>Maximize</b> .
<b>False</b>	O formulário não tem um botão <b>Maximize</b> .

### Comentários

Um botão **Maximize** permite aos usuários ampliar uma janela de formulário até um tamanho de tela inteira. Para exibir um botão **Maximize**, você também deve configurar a propriedade **BorderStyle** do formulário como 1 (Fixed Single), 2 (Sizable) ou 3 (Fixed Double).

Um botão **Maximize** torna-se automaticamente um botão **Restore** quando uma janela é maximizada. Minimizar ou restaurar uma janela altera automaticamente o botão **Restore** de volta para o

---

botão **Maximizar**.

As configurações que você especifica para as propriedades **MaxButton**, **MinButton**, **BorderStyle** e **ControlBox** não são refletidas na aparência do formulário até o tempo de execução.

**Observação:** Maximizar um formulário durante o tempo de execução gera um evento **Resize**. A propriedade **WindowState** reflete o estado atual da janela. Se você configurar a propriedade **WindowState** como 2 (Maximized), o formulário é maximizado independente de qualquer configuração válida para as propriedades **MaxButton** e **BorderStyle**.

## Propriedade MinButton

Retorna um valor indicando se um formulário tem um botão **Minimize**.

### Sintaxe

*object*.MinButton

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Valores de retorno

Os valores de retorno **MinButton** são:

Configuração	Descrição
<b>True</b>	(Padrão) O formulário tem um botão <b>Minimize</b> .
<b>False</b>	O formulário não tem um botão <b>Minimize</b> .

### Comentários

Um botão **Minimize** permite aos usuários minimizar uma janela de formulário como um ícone. Para exibir um botão **Minimize**, você precisa configurar a propriedade **BorderStyle** do formulário como 1 (Fixed Single), 2 (Sizable) ou 3 (Fixed Double).

As configurações que você especifica para as propriedades **MaxButton**, **MinButton**, **BorderStyle** e **ControlBox** não estão refletidas na aparência do formulário até o tempo de execução.

**Observação:** Minimizar um formulário como um ícone durante o tempo de execução gera um evento **Resize**. A propriedade **WindowState** reflete o estado atual da janela. Se você configurar a propriedade **WindowState** como 2 (Maximized), o formulário será maximizado independente de qualquer configuração válida para as propriedades **MaxButton** e **BorderStyle**.



---

## Propriedade Picture

Retorna ou configura um elemento gráfico a ser exibido em um controle. Para o controle de recipiente **OLE**, não disponível durante o tempo de criação e somente leitura durante o tempo de execução.

### Sintaxe

*object*.Picture [= *picture*]

A propriedade **Picture** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>picture</i>	Uma <u>expressão de sequência de caracteres</u> especificando o arquivo contendo o elemento gráfico, conforme descrito em Configurações.

### Configurações

As configurações de *picture* são:

Configuração	Descrição
(None)	(Padrão) Nenhuma figura.
(Bitmap, icon, metafile, GIF, JPEG)	Especifica um elemento gráfico. Você pode carregar o elemento gráfico a partir da <u>janela <b>Properties</b></u> durante o tempo de criação. Durante o tempo de execução, você também pode configurar esta propriedade usando a função <b>LoadPicture</b> em um <u>bitmap</u> , <u>ícone</u> ou <u>metarquivo</u> .

### Comentários

Durante o tempo de criação, você pode transferir um elemento gráfico com a **Área de transferência** usando os comandos **Copy**, **Cut** e **Paste** no menu **Edit**. Durante o tempo de execução, você pode usar os métodos Clipboard como, por exemplo **GetData**, **SetData** e **GetFormat** com as constantes não-texto de Clipboard **vbCFBitmap**, **vbCFMetafile** e **vbCFDIB**, que estão listadas na biblioteca de objetos do Visual Basic (VB) no Object Browser.

Ao configurar a propriedade **Picture** durante o tempo de criação, o elemento gráfico é salvo e carregado com o formulário. Se você criou um arquivo executável, o arquivo contém a imagem. Quando você carrega um elemento gráfico durante o tempo de execução, o elemento gráfico não é salvo com o aplicativo. Use a instrução **SavePicture** para salvar um elemento gráfico de um formulário ou caixa de figura para um arquivo.

**Observação:** Durante o tempo de execução, a propriedade **Picture** pode ser configurada como qualquer outra propriedade **DragIcon**, **Icon**, **Image** ou **Picture** do objeto, ou você pode atribuir-lhe o elemento gráfico retornado pela função **LoadPicture**.

## Propriedade Sorted

Retorna um valor indicando se os elementos de um controle forem automaticamente classificados em ordem alfabética.

### Sintaxe

*object*.Sorted

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Valores de retorno

Os valores de retorno da propriedade **Sorted** são:

Configuração	Descrição
<b>True</b>	Itens de lista são classificados por código de caractere.
<b>False</b>	(Padrão) Os itens da lista não são classificados em ordem alfabética.

### Comentários

---

Quando esta propriedade estiver configurada como True, o Visual Basic trata de quase todo o processamento necessário de seqüências de caracteres para manter a ordem alfabética, incluindo mudar os número de índice para itens conforme seja necessário através da adição ou remoção de itens.

**Observação:** Usar o método **AddItem** para adicionar um elemento a uma localização específica na lista pode violar a ordem de classificação, e adições subseqüentes podem não ser corretamente classificadas.

## Propriedade TabIndex

Retorna ou configura a ordem de tabulação da maioria dos objetos dentro de seu formulário-pai.

### Sintaxe

*object.TabIndex* [= *index*]

A propriedade **TabIndex** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	Um número inteiro de 0 a ( $n-1$ ), onde $n$ é o número de controles no formulário que têm uma propriedade <b>TabIndex</b> . Atribuir a <b>TabIndex</b> um valor menor que 0 gera um erro.

### Comentários

Como padrão, o Visual Basic atribui uma ordem de tabulação a controles, à medida que você os desenha em um formulário, com exceção dos controles **Menu**, **Timer**, **Data**, **Image**, **Line** e **Shape**, que não são incluídos na ordem de tabulação. Durante o tempo de execução, controles invisíveis ou desativados e controles que não podem receber o foco (controles **Frame** e **Label**) permanecem na ordem de tabulação, mas são ignorados durante a tabulação.

Cada novo controle é colocado em último lugar na ordem de tabulação. Se você alterar o valor da propriedade **TabIndex** de um controle para ajustar a ordem de tabulação padrão, o Visual Basic renumera automaticamente as **TabIndex** de outros controles para refletir inserções e exclusões. Você pode efetuar alterações durante o tempo de criação usando a janela Properties ou durante o tempo de execução por meio de código.

A propriedade **TabIndex** não é afetada pelo método **ZOrder**.

**Observação:** A ordem de tabulação de um controle não afeta sua tecla de acesso associada. Se você pressionar a tecla de acesso para um controle **Frame** ou **Label**, o foco se move até o próximo controle na ordem de tabulação que pode receber o foco.

Ao carregar formulários salvos como texto ASCII, os controles com uma propriedade **TabIndex** que não estão listados na descrição do formulário recebem automaticamente um valor **TabIndex**. Em controles carregados subseqüentemente, se valores **TabIndex** existente entram em conflito com valores anteriores atribuídos, os controles recebem automaticamente novos valores.

Quando exclui um ou mais controles, você pode usar o comando **Undo** para restaurar os controles e todas as suas propriedades, exceto a propriedade **TabIndex**, que não pode ser restaurada. **TabIndex** é reconfigurada ao final da ordem de tabulação quando você usa **Undo**.

## Propriedade Tag

Retorna ou configura uma expressão que armazena qualquer dado extra necessário para seu programa. Diferente de outras propriedades, o valor da propriedade **Tag** não é usado pelo Visual Basic; você pode usar esta propriedade para identificar objetos.

### Sintaxe

*object.Tag* [= *expression*]

A propriedade **Tag** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>expression</i>	Uma <u>expressão de seqüência de caracteres</u> identificando o objeto. O padrão é uma seqüência de ca-

---

racteres de comprimento zero ("").

### Comentários

Você pode usar esta propriedade para atribuir uma seqüência de caracteres de identificação a um objeto sem afetar qualquer de suas configurações de propriedade ou causar qualquer efeito colateral. A propriedade **Tag** é útil quando você precisa verificar a identidade de um controle ou objeto **MDIForm** que é passado como uma variável a um procedimento.

**Dica** Quando você cria uma nova ocorrência de um formulário, atribua um valor exclusivo à propriedade **Tag**.

## Propriedade Text

- Controle **ComboBox** (propriedade **Style** configurada como 0 [Dropdown Combo] ou como 1 [Simple Combo]) e o controle **TextBox** — retorna ou configura o texto contido na área de edição.
- Controle **ComboBox** (propriedade **Style** configurada como 2 [Dropdown List]) e controle **ListBox** — retornam o item selecionado na caixa de listagem; o valor retornado é sempre equivalente ao valor retornado pela expressão `List(ListIndex)`. Somente leitura durante o tempo de criação; somente leitura durante o tempo de execução.

### Sintaxe

*object.Text* [= *string*]

A propriedade **Text** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>string</i>	Uma <u>expressão de seqüência de caracteres</u> especificando texto.

### Comentários

Somente durante o tempo de criação, os padrões para a propriedade **Text** são:

- Controles **ComboBox** e **TextBox** — a propriedade **Name** do controle.
- Controle **ListBox** — uma seqüência de caracteres de comprimento zero ("").

Para uma **ComboBox** com a propriedade **Style** configurada como 0 (Dropdown Combo) ou como 1 (Simple Combo), ou para uma **TextBox**, esta propriedade é útil para ler a seqüência de caracteres efetiva contida na área de edição do controle. Para um controle **ComboBox** ou **ListBox** com a propriedade **Style** configurada como 2 (Dropdown List), você pode usar a propriedade **Text** para determinar o item atualmente selecionado.

A configuração **Text** para um controle **TextBox** está limitada a 2048 caracteres, a menos que a propriedade **MultiLine** seja definida como **True**, caso em que o limite é cerca de 32.

## Propriedade Value

- Controles **CheckBox** e **OptionButton** — retorna ou configura o estado do controle.
- Controle **CommandButton** — retorna ou configura um valor indicando se o botão foi escolhido; não disponível durante o tempo de criação.
- Objeto **Field** — retorna ou configura o conteúdo de um campo; não disponível durante o tempo de criação.
- Controles **HScrollBar** e **VScrollBar** (barras de rolagem horizontal e vertical) — retornam ou configuram a posição atual da barra de rolagem cujo valor de retorno está sempre entre os valores da propriedade **Max** e **Min**, inclusive.

### Sintaxe

*object.Value* [= *value*]

A propriedade **Value** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que é avaliada como um objeto na lista Applies To.

---

<i>value</i>	Valor especificando o estado, conteúdo ou posição de um controle, conforme descrito em Configurações.
--------------	---

### Configurações

As configurações de *value* são:

- Controle **CheckBox** — 0 é Não selecionada (padrão), 1 é Selecionada e 2 é Acinzentada (esmaecida).
- Controle **CommandButton** — **True** indica que o botão está selecionado; **False** (padrão) indica que o botão não está selecionado. Configurar a propriedade **Value** como **True** em código aciona o evento Clic do botão.
- Objeto **Field** — restrito apenas pelos tipos de dado do campo.
- Controles **HScrollBar** e **VScrollBar** — configura valores entre -32.768 e 32.767 para posicionar a caixa de rolagem.
- Controle **OptionButton** — **True** indica que o botão está selecionado; **False** (padrão) indica que o botão não está selecionado.

### Comentários

A propriedade padrão de um objeto é presumida e não precisa ser especificada em código. Por exemplo, **Field** é a propriedade padrão de qualquer **Recordset**, e **Value** é a propriedade padrão de um objeto **Field**. Isto torna as duas instruções abaixo equivalentes:

```
Dn.Fields("PubID").Value = X
```

```
Dn("PubID") = X
```

A primeira instrução *especifica* as propriedades padrão; a segunda instrução as *presume*.

---

## Propriedade Visible

Retorna ou configura um valor indicando se um objeto está visível ou oculto.

### Sintaxe

*object.Visible* [= *boolean*]

A propriedade **Visible** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que é avaliada como um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Boolean</u> especificando se o objeto é visível ou oculto.

### Configurações

As configurações para *boolean* são:

Configuração	Descrição
<b>True</b>	(Padrão) O objeto é visível.
<b>False</b>	O objeto está oculto.

### Comentários

Para ocultar um objeto na inicialização, configure a propriedade **Visible** como **False** durante o tempo de criação. Configurar esta propriedade em código permite ocultar e, posteriormente, reexibir um controle durante o tempo de execução em resposta a um determinado evento.

**Observação:** Usar o método **Show** ou **Hide** em um formulário é o mesmo que configurar a propriedade **Visible** do formulário em código como **True** ou **False**, respectivamente.

## Exemplo das propriedades BackColor, ForeColor

Este exemplo reconfigura de maneira aleatória as cores de segundo e primeiro planos, duas vezes por segundo para um formulário e controle **PictureBox**. Para experimentar este exemplo, cole o código na seção Declarations de um formulário que contenha um controle **PictureBox** e um controle **Timer** e, em seguida, pressione F5.

```
Private Sub Form_Load ()
    Timer1.Interval = 500
End Sub

Private Sub Timer1_Timer ()
    BackColor = QBColor(Rnd * 15)
    ForeColor = QBColor(Rnd * 10)
    Picture1.BackColor = QBColor(Rnd * 15)
    Picture1.ForeColor = QBColor(Rnd * 10)
End Sub
```

## Exemplo da propriedade BorderWidth

Este exemplo usa dois controles **ComboBox** para selecionar diferentes larguras e estilos para as bordas de um controle **Shape**. Para experimentar este exemplo, cole o código na seção Declarations de um formulário que contenha um controle **Shape** e um controle **ComboBox**. Para o **ComboBox**, configure **Style=2** e **Index=0** (para criar uma matriz de controle e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Load ()
    Combo1(0).Width = 1440 * 1.5
    Load Combo1(1)
    Combo1(1).Top = Combo1(0).Top + Combo1(0).Height * 1.5
    Combo1(1).Visible = True
    For I = 0 To 6
        Combo1(0).AddItem "BorderStyle = " & I
    Next I
    For I = 1 To 10
        Combo1(1).AddItem "BorderWidth = " & I
    Next I
    Combo1(0).ListIndex = 1
    Combo1(1).ListIndex = 0
End Sub
```

```

Private Sub Combol_Clic (Index As Integer)
    If Index = 0 Then
        Shapel.BorderStyle = Combol(0).ListIndex
    Else
        Shapel.BorderWidth = Combol(1).ListIndex + 1
    End If
End Sub

```

## Exemplo das propriedades Col, Row

Este exemplo coloca "Here" na célula atual e, em seguida, altera a célula ativa para a terceira célula na terceira linha e coloca "There" na célula. Para experimentar este exemplo, use a caixa de diálogo **Components** para adicionar um controle **MS Flex Grid** à caixa de ferramentas (no menu **Project**, escolha **Components** e, em seguida, selecione **Microsoft Flex Grid Control**) e, em seguida, desenhe uma grade em um novo formulário. Para executar o programa, pressione F5 e, em seguida, clique na grade.

```

Private Sub Form_Load ()
    MSFlexGrid1.Rows = 8 ' Configurar linhas e colunas.
    MSFlexGrid1.Cols = 5
End Sub

Private Sub MSFlexGrid1_Clic ()
    ' Colocar texto na célula atual.
    MSFlexGrid1.Text = "Here"
    ' Colocar texto na terceira linha, terceira coluna.
    MSFlexGrid1.Col = 2
    MSFlexGrid1.Row = 2
    MSFlexGrid1.Text = "There"
End Sub

```

O próximo exemplo exibe a localização da célula ativa e o intervalo da seleção à medida que um usuário seleciona uma célula ou intervalo de células. Observe que ao selecionar um intervalo, a célula ativa não se altera. Selecione um intervalo e, em seguida, clique no formulário para mover a célula ativa ao redor do perímetro da seleção. Observe que o intervalo selecionado não se altera.

Para experimentar este exemplo, crie um novo projeto, adicione um controle **MS Flex Grid** usando a caixa de diálogo **Components** (no menu **Project**, escolha **Components** e, em seguida, selecione **Microsoft Flex Grid Control**) e, em seguida, desenhe um **MS Flex Grid** e dois rótulos. Copie o código para a seção **Declarations** e, em seguida, pressione F5 para executar o programa.

```

Private Sub Form_Load ()
    MSFlexGrid1.Cols = 6 ' Configurar colunas e linhas.
    MSFlexGrid1.Rows = 7
End Sub

Private Sub MSFlexGrid1_RowColChange ()
    Msg = "Active cell: " & Chr(64 + MSFlexGrid1.Col)
    Mst = Msg & MSFlexGrid1.Row
    Label1.Caption = Msg
End Sub

Private Sub MSFlexGrid1_SelChange ()
    Msg = "Selection: " & Chr(64 + MSFlexGrid1.SelStartCol)
    Msg = Msg & MSFlexGrid1.SelStartRow
    Msg = Msg & ":" & Chr(64 + MSFlexGrid1.SelEndCol)
    Msg = Msg & MSFlexGrid1.SelEndRow
    Label2.Caption = Msg
End Sub

Private Sub Form_Clic ()
    ' Este procedimento move a célula ativa pelo
    ' perímetro do intervalo selecionado de
    ' células a cada clique no formulário.
    Dim GR, GC As Integer
    If MSFlexGrid1.Row = MSFlexGrid1.SelStartRow Then
        If MSFlexGrid1.Col = MSFlexGrid1.SelEndCol Then
            GR = 1: GC = 0

```

---

```

Else
    GR = 0: GC = 1
End If
ElseIf MSFlexGrid1.Row = MSFlexGrid1.SelEndRow Then
    If MSFlexGrid1.Col = MSFlexGrid1.SelStartCol Then
        GR = -1: GC = 0
    Else
        GR = 0: GC = -1
    End If
Else
    If MSFlexGrid1.Col = MSFlexGrid1.SelStartCol Then
        GR = -1: GC = 0
    Else
        GR = 1: GC = 0
    End If
End If
MSFlexGrid1.Row = MSFlexGrid1.Row + GR
MSFlexGrid1.Col = MSFlexGrid1.Col + GC
End Sub

```

## Exemplo da propriedade DrawWidth

Este exemplo desenha uma linha que se torna gradualmente mais espessa por um formulário. Para experimentar este exemplo, cole o código na seção Declarations de um formulário e, em seguida, pressione F5 e clique no formulário.

```

Private Sub Form_Click ()
    Dim I ' Declara variável.
    DrawWidth = 1 ' Configurar a largura inicial da caneta.
    PSet (0, ScaleHeight / 2) ' Configurar o ponto inicial.
    ForeColor = QBColor(5) ' Configurar a cor da caneta.
    For I = 1 To 100 Step 10 ' Configurar o loop.
        DrawWidth = I ' Reconfigura a largura da caneta.
        Line - Step(ScaleWidth / 10, 0) ' Desenhando uma linha.
    Next I
End Sub

```

## Exemplo das propriedades FontBold, FontItalic, FontStrikethru, FontUnderline

Este exemplo coloca texto em um formulário em uma de duas combinações de estilos a cada clique de mouse. Para experimentar este exemplo cole o código na seção Declarations de um formulário e, em seguida, pressione F5 e clique no formulário.

```

Private Sub Form_Click ()
    FontStrikethru = Not FontStrikethru ' Ligar e desliga o tachado.
    FontItalic = Not FontItalic ' Ligar e desligar o estilo de fonte.
    Print "Now is the time!" ' Imprimir algum texto.
End Sub

```

---

## Exemplo da propriedade FontName

Este exemplo imprime o nome de cada fonte usando uma determinada fonte. Para experimentar este exemplo cole o código na seção Declarations de um formulário. Pressione F5 para executar o programa e, em seguida, clique no formulário. Cada vez que você clica no formulário, o nome da fonte é impresso.

```
Private Sub Form_Clic ()
    Static I ' Declarar variáveis.
    Dim OldFont
    OldFont = FontName ' Preservar a fonte original.
    FontName = Screen.Fonts(I) ' Alterar para uma nova fonte.
    Print Screen.Fonts(I) ' Imprimir o nome da fonte.
    I = I + 1 ' Incrementar o contador.
    If I = FontCount Then I = 0 ' Reiniciar.
    FontName = OldFont ' Restaurar a fonte original.
End Sub
```

## Exemplo da propriedade FontSize

Este exemplo imprime texto em seu formulário em dois tamanhos diferentes de ponto a cada clique de mouse. Para experimentar este exemplo cole o código na seção Declarations de um formulário. Pressione F5 para executar o programa e, em seguida, clique no formulário.

```
Private Sub Form_Clic ()
    FontSize = 24 ' Configurar FontSize.
    Print "This is the 24-point type." ' Imprimir o tipo grande.
    FontSize = 8 ' Configurar FontSize.
    Print "This is the 8-point type." ' Imprimir tipo pequeno.
End Sub
```

## Exemplo das propriedades Height, Width

Este exemplo configura o tamanho de um formulário como 75 por cento do tamanho da tela e centraliza o formulário quando ele é carregado. Para experimentar este exemplo cole o código na seção Declarations de um formulário. Em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Clic ()
    Width = Screen.Width * .75 ' Configurar a largura do formulário.
    Height = Screen.Height * .75 ' Configurar a altura do formulário.
    Left = (Screen.Width - Width) / 2 ' Centralizar horizontalmente o
    formulário.
    Top = (Screen.Height - Height) / 2 ' Centralizar verticalmente o
    formulário.
End Sub
```

## Exemplo da propriedade Icon

Este exemplo cria um ícone em branco para um formulário e desenha pontos coloridos sobre o ícone, enquanto o formulário está minimizado. Para experimentar este exemplo cole o código na seção Declarations de um formulário e, em seguida, pressione F5 e minimize o formulário.

**Observação:** Este exemplo funciona somente com o Windows NT 3.5x.

```
Private Sub Form_Resize ()
    Dim X, Y ' Declarar variáveis.
    If Form1.WindowState = vbMinimized Then
        Form1.Icon = LoadPicture() ' Carregar um ícone em branco.
        Do While Form1.WindowState = vbMinimized
            ' Enquanto o formulário está minimizado,
            Form1.DrawWidth = 10 ' configurar o tamanho do ponto.
            ' Escolher a cor aleatória para o ponto.
            Form1.ForeColor = QBColor(Int(Rnd * 15))
            ' Configurar a localização aleatória no ícone.
            X = Form1.Width * Rnd
            Y = Form1.Height * Rnd
            PSet (X, Y) ' Desenhar o ponto no ícone.
            DoEvents ' Permitir outros eventos.
        Loop
    End If
```



---

```
End Sub
```

Este é o mesmo exemplo, exceto que ele usa o método **LoadPicture** para configurar a propriedade **Icon**. Este exemplo funciona com todas as versões do Windows:

```
Private Sub Form_Resize ()
    Dim X, Y ' Declarar variáveis.
    If Form1.WindowState = vbMinimized Then
        Form1.Icon = LoadPicture("c:\myicon.ico")
        ' Um ícone chamado "myicon.ico" deve existir no
        ' diretório c:\ para que este exemplo funcione
        ' corretamente.
    End If
End Sub
```

## Exemplo da propriedade Interval

Este exemplo permite ajustar a velocidade em que um formulário troca as cores. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha um controle **Timer**, um controle **HScrollBar** (barra de rolagem horizontal) e um controle **PictureBox** e, em seguida, pressione F5 e clique na barra de rolagem.

```
Private Sub Form_Load ()
    Timer1.Interval = 900 ' Configurar o intervalo.
    HScroll1.Min = 100 ' Configurar o mínimo.
    HScroll1.Max = 900 ' Configurar o máximo.
End Sub
Private Sub HScroll1_Change ()
    ' Configurar o intervalo de acordo com o valor da barra de rolagem.
    Timer1.Interval = 1000 - HScroll1.Value
End Sub
Private Sub Timer1_Timer ()
    ' Alternar BackColor entre vermelho e azul.
    If Picture1.BackColor = RGB(255, 0, 0) Then
        Picture1.BackColor = RGB(0, 0, 255)
    Else
        Picture1.BackColor = RGB(255, 0, 0)
    End If
End Sub
```

## Exemplo das propriedades Left, Top

Este exemplo configura o tamanho de um formulário como 75 por cento do tamanho da tela e centraliza o formulário quando ele é carregado. Para experimentar este exemplo, cole o código na seção Declarations de um formulário e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Click ()
    Width = Screen.Width * .75 ' Configurar a largura do formulário.
    Height = Screen.Height * .75 ' Configurar a altura do formulário.
    Left = (Screen.Width - Width) / 2 ' Centralizar o formulário horizontalmente.
    Top = (Screen.Height - Height) / 2 ' Centralizar o formulário verticalmente.
End Sub
```

---

## Exemplo da propriedade List

Este exemplo carrega um controle **ComboBox** com uma lista de nomes de sanduíches e exibe o primeiro item da lista. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha um controle **ComboBox** e, em seguida pressione F5.

```
Private Sub Form_Load ()
    Combol.AddItem "Denver Sandwich"      ' Adicionar cada item à lista.
    Combol.AddItem "Reuben Sandwich"
    Combol.AddItem "Turey Sandwich"
    Combol.Text = Combol.List(0)          ' Exibir o primeiro item.
End Sub
```

## Exemplo da propriedade ListCount

Este exemplo carrega uma lista de fontes de impressora em um controle **ComboBox**, exibe o primeiro item da lista, e imprime o número total de fontes. Cada clique no botão de comando altera todos os itens da lista para maiúsculas ou minúsculas. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha um controle **ComboBox (Style = 2)** e um controle **CommandButton** e, em seguida, pressione F5 e clique no **CommandButton**.

```
Private Sub Form_Load ()
    Dim I ' Declarar variável.
    AutoRedraw = True ' Configurar AutoRedraw.
    For I = 0 To Printer.FontCount - 1      ' Colocar nomes de fonte em
        lista.
        Combol.AddItem Printer.Fonts(I)
    Next I
    Combol.ListIndex = 0 ' Configurar o texto para o primeiro item.
    ' Imprimir informações de ListCount no formulário.
    Print "Number of printer fonts: "; Combol.ListCount
End Sub

Private Sub Command1_Click ()
    Static UpperCase
    Dim I ' Declarar a variável.
    For I = 0 To Combol.ListCount - 1      ' Executar loop pela lista.
        If UpperCase Then
            Combol.List(I) = UCase(Combol.List(I))
        Else
            Combol.List(I) = LCase(Combol.List(I))
        End If
    Next I
    UpperCase = Not UpperCase ' Alterar maiúsculas e minúsculas.
End Sub
```

## Exemplo da propriedade ListIndex

Este exemplo exibe os nomes de três jogadores em um controle **ListBox** e os salários correspondentes ao jogador selecionado em um controle **Label**. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha um controle **ComboBox** e um controle **Label** e, em seguida, pressione F5 e escolha um nome na **ComboBox**.

```
Dim Player(0 To 2) ' Dimensionar duas matrizes.
Dim Salary(0 To 2)
Private Sub Form_Load ()
    Dim I ' Declara variável.
    AutoSize = True
    Player(0) = "Miggey McMoo"           ' Inserir dados em matrizes.
    Player(1) = "Alf Hinshaw"
    Player(2) = "Woofers Dean"
    Salary(0) = "$234,500"
    Salary(1) = "$158,900"
    Salary(2) = "$1,030,500"
    For I = 0 To 2 ' Adicionar nomes a lista.
        Combol.AddItem Player(I)
    Next I
    Combol.ListIndex = 0 ' Exibir o primeiro item da lista.
End Sub
```

```
Private Sub Combol_Clic ()
    ' Exibir o salário correspondente ao nome.
    Labell1.Caption = Salary(Combol1.ListIndex)
End Sub
```

## Exemplo da propriedade Picture

Este exemplo carrega ícones da biblioteca de ícones do Visual Basic em dois entre três controles **PictureBox**. Quando você clica no formulário, a terceira **PictureBox** é usada para alternar os ícones. Você pode usar dois ícones quaisquer. Cole o código na seção Declarations de um formulário que contenha três pequenos controles **PictureBox** (para `Picture3`, configure **Visible** = **False**). Pressione F5 para executar o programa e, em seguida, clique no formulário.

```
Private Sub Form_Load ()
    ' Carregue os ícones.
    Picture1.Picture = LoadPicture("ICONS\COMPUTER\TRASH02A.ICO")
    Picture2.Picture = LoadPicture("ICONS\COMPUTER\TRASH02B.ICO")
End Sub

Private Sub Form_Clic ()
    ' Alterna os ícones.
    Picture3.Picture = Picture1.Picture
    Picture1.Picture = Picture2.Picture
    Picture2.Picture = Picture3.Picture
    ' Limpar a terceira figura (desnecessária se não for visível).
    Picture3.Picture = LoadPicture()
End Sub
```

Este exemplo cola um bitmap da **Área de transferência** em um controle **PictureBox**. Para localizar o valor de constantes de formato da **Área de transferência** (começando com **vbCF**), consulte a biblioteca de objetos Visual Basic (VB) no Object Browser. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha um controle **PictureBox**. Pressione F5 e, em seguida, em outro aplicativo, copie um ícone para a **Área de transferência**, alterne para o Visual Basic e clique no formulário.

```
Private Sub Form_Clic ()
    Picture1.Picture = Clipboard.GetData(vbCFDIB)
End Sub
```

## Exemplo da propriedade TabIndex

Este exemplo inverte a ordem de tabulação de um grupo de botões alterando a propriedade **TabIndex** de uma matriz de botões de comando. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha quatro controles **CommandButton**. Configure a propriedade **Name** como `CommandX` para cada botão para criar uma matriz de controle e, em seguida, pressione F5 e clique no formulário para inverter a ordem de tabulação dos botões.

```
Private Sub Form_Clic ()
    Dim I, X ' Declara variáveis.
    ' Inverter a ordem de tabulação configurando o valor inicial de X.
    If CommandX(0).TabIndex = 0 Then X = 4 Else X = 1
    For I = 0 To 3
        CommandX(I).Caption = X ' Configurar a legenda.
        CommandX(I).TabIndex = X - 1 ' Configurar a ordem de tabulação.
    Next I
    If CommandX(0).TabIndex = 3 Then
        X = X - 1 ' Decrementar X.
    Else
        X = X + 1 ' Incrementar X.
    End If
    Next I
End Sub
```

## Exemplo da propriedade Tag

Este exemplo exibe um ícone único para cada controle que está sendo arrastado. Para experimentar este exemplo cole o código na seção Declarations de um formulário contendo três con-

---

troles PictureBox. Configure a propriedade DragMode como 1 para Picture1 e Picture2 e, em seguida, pressione F5. Use o mouse para arrastar os controles Picture1 ou Picture2 sobre Picture3.

```
Private Sub Form_Load ()
    Picture1.Tag = "ICONS\ARROWS\POINT03.ICO"
    Picture2.Tag = "ICONS\ARROWS\POINT04.ICO"
End Sub

Private Sub Picture3_DragOver (Source As Control, X As Single, Y As Single, State As Integer)
    If State = vbEnter Then
        ' Selecionar com base na propriedade Name de cada PictureBox.
        Select Case Source.Name
            Case "Picture1"
                ' Carregar o ícone de Picture1.
                Source.DragIcon = LoadPicture(Picture1.Tag)
            Case "Picture2"
                ' Carregar o ícone de Picture2.
                Source.DragIcon = LoadPicture(Picture2.Tag)
        End Select
    ElseIf State = vbLeave Then
        ' Quando a origem não está sobre Picture3, descarregar o ícone.
        Source.DragIcon = LoadPicture ()
    End If
End Sub
```

## Exemplo da propriedade Text

Este exemplo ilustra a propriedade **Text**. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha três controle **TextBox** e um controle **CommandButton** e, em seguida, pressione F5 e digite texto em Text1.

```
Private Sub Text1_Change ()
    Text2.Text = LCase(Text1.Text) ' Exibir texto em minúsculas.
    Text3.Text = UCase(Text1.Text) ' Exibir texto em maiúsculas.
End Sub

Private Sub Command1_Click () ' Excluir texto.
    Text1.Text = ""
End Sub
```

## Exemplo da propriedade Value

Este exemplo exibe um valor numérico do controle **HScrollBar** (barra de rolagem horizontal) em um controle **TextBox**. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha um controle **TextBox** e um controle **HScrollBar**. Pressione F5 para executar o programa e, em seguida, clique na barra de rolagem.

```
Private Sub Form_Load ()
    HScroll1.Min = 0 ' Inicializar a barra de rolagem.
    HScroll1.Max = 1000
    HScroll1.LargeChange = 100
    HScroll1.SmallChange = 1
End Sub

Private Sub HScroll1_Change ()
    Text1.Text = Format (HScroll1.Value)
End Sub
```

## Exemplo da propriedade Visible

Este exemplo cria animação usando dois controle **PictureBox**. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha dois controles **PictureBox** do tamanho de ícones. Configure a propriedade **Name** como FileCab para ambos os controles **PictureBox** para criar uma matriz e, em seguida, pressione F5 e clique na figura para visualizar a animação.

```
Private Sub Form_Load ()
    Dim I ' Declara variável.
    FileCab(0).BorderStyle = 0 ' Configurar BorderStyle.
```

```

FileCab(1).BorderStyle = 0
' Carregar os ícones nas caixas de figura.
FileCab(1).Picture = LoadPicture("ICONS\OFFICE\FILES03B.ICO")
FileCab(0).Picture = LoadPicture("ICONS\OFFICE\FILES03A.ICO")
For I = 0 To 1
    FileCab(I).Move 400, 400      ' Posicionar elementos gráficos no
    mesmo ponto.
Next I
FileCab(1).Visible = False      ' Configurar como invisível.
FileCab(0).Visible = True      ' Configurar como visível.
End Sub

Private Sub FileCab_Clic (Index As Integer)
    Dim I ' Declara variável.
    For I = 0 To 1
        ' Alternar para visibilidade para ambos os elementos gráficos.
        FileCab(I).Visible = Not FileCab(I).Visible
    Next I
End Sub

```

## Propriedade Align

Retorna ou configura um valor que determina se um objeto é exibido de qualquer tamanho em qualquer lugar em um formulário, ou se ele é exibido na parte superior, inferior, esquerda ou direita do formulário, e é automaticamente dimensionado para ajustar-se à largura do formulário.

### Sintaxe

*object.Align* [= *number*]

A sintaxe da propriedade **Align** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Um número inteiro que especifica como um objeto é exibido, conforme descrito em Configurações.

### Configurações

As configurações de *number* são:

Constante	Configuração	Descrição
<b>vbAlignNone</b>	0	(Padrão em um formulário não-MDI) Nenhum — o tamanho e a localização pode ser configurado durante o tempo de criação ou em código. Esta configuração é ignorada se o objeto está em um <u>formulário MDI</u> .
<b>vbAlignTop</b>	1	(Padrão em um formulário MDI) Top — o objeto está em primeiro plano no formulário, e sua largura é igual à configuração da propriedade <b>ScaleWidth</b> do formulário.
<b>vbAlignBottom</b>	2	Inferior — o objeto está na parte inferior do formulário e sua largura é igual à configuração da propriedade <b>ScaleWidth</b> do formulário.
<b>vbAlignLeft</b>	3	Esquerda — o objeto está à esquerda do formulário e sua largura é igual à configuração da propriedade <b>ScaleWidth</b> do formulário.
<b>vbAlignRight</b>	4	Direita — o objeto está à direita do formulário e sua largura é igual à configuração da propriedade <b>ScaleWidth</b> do formulário.

## Comentários

Você pode usar a propriedade **Align** para criar rapidamente uma barra de ferramentas ou barra de status na parte superior ou na parte inferior de um formulário. À medida que um usuário altera o tamanho do formulário, um objeto com **Align** configurado como 1 ou 2 é automaticamente redimensionado para ajustar-se à largura do formulário.

Os controles **PictureBox** e **Data** são os únicos controles padrão que podem ser colocados em um formulário MDI. A área interna de um formulário MDI é definida pelo espaço não coberto por controles. Quando um formulário **MDI filho** é maximizado dentro de um formulário MDI pai, ele não cobrirá nenhum controle.

Use as configurações de *number* 3 e 4 para as alinhar barras de ferramentas à esquerda e à direita de um formulário ou formulário MDI. Se existirem duas barras de ferramentas em um canto de um formulário MDI, aquele alinhado na parte superior ou inferior estende-se até o canto, tendo preferência sobre aquele que é alinhado à esquerda ou à direita. Objetos alinhados à esquerda e à direita ocupam a área interna em um formulário MDI, exatamente como objetos alinhados pela parte superior e inferior.

## Propriedade Alignment

Retorna ou configura um valor que determina o alinhamento de um controle **CheckBox** ou **OptionButton**, texto em um controle, ou valores em uma coluna de um controle **DBGrid**. Somente leitura durante o tempo de execução para os controles **CheckBox**, **OptionButton** e **TextBox**.

### Sintaxe

*object.Alignment* [= *number*]

A sintaxe da propriedade **Alignment** tem estas partes:

Parte	Descrição
<i>object</i>	Uma expressão de objeto que avalia para um objeto na lista Applies To.
<i>Number</i>	Um inteiro que especifica o tipo de alinhamento, conforme descrito em Configurações.

### Configurações

Para os controles **CheckBox** e **OptionButton**, as configurações de *number* são:

Constante	Configuração	Descrição
<b>vbLeftJustify</b>	0	(Padrão) O texto é alinhado à esquerda; o controle é alinhado à direita.
<b>vbRightJustify</b>	1	O texto é alinhado à direita; o controle é alinhado à esquerda.

Para controles **Label** e **TextBox**, as configurações de *number* são:

Constante	Configuração	Descrição
<b>vbLeftJustify</b>	0	(Padrão) O texto é alinhado à esquerda.
<b>vbRightJustify</b>	1	O texto é alinhado à direita.
<b>vbCenter</b>	2	O texto é centralizado.

Para a coluna de um **DBGrid**, as configurações de *number* são:

Constante	Configuração	Descrição
<b>dbgLeft</b>	0	O texto é alinhado à esquerda.
<b>dbgRight</b>	1	O texto é alinhado à direita.
<b>dbgCenter</b>	2	O texto é centralizado.
<b>dbgGeneral</b>	3	(Padrão) General — o texto é alinhado à esquerda; os números são alinhados à direita.

### Comentários

Você pode exibir texto à direita ou à esquerda dos controles **OptionButton** e **CheckBox**. Como padrão, o texto é alinhado à esquerda.

A propriedade **MultiLine** em um controle **Textbox** deve ser configurada como **True** para que a propriedade **Alignment** funcione corretamente. Se a configuração da propriedade **MultiLine** de um controle **TextBox** for **False**, a propriedade **Alignment** é ignorada.

---

## Propriedades Archive, Hidden, Normal e System

Retorna ou configura um valor que determina se um controle **FileListBox** exibe arquivos com atributos Archive, Hidden, Normal ou System.

### Sintaxe

*object*.**Archive** [= *boolean*]

*object*.**Hidden** [= *boolean*]

*object*.**Normal** [= *boolean*]

*object*.**System** [= *boolean*]

A sintaxes das propriedades **Archive**, **Hidden**, **Normal** e **System** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> que especifica o tipo de arquivos exibidos, conforme descrito em Configurações.

### Configurações

As configurações de *boolean* são:

Configuração	Descrição
<b>True</b>	(Padrão para Archive e Normal) Exibe arquivos com o atributo de propriedade no controle <b>FileListBox</b> .
<b>False</b>	(Padrão para Hidden e System) Exibe arquivos sem o atributo de propriedade no controle <b>FileListBox</b> .

### Comentários

Use estas propriedades para especificar os tipos de arquivos a serem exibidos em um controle **FileListBox**, com base nos atributos padrão de arquivos usados no ambiente operacional. A configuração destas propriedades com código durante o tempo de execução reconfigura o controle **FileListBox** para exibir apenas aqueles arquivos com os atributos especificados.

Por exemplo, em uma operação de localização e substituição, você poderia exibir somente arquivos de sistema configurando a propriedade **System** como **True** e as outras propriedades como **False**. Ou, como parte de um procedimento de backup de arquivo, você poderia configurar a propriedade **Archive** como **True**, para listar apenas aqueles arquivos modificados desde o backup anterior.

## Propriedade AutoRedraw

Retorna ou configura o resultado de um método gráfico para um gráfico permanente.

### Sintaxe

*object*.**AutoRedraw** [= *boolean*]

A sintaxe da propriedade **AutoRedraw** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> que especifica como o objeto é regenerado, conforme descrito em Configurações.

### Configurações

As configurações de *boolean* são:

Configuração	Descrição
<b>True</b>	Ativa a regeneração automática de um objeto <b>Form</b> ou controle <b>PictureBox</b> . Os elementos gráficos e texto são gravados na tela e em uma imagem armazenada na memória. O objeto não recebe eventos Paint; ele é regenerado quando necessário, usando a imagem armazenada na memória.

---

<b>False</b>	(Padrão) Desativa a regeneração automática de um objeto e grava elementos gráficos ou texto somente na tela. O Visual Basic aciona o evento Paint do objeto quando necessário para regenerar o objeto.
--------------	--

### Comentários

Esta propriedade é fundamental no trabalho com os seguintes métodos gráficos: **Circle**, **Cls**, **Line**, **Point**, **Print** e **PSet**. A configuração de **AutoRedraw** como **True** automaticamente redesenha o resultado destes métodos em um objeto **Form** ou controle **PictureBox** quando, por exemplo, o objeto é redimensionado ou reexibido após ter sido ocultado por outro objeto.

Você pode configurar **AutoRedraw** no código durante o tempo de execução para alternar entre desenhar elementos gráficos permanentes (como, por exemplo, o segundo plano ou grade) e elementos gráficos temporários. Se você configura **AutoRedraw** como **False**, resultados anteriores tornam-se parte da tela de segundo plano. Quando **AutoRedraw** é configurada como **False**, os elementos gráficos de segundo plano não são excluídos se você limpa a área de desenho com o método **Cls**. A configuração de **AutoRedraw** novamente como **True** e, em seguida, o uso de **Cls** limpa os elementos gráficos de segundo plano.

**Observação:** Se você configura a propriedade **BacColor**, todos os elementos gráficos e texto são apagados, incluindo o elemento gráfico permanente. Em geral, todos os elementos gráficos devem ser exibidos usando o evento Paint, a menos que **AutoRedraw** seja configurada como **True**.

Para recuperar o gráfico permanente criado quando **AutoRedraw** for configurada como **True**, use a propriedade **Image**. Para passar o elemento gráfico permanente a uma API do Windows quando **AutoRedraw** for configurada como **True**, use a propriedade **hDC** do objeto.

Se você configura a propriedade **AutoRedraw** de um formulário como **False** e, em seguida, minimiza o formulário, as propriedades **ScaleHeight** e **ScaleWidth** são configuradas como o tamanho do ícone. Quando **AutoRedraw** for configurada como **True**, **ScaleHeight** e **ScaleWidth** permanecem no tamanho da janela restaurada.

Se **AutoRedraw** é configurada como **False**, o método **Print** imprime sobre os controles gráficos como, por exemplo, os controles **Image** e **Shape**.

## Propriedade AutoShowChildren

Retorna ou configura um valor que determina se os formulários MDI filho são exibidos ao serem carregados.

### Sintaxe

*object*.**AutoShowChildren** [= *boolean*]

A sintaxe da propriedade **AutoShowChildren** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> que especifica se os formulário MDI filho são automaticamente visíveis, conforme descrito em Configurações.

### Configurações

As configurações de *boolean* são:

Configuração	Descrição
<b>True</b>	(Padrão) Os formulários MDI filho são automaticamente exibidos ao serem carregados.
<b>False</b>	Formulários MDI filho não são automaticamente exibidos ao serem carregados.

### Comentários

Você pode usar a propriedade **AutoShowChildren** para carregar formulários MDI filho e deixá-los ocultos até que sejam exibidos usando o método **Show**.



---

## Propriedade AutoSize

Retorna ou configura um valor que determina se um controle é automaticamente redimensionado para exibir todo o seu conteúdo.

### Sintaxe

*object*.AutoSize [= *boolean*]

A sintaxe da propriedade **AutoSize** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	A <u>expressão Booleana</u> que especifica se o controle é redimensionado, conforme descrito em Configurações.

### Configurações

As configurações de *boolean* são:

Configuração	Descrição
<b>True</b>	Redimensiona automaticamente o controle para exibir todo o seu conteúdo.
<b>False</b>	(Padrão) Mantém o tamanho do controle constante. O conteúdo é recortado quando excede a área do controle.

## Propriedade ClipControls

Retorna ou configura um valor que determina se métodos gráficos em eventos Paint regeneram o objeto inteiro ou somente as áreas recém-exibidas. Também determina se o ambiente operacional do Microsoft Windows cria uma área de recorte que exclui controles não-gráficos contidos no objeto. Somente leitura durante o tempo de execução.

### Sintaxe

*object*.ClipControls

A sintaxe da propriedade **ClipControls** em estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> que especifica como os objetos são regenerados, conforme descrito em Configurações.

### Configurações

As configurações de *boolean*:

Configuração	Descrição
<b>True</b>	(Padrão) Métodos gráficos em eventos Paint regeneram o objeto inteiro. Uma área de recorte é criada ao redor de controles não-gráficos no formulário, antes de um evento Paint.
<b>False</b>	Métodos gráficos em eventos Paint regeneram somente as áreas recém-exibidas. Uma área de recorte não é criada ao redor de controles não-gráficos antes de um evento Paint. Formulários complexos, geralmente são carregados mais rapidamente quando <b>ClipControls</b> é configurada como <b>False</b> .

### Comentários

Recorte é o processo de determinação de quais partes de um formulário ou recipiente, por exemplo, os controle **Frame** ou **PictureBox**, são pintados quando o formulário é exibido. Um contorno

do formulário e controles é criado na memória. O ambiente operacional do Windows usa este contorno para pintar algumas partes como, por exemplo, o segundo plano, sem afetar outras partes, por exemplo, o conteúdo de um controle **TextBox**. Como a região de recorte é criada na memória, a configuração desta propriedade como **False** pode reduzir o tempo necessário para pintar ou repintar um formulário.

A região de recorte inclui a maioria dos controles, mas não recorta ao redor dos controles **Image**, **Label**, **Line** ou **Shape**.

Evite aninhar os controles intrínsecos com **ClipControls** configurada como **True** dentro de um controle com **ClipControls** configurada como **False** (por exemplo, um botão de comando dentro de uma caixa de figura). Este tipo de aninhamento de controle faz com que os controles se regenerem de maneira incorreta. Para corrigir este problema, configure a propriedade **ClipControls** tanto do controle de recipiente quanto dos controles aninhados como **True**.

## Propriedade ColorMode

Retorna ou configura um valor que determina se uma impressora em cores imprime o resultado em cores ou monocromaticamente. Não está disponível durante o tempo de criação.

### Sintaxe

*object.ColorMode* [= *value*]

A sintaxe da propriedade **ColorMode** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Uma constante ou número inteiro que especifica um modo de impressão, conforme descrito em Configurações.

### Configurações

As configurações de *value* são:

Configuração	Valor	Descrição
<b>vbPRCMMonochrome</b>	1	Imprime resultado monocromático (geralmente tons de preto e branco).
<b>vbPRCMColor</b>	2	Imprime resultado a cores.

### Comentários

O valor padrão depende do driver de impressora e das configurações atuais da impressora. Impressoras monocromáticas ignoram esta propriedade.

**Observação:** O efeito das propriedades do objeto **Printer** dependem do driver fornecido pelo fabricante da impressora. Algumas configurações de propriedade podem não ter efeito algum, ou diversas configurações de propriedades diferentes podem ter o mesmo efeito. Se você configura a propriedade **ColorMode** para uma impressora que não suporta cor, a configuração é ignorada. Se você tenta referir-se à propriedade **ColorMode**, entretanto, você obterá uma mensagem de erro. As configurações fora do intervalo aceito também podem produzir um erro. Para maiores informações, consulte a documentação do fabricante do driver específico.

## Propriedade Columns (ListBox)

Retorna ou configura um valor que determina se um controle **ListBox** rola vertical ou horizontalmente e como os itens nas colunas são exibidos. Se ele rola horizontalmente, a propriedade **Columns** determina quantas colunas são exibidas.

### Sintaxe

*object.Columns* [= *number*]

A sintaxe da propriedade **Columns** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Um número inteiro que especifica como um controle rola e como os itens são dispostos em colunas,

---

conforme descrito em Configurações.

### Configurações

As configurações de *number* são:

Configuração	Descrição
0	(Padrão) Os itens são dispostos em uma única coluna e o <b>ListBox</b> rola verticalmente.
1 to <i>n</i>	Os itens são dispostos em colunas do tipo jornalístico, preenchendo a primeira coluna, em seguida, a segunda coluna, e assim por diante. O <b>ListBox</b> rola horizontalmente e exibe o número especificado de colunas.

### Comentários

Para os controles **ListBox** de rolagem horizontal, a largura da coluna é igual à largura do **ListBox** dividido pelo número de colunas.

Esta propriedade não pode ser configurada como 0 ou alterada de 0 durante o tempo de execução — isto é, você não pode alterar um **ListBox** de colunas múltiplas para um **ListBox** de coluna única, ou um **ListBox** de coluna única para um **ListBox** de múltiplas colunas durante o tempo de execução. Entretanto, você pode alterar o número de colunas em um **ListBox** de colunas múltiplas durante o tempo de execução.

## Propriedade Copies

Retorna ou configura um valor que determina o número de cópias a serem impressas. Para o objeto **Printer**, não está disponível durante o tempo de criação.

### Sintaxe

*object.Copies* [= *number*]

A sintaxe da propriedade **Copies** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To
<i>number</i>	Uma <u>expressão numérica</u> que especifica o número de cópias a serem impressas. Este valor deve ser um número inteiro.

### Comentários

Para a caixa de diálogo **Print**, esta propriedade retorna o número de cópias inserido pelo usuário na caixa **Copies**. Se o sinalizador **cdIPDUseDevModeCopies** é configurado para o controle **CommonDialog**, esta propriedade sempre retorna 1.

Para o objeto **Printer**, cópias múltiplas podem ou não ser agrupadas, dependendo do driver da impressora. Cópias múltiplas do documento inteiro, ou cópias múltiplas de cada página podem ser impressas. Para impressoras que não suportam agrupamento, configure **Copies** = 1, e, em seguida, use um loop no código para imprimir cópias múltiplas do documento inteiro.

**Observação:** O efeito das propriedades do objeto **Printer** depende do driver fornecido pelo fabricante da impressora. Algumas configurações de propriedade não têm qualquer efeito, ou diversas configurações diferentes de propriedade podem ter o mesmo efeito. As configurações fora do intervalo aceito podem produzir um erro. Para maiores informações, consulte a documentação do fabricante para o driver específico.

---

## Propriedades CurrentX, CurrentY

Retorna ou configura as coordenadas horizontal (**CurrentX**) ou vertical (**CurrentY**) do próximo método de desenho ou impressão. Não está disponível durante o tempo de criação.

### Sintaxe

*object*.**CurrentX** [= *x*]  
*object*.**CurrentY** [= *y*]

A sintaxe das propriedades **CurrentX** e **CurrentY** têm estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>x</i>	Um número que especifica a coordenada horizontal.
<i>y</i>	Um número que especifica a coordenada vertical.

### Comentários

As coordenadas são medidas do canto superior esquerdo de um objeto. A configuração da propriedade **CurrentX** é 0 na borda esquerda de um objeto, e a configuração da propriedade **CurrentY** é 0 em sua borda superior. As coordenadas são expressas em twips, ou a unidade atual de medida definida pelas propriedades **ScaleHeight**, **ScaleWidth**, **ScaleLeft**, **ScaleTop** e **ScaleMode**.

Quando você usa os métodos gráficos abaixo, as configurações de **CurrentX** e **CurrentY** são alteradas conforme indicado:

Este método	Configura CurrentX, CurrentY como
<b>Circle</b>	O centro do objeto.
<b>Cls</b>	0, 0.
<b>EndDoc</b>	0, 0.
<b>Line</b>	O ponto final da linha.
<b>NewPage</b>	0, 0.
<b>Print</b>	A próxima posição de impressão.
<b>PSet</b>	O ponto desenhado.

## Propriedade Default

Retorna ou configura um valor que determina qual controle **CommandButton** é o botão de comando padrão em um formulário.

### Sintaxe

*object*.**Default** [= *boolean*]

A sintaxe da propriedade **Default** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> que especifica se o botão de comando é o padrão, conforme descrito em Configurações.

### Configurações

As configurações de *boolean* são:

Configuração	Descrição
<b>True</b>	O <b>CommandButton</b> é o botão de comando padrão.
<b>False</b>	(Padrão) O <b>CommandButton</b> não é o botão de comando padrão.

### Comentários

Somente um botão de comando em um formulário pode ser o botão de comando padrão. Quando **Default** é configurada como **True** para um botão de comando, ela é automaticamente configurada como **False** para todos os outros botões de comando no formulário. Quando a configuração da propriedade **Default** do botão de comando é **True** e seu formulário pai está ativo, o usuário pode escolher o botão de comando (acionando seu evento Clic) pressionando ENTER. Qualquer outro controle com o foco não recebe um evento de teclado (eyDown, eyPress ou eyUp) para a tecla ENTER, a menos que o usuário tenha movido o foco para outro botão de comando do mesmo for-

mulário. Neste caso, pressionar ENTER escolhe o botão de comando que tem o foco, ao invés do botão de comando padrão.

Para um formulário ou caixa de diálogo que suporte uma ação irreversível como, por exemplo, uma operação de exclusão, torna o botão **Cancel** o botão de comando padrão configurando sua propriedade **Default** como **True**.

Para controles de recipiente **OLE**, a propriedade **Default** é fornecida somente para aqueles objetos que se comportam especificamente como controles **CommandButton**.

## Propriedade DeviceName

Retorna o nome do dispositivo suportado por um driver.

### Sintaxe

*object*.**DeviceName**

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Cada driver de impressora suporta um ou mais dispositivos — por exemplo, HP LaserJet IIISi é um nome de dispositivo.

**Observação:** O efeito de propriedades do objeto **Printer** depende do driver fornecido pelo fabricante da impressora. Algumas configurações de propriedade podem não ter qualquer efeito, ou diversas configurações de propriedade diferentes podem ter todas o mesmo efeito. As configurações fora do intervalo aceitável podem produzir um erro. Para maiores informações, consulte a documentação do fabricante para o driver específico.

## Propriedade DragIcon

Retorna ou configura o ícone a ser exibido como ponteiro em uma operação de arrastar-e-soltar.

### Sintaxe

*object*.**DragIcon** [= *icon*]

A sintaxe da propriedade **DragIcon** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>icon</i>	Qualquer referência de código que retorna um ícone válido como, por exemplo, uma referência a um ícone de formulário ( <code>Form1.Icon</code> ), uma referência à propriedade <b>DragIcon</b> de outro controle ( <code>Text1.DragIcon</code> ), ou à função <b>LoadPicture</b> .

### Configurações

As configurações de *icon* são:

Configuração	Descrição
(nenhuma)	(Padrão) Um ponteiro de seta dentro de um retângulo.
Icon	Um ponteiro de mouse personalizado. Você especifica o ícone configurando-o na janela <b>Properties</b> durante o tempo de criação. Você também pode usar a função <b>LoadPicture</b> durante o tempo de execução. O arquivo que você carrega deve ter o formato e extensão de nome de arquivo .ico.

### Comentários

Você pode usar a propriedade **DragIcon** para oferecer retorno de informações visual durante uma operação de arrastar-e-soltar — por exemplo, para indicar que o controle de origem está sobre o destino correto. **DragIcon** começa a ter efeito quando o usuário inicia uma operação de arrastar-e-soltar. Geralmente, você configura **DragIcon** como parte de um procedimento de evento `MouseDown` ou `DragOver`.

**Observação:** Durante o tempo de execução, a propriedade **DragIcon** pode ser configurada como qualquer propriedade **DragIcon** ou **Icon** do objeto, ou você pode atribuir-lhe um ícone de

tornado pela função **LoadPicture**.

Quando você configura a propriedade **DragIcon** durante o tempo de execução atribuindo a propriedade **Picture** de um controle à propriedade **DragIcon** de outro controle, a propriedade **Picture** deve conter um arquivo .ico, não um arquivo .bmp.

## Propriedade DragMode

Retorna ou configura um valor que determina se o modo de arraste automático ou manual é usado para uma operação arrastar-e-soltar.

### Sintaxe

*object*.**DragMode** [= *number*]

A sintaxe da propriedade **DragMode** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Um número inteiro que especifica o modo de arraste, conforme descrito em Configurações.

### Configurações

As configurações de *number* são:

Constante	Configuração	Descrição
<b>vbManual</b>	0	(Padrão) Manual — exige o uso do método <b>Drag</b> para iniciar uma operação arrastar-e-soltar no controle de origem.
<b>vbAutomatic</b>	1	Automático — clicar o controle de origem inicia automaticamente uma operação arrastar-e-soltar. Controles de recipientes <b>OLE</b> são automaticamente arrastados somente quando não têm o foco.

### Comentários

Quando **DragMode** é configurada como 1 (Automática), o controle não responde a eventos normais do mouse. Use a configuração 0 (Manual) para determinar quando uma operação arrastar-e-soltar inicia ou termina; você pode usar esta configuração para iniciar uma operação arrastar-e-soltar em resposta a um comando de menu ou de teclado ou para ativar um controle de origem para reconhecer um evento MouseDown antes de uma operação arrastar-e-soltar.

Clicar enquanto o ponteiro do mouse está sobre um objeto de destino durante uma operação arrastar-e-soltar gera um evento DragDrop para o objeto de destino. Isto encerra a operação arrastar-e-soltar. Uma operação arrastar-e-soltar também pode gerar um evento DragOver.

**Observação:** Enquanto um controle está sendo arrastado, ele não pode reconhecer outros eventos de teclado ou mouse iniciados pelo usuário (eyDown, eyPress ou eyUp, MouseDown, MouseMove ou MouseUp). Entretanto, o controle pode receber eventos iniciados por código ou por um vínculo DDE.

## Propriedade DrawMode

Retorna ou configura um valor que determina a aparência de resultado de métodos gráficos ou a aparência de um controle **Shape** ou **Line**.

### Sintaxe

*object*.**DrawMode** [= *number*]

A sintaxe da propriedade **DrawMode** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Um número inteiro que especifica a aparência, con-

forme descrito em Configurações.

### Configurações

As configurações de *number* são:

Constante	Configuração	Descrição
<b>vbBlacness</b>	1	Tela preta.
<b>vbNotMergePen</b>	2	Not Merge <u>Pen</u> — Inverso da configuração 15 (Merge Pen).
<b>vbMasNotPen</b>	3	Mas Not Pen — Combinação das cores comuns ao segundo plano e o inverso da caneta.
<b>vbNotCopyPen</b>	4	Not Copy Pen — Inverso da configuração 13 (Copy Pen).
<b>vbMasPenNot</b>	5	Mas Pen Not — Combinação das cores comuns tanto à caneta quanto ao inverso da exibição.
<b>vbInvert</b>	6	Invert — Inverso da cor de exibição.
<b>vbXorPen</b>	7	Xor Pen — Combinação das cores na caneta e na cor de exibição, mas não em ambos.
<b>vbNotMasPen</b>	8	Not Mas Pen — Inverso da configuração 9 (Mas Pen).
<b>vbMasPen</b>	9	Mas Pen — Combinação das cores comuns tanto à caneta quanto à exibição.
<b>vbNotXorPen</b>	10	Not Xor Pen — Inverso da configuração 7 (Xor Pen).
<b>vbNop</b>	11	Nop — Nenhuma operação — o resultado permanece inalterado. Na realidade, esta configuração desativa o desenho.
<b>vbMergeNotPen</b>	12	Merge Not Pen — Combinação da cor de exibição e o inverso da cor da caneta.
<b>vbCopyPen</b>	13	Copy Pen (Padrão) — Cor especificada pela propriedade <b>ForeColor</b> .
<b>vbMergePenNot</b>	14	Merge Pen Not — Combinação da cor da caneta e o inverso da cor de exibição.
<b>vbMergePen</b>	15	Merge Pen — Combinação da cor da caneta e da cor de exibição.
<b>vbWhiteness</b>	16	Tela branca.

### Comentários

Use esta propriedade para produzir efeitos visuais com os controles **Shape** ou **Line** ou quando desenha com os métodos gráficos. O Visual Basic compara cada pixel no padrão de desenho ao pixel correspondente no segundo plano existente e, em seguida, aplica operações voltadas para bit. Por exemplo, a configuração 7 (Xor Pen) usa o operador **Xor** para combinar um pixel de padrão de desenho a um pixel de segundo plano.

O efeito exato de uma configuração **DrawMode** depende da maneira como a cor de uma linha desenhada durante o tempo de execução combina-se as cores já na tela. As configurações 1,5,7,11,13 e 16 produzem os resultados mais previsíveis.

## Propriedade DrawStyle

Retorna ou configura um valor que determina o estilo de linha para resultado de métodos gráficos.

### Sintaxe

*object*.**DrawStyle** [= *number*]

A sintaxe da propriedade **DrawStyle** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Um número inteiro que especifica um estilo de linha, conforme descrito em Configurações.

### Configurações

As configurações de *number* são:

Constante	Configuração	Descrição
<b>vbSolid</b>	0	(Padrão) Uniforme
<b>vbDash</b>	1	Traço
<b>vbDot</b>	2	Ponto
<b>vbDashDot</b>	3	Traço-ponto
<b>vbDashDotDot</b>	4	Traço-ponto-ponto
<b>vbInvisible</b>	5	Transparente
<b>vbInsideSolid</b>	6	Uniforme interna

### Comentários

Se **DrawWidth** for configurada como um valor maior que 1, as configurações de **DrawStyle** 1 até 4 produzem uma linha uniforme (o valor da propriedade **DrawStyle** não se altera). Se **DrawWidth** for configurada como 1, **DrawStyle** produz o efeito descrito na tabela anterior para cada configuração.

## Propriedade Drive

Retorna ou configura a unidade de disco selecionada durante o tempo de execução. Não está disponível durante o tempo de criação.

### Sintaxe

*object*.**Drive** [= *drive*]

A sintaxe da propriedade **Drive** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>drive</i>	Uma <u>expressão de seqüência de caracteres</u> que especifica a unidade de disco selecionada.

### Comentários

As unidades de disco válidas para a propriedade **Drive** incluem todas as unidades de disco presentes em ou conectadas ao sistema quando o controle é criado e atualizado durante o tempo de execução. A configuração padrão da propriedade **Drive** é a unidade de disco atual.

Ao ler esta configuração de propriedade, a unidade de disco selecionada é retornada em um dos formatos abaixo:

- Disquetes — "a:" ou "b:", e assim por diante
- Meio fixo — "c: [*volume id*]"
- Conexões de rede — "x: \\servidor\compartilhamento"

Ao configurar esta propriedade:

- Somente o primeiro caractere da seqüência de caracteres é significativo (a seqüência de caracteres não distingue entre maiúsculas e minúsculas).
- A alteração da configuração para a propriedade **Drive** aciona um evento Change.
- Selecionar uma unidade que não está presente causa um erro.
- A configuração desta propriedade também regenera a lista de unidades de disco, oferecendo uma maneira no código de rastrear conexões de rede adicionados desde que o controle foi criado.

Se a propriedade **FileName** for configurada como um caminho de rede qualificado sem uma unidade de disco de destino, o valor da propriedade **Drive** é uma seqüência de caracteres de comprimento zero (" "), nenhuma unidade de disco está selecionada e a configuração da propriedade **ListIndex** é 1.

**Observação:** A propriedade **Drive** retorna um valor diferente para a propriedade **ListIndex** que retorna a seleção da caixa de listagem.



---

## Propriedade DriverName

Retorna o nome do driver para um objeto **Printer**.

### Sintaxe

*object.DriverName*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Cada driver tem um nome exclusivo. Por exemplo, o **DriverName** para diversas impressoras Hewlett-Packard é HPPCL5MS. O **DriverName** é geralmente o nome de arquivo do driver sem uma extensão.

**Observação:** O efeito sobre as propriedades do objeto **Printer** depende do driver fornecido pelo fabricante da impressora. Algumas configurações de propriedade podem não ter efeito algum, ou diversas configurações diferentes de propriedade podem ter todas o mesmo efeito. As configurações fora do intervalo aceito podem produzir um erro. Para maiores informações, consulte a documentação do fabricante para o driver específico.

## Propriedade Duplex

Retorna ou configura um valor que determina se uma página é impressa em ambos os lados (se a impressora suporta este recurso). Não está disponível durante o tempo de criação.

### Sintaxe

*object.Duplex* [= *value*]

A sintaxe da propriedade **Duplex** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Um valor ou constante que especifica o tipo de impressão, conforme descrito em Configurações.

### Configurações

As configurações de *value* são:

Constante	Valor	Descrição
<b>vbPRDPSimplex</b>	1	Impressão em apenas uma face com a configuração atual de orientação.
<b>vbPRDPHorizontal</b>	2	Impressão em dupla face usando um giro de página horizontal.
<b>vbPRDPVertical</b>	3	Impressão em dupla face usando um giro de página vertical.

### Comentários

Com uma impressão duplex horizontal, a parte superior de ambos os lados da página estão na mesma extremidade da página. Com a impressão duplex vertical, a parte inferior de uma página está na mesma extremidade da folha que a parte superior da próxima página. O diagrama abaixo ilustra a impressão duplex horizontal e vertical:

{bmc bm4.BMP}

**Observação:** O efeito das propriedades do objeto **Printer** depende do driver fornecido pelo fabricante da impressora. Algumas configurações de propriedade podem não ter qualquer efeito, ou diversas configurações de propriedade diferentes podem ter todas o mesmo efeito. As configurações fora do intervalo aceito podem produzir um erro. Para maiores informações, consulte a documentação do fabricante para o driver específico.

---

## Propriedade FileName

Retorna ou configura o caminho e nome de arquivo de um arquivo selecionado. Não está disponível durante o tempo de criação para o controle **FileListBox**.

### Sintaxe

*object.FileName* [= *pathname*]

A sintaxe da propriedade **FileName** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>pathname</i>	Uma <u>expressão de seqüência de caracteres</u> que especifica o caminho e nome de arquivo.

### Comentários

Quando você cria o controle durante o tempo de execução, a propriedade **FileName** é configurada como uma seqüência de caracteres de comprimento zero (""), significando que nenhum arquivo está selecionado atualmente.

No controle **CommonDialog**, você pode configurar a propriedade **FileName** antes de abrir uma caixa de diálogo para configurar o nome de arquivo inicial.

Ler esta propriedade retorna o nome de arquivo atualmente selecionado na lista. O caminho é recuperado separadamente, usando a propriedade **Path**. O valor é funcionalmente equivalente a **List(ListIndex)**. Se nenhum arquivo estiver selecionado, a **FileName** retorna uma seqüência de caracteres de comprimento zero.

Ao configurar esta propriedade:

- Incluir uma unidade de disco, caminho ou padrão na seqüência de caracteres altera a configuração das propriedades **Drive**, **Path** e **Pattern** de maneira correspondente.
- Incluir o nome de um arquivo existente (sem caracteres curinga) na seqüência de caracteres provoca a seleção do arquivo.
- Alterar o valor desta propriedade também pode provocar um ou mais destes eventos: **PathChange** (se você alterar o caminho), **PatternChange** (se você alterar o padrão) ou **DbtClic** (se você atribuir um nome de arquivo existente).
- Esta configuração de propriedade pode ser um caminho de rede qualificado e um nome de arquivo usando a sintaxe abaixo:

\\servername\sharename\pathname

## Propriedade FillColor

Retorna ou configura a cor usada para preencher formulários; **FillColor** também é usada para preencher círculos e caixas criadas como os métodos gráficos **Circle** e **Line**.

### Sintaxe

*object.FillColor* [= *value*]

A sintaxe da propriedade **FillColor** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Um valor ou constante que determina a cor de preenchimento, conforme descrito em Configurações.

### Configurações

As configurações de *value* são:

Configuração	Descrição
Cores RGB normais	As cores configuradas com as funções <b>RGB</b> ou <b>QBColor</b> no código.
Cores padrão do sistema	As cores especificadas com <u>constantes</u> de cor do sistema na biblioteca de objetos Visual Basic (VB) no Object Browser. O ambiente operacional do Microsoft Windows substitui as alternativas do usuário, conforme especificado pelas configurações de <b>Painel de controle</b> do usuário.

---

Como padrão, **FillColor** é configurada como 0 (Preto).

### Comentários

Exceto pelo objeto **Form**, quando a propriedade **FillStyle** está configurada como padrão, 1 (Transparente), a configuração **FillColor** é ignorada.

## Propriedade FillStyle

Retorna ou configura o padrão usado para preencher os controles Shape, assim como, círculos e caixas criadas com os métodos gráficos **Circle** e **Line**.

### Sintaxe

*object.FillStyle* [= *number*]

A sintaxe da propriedade **FillStyle** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Um número inteiro que especifica o estilo de preenchimento, conforme descrito em Configurações.

### Configurações

As configurações de *number* são:

Constante	Configuração	Descrição
<b>vbFSSolid</b>	0	Uniforme
<b>vbFSTransparent</b>	1	(Padrão) Transparente
<b>vbHorizontalLine</b>	2	Linha Horizontal
<b>vbVerticalLine</b>	3	Linha Vertical
<b>vbUpwardDiagonal</b>	4	Diagonal para cima
<b>vbDownwardDiagonal</b>	5	Diagonal para baixo
<b>vbCross</b>	6	Cruzado
<b>vbDiagonalCross</b>	7	Cruzado diagonal

### Comentários

Quando **FillStyle** é configurada como 1 (Transparente), a propriedade **FillColor** é ignorada, exceto pelo objeto **Form**.

---

## Propriedade FontCount

Retorna o número de fontes disponíveis para o dispositivo de exibição atual ou impressora ativa.

### Sintaxe

*object*.FontCount

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Use esta propriedade com a propriedade Fonts para visualizar uma lista de telas disponíveis e fontes de impressora. As fontes disponíveis no Visual Basic variam de acordo com a configuração de seu sistema, dispositivos de exibição e dispositivos de impressão.

## Propriedade Fonts

Retorna todos os nomes de fonte disponíveis para o dispositivo de exibição atual ou impressora ativa.

### Sintaxe

*object*.Fonts(*index*)

A sintaxe da propriedade **Fonts** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	Um número inteiro de 0 a <b>FontCount</b> - 1.

### Comentários

A propriedade **Fonts** funciona em conjunto com a propriedade **FontCount** que retorna o número de nomes de fonte disponíveis para o objeto. As fontes disponíveis no Visual Basic variam de acordo com a configuração de seu sistema, dispositivos de exibição e dispositivos de impressão. Use ambas as propriedades **Fonts** e **FontCount** para obter informações sobre fontes de tela ou impressora disponíveis.

## Propriedade hDC

Retorna um identificador fornecido pelo ambiente operacional do Microsoft Windows ao contexto de dispositivo de um objeto.

### Sintaxe

*object*.hDC

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Esta propriedade é um identificador de contexto de dispositivo de ambiente operacional do Windows. O ambiente operacional do Windows gerencia a exibição do sistema atribuindo um contexto de dispositivo para o objeto **Printer** e para cada formulário e controle **PictureBox** em seu aplicativo. Você pode usar a propriedade **hDC** para referir-se ao identificador de contexto de dispositivo de um objeto. Este fornece um valor a ser passado a chamadas de API do Windows.

Com um controle **CommonDialog**, esta propriedade retorna um contexto de dispositivo para a impressora selecionada na caixa de diálogo **Print** quando o sinalizador **cdlReturnDC** é configurado, ou um contexto de informações quando o sinalizador **cdlReturnIC** é configurado.

**Observação:** O valor da propriedade **hDC** pode alterar enquanto um programa está sendo executado, portanto não armazena o valor em uma variável; ao invés, use a propriedade **hDC** cada vez que você precisar dela.

A propriedade **AutoRedraw** pode provocar a mudança da propriedade **hDC**. Se **AutoRedraw** for configurada como **True** para um formulário ou recipiente **PictureBox**, **hDC** funciona como um identificador para o contexto de dispositivo de um elemento gráfico permanente (equivalente à propriedade **Image**). Quando **AutoRedraw** for **False**, **hDC** é o valor efetivo de **hDC** da janela **Form** ou do recipiente **PictureBox**. A configuração da propriedade **hDC** pode alterar-se, enquanto o programa está sendo executado, independente da configuração de **AutoRedraw**.

---

## Propriedade HideSelection

Retorna um valor que determina se o texto selecionado aparece realçado quando o controle perde o foco.

### Sintaxe

*object.HideSelection*

O espaço reservado *object* representa uma expressão de objeto que avalia como um objeto na lista Applies To.

### Valores de retorno

Os valores de retorno da propriedade **HideSelection** são:

Value	Descrição
<b>True</b>	(Padrão) O texto selecionado não aparece realçado quando o controle perde o foco.
<b>False</b>	O texto selecionado aparece realçado quando o controle perde o foco.

### Comentários

Você pode usar esta propriedade para indicar qual texto é realçado, enquanto outro formulário ou uma caixa de diálogo tem o foco — por exemplo, em uma rotina de verificação ortográfica.

## Propriedade hWnd

Retorna um identificador a um formulário ou controle.

**Observação:** Esta propriedade não é suportada para o controle do recipiente **OLE**.

### Sintaxe

*object.hWnd*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To

### Comentários

O ambiente operacional do Microsoft Windows identifica cada formulário e controle em um aplicativo atribuindo-lhe um identificador, ou **hWnd**. A propriedade **hWnd** é usada com chamadas API do Windows. Muitas funções do ambiente operacional do Windows exigem o **hWnd** da janela ativa como argumento.

**Observação:** Como o valor desta propriedade pode se alterar enquanto o programa é executado, nunca armazene o valor **hWnd** em uma variável.

---

## Propriedade Image

Retorna um identificador a um elemento gráfico permanente; o identificador é fornecido pelo ambiente operacional do Microsoft Windows.

### Sintaxe

*object.Image*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

A propriedade **AutoRedraw** de um objeto determina se a regeneração de um objeto ocorre com um elemento gráfico permanente, ou através de eventos Paint. O ambiente operacional do Windows identifica o elemento gráfico permanente de um objeto atribuindo-lhe um identificador; você pode usar a propriedade **Image** para obter este identificador.

Um valor **Image** existe independente da configuração da propriedade **AutoRedraw**. Se **AutoRedraw** for **True**, e nada tiver sido desenhado, a imagem exibe apenas a cor definida pela propriedade **BackColor** e a figura.

Você pode atribuir o valor de **Image** à propriedade **Picture**. A propriedade **Image** também oferece um valor a ser passado a chamadas API do Windows.

As propriedades **Image**, **DragIcon** e **Picture** são normalmente usadas ao se atribuir valores a outras propriedades, ao salvar com a instrução **SavePicture** ou ao colocar alguma coisa na **Área de transferência**. Você não pode atribuí-los a uma variável temporária, diferente do tipo de dados **Picture**.

A propriedade **AutoRedraw** pode fazer com que **Image**, que é um identificador se altere. Quando **AutoRedraw** é **True**, a propriedade **hDC** de um objeto torna-se um identificador de um contexto de dispositivo que contém o bitmap retornado por **Image**.

## Propriedade ItemData

Retorna ou configura um número específico para cada item em um controle **ComboBox** ou **ListBox**.

### Sintaxe

*object.ItemData(index) [= number]*

A sintaxe da propriedade **ItemData** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	O número de um item específico no objeto.
<i>Number</i>	O número a ser associado ao item específico.

### Comentários

A propriedade **ItemData** é uma matriz de valores inteiros longos com o mesmo número de itens que uma propriedade **List** de um controle. Você pode usar os números associados a cada item para identificar os itens. Por exemplo, você pode usar o número de identificação de funcionário para identificar cada nome de funcionário em um controle **ListBox**. Ao preencher o **ListBox**, preencha também os elementos correspondentes na matriz **ItemData** com os números de funcionários.

A propriedade **ItemData** é com frequência usada como índice para uma matriz de estruturas de dados associadas a itens em um controle **ListBox**.

**Observação:** Quando você insere um item em uma lista com o método **AddItem**, um item também é automaticamente inserido na matriz **ItemData**. Entretanto, o valor não é reinicializado como zero; ele retém o valor que estava naquela posição, antes que você adicionasse o item à lista. Quando você usa a propriedade **ItemData**, certifique-se de configurar seu valor ao adicionar novos itens a uma lista.

## Propriedade eyPreview

Retorna ou configura um valor que determina se eventos de teclado para formulários foram acionados antes de eventos de teclado para controles. Os eventos de teclado são **eyDown**, **eyUp** e

---

eyPress.

### Sintaxe

*object.eyPreview* [= *boolean*]

A sintaxe de propriedade **eyPreview** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> que especifica como os eventos são recebidos, conforme descrito em Configurações.

### Configurações

As configurações de *boolean* são:

Configuração	Descrição
<b>True</b>	O formulário recebe eventos de teclado primeiro e, em seguida, o controle ativo.
<b>False</b>	(Padrão) O controle ativo recebe eventos de teclado; o que não ocorre com o formulário.

### Comentários

Você pode usar esta propriedade para criar um procedimento de tratamento de teclado para um formulário. Por exemplo, quando um aplicativo usa teclas de função, você querará processar os pressionamentos de teclas a nível de formulário, ao invés de escrever código para cada controle que possa receber eventos de teclado.

Se um formulário não tem controles visíveis e ativados, ele automaticamente recebe todos os eventos de teclado.

Para manipular eventos de teclado somente a nível de formulário e não permitir que controles recebam eventos de teclado, configure *eyAscii* como 0 no evento *eyPress* do formulário, e configure *eyCode* como 0 no evento *eyDown* do formulário.

**Observação:** Alguns controles interceptam eventos de teclado, de modo que o formulário não os possa receber. Os exemplos incluem a tecla ENTER quando o foco está no controle **CommandButton**, e teclas de direção quando o foco está em um controle **ListBox**.

## Propriedades LargeChange, SmallChange

- **LargeChange** — Retorna ou configura a quantidade de alteração da configuração da propriedade **Value** em um controle de barra de rolagem (**HScrollBar** ou **VScrollBar**) quando o usuário clica na área entre a caixa de rolagem e a seta de rolagem.
- **SmallChange** — Retorna ou configura a quantidade de alteração da configuração da propriedade **Value** em um controle de barra de rolagem quando o usuário clica em uma seta de rolagem.

### Sintaxe

`object.LargeChange` [= *number*]  
`object.SmallChange` [= *number*]

A sintaxe das propriedades **LargeChange** e **SmallChange** têm estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Um número inteiro que especifica a quantidade de alteração da propriedade <b>Value</b> .

### Comentários

Para ambas as propriedades, você pode especificar um número inteiro entre 1 e 32.767 inclusive. Como padrão, cada propriedade é configurada como 1.

O ambiente operacional do Microsoft Windows configura incrementos de rolagem proporcional para barras de rolagem em objetos **MDI Form**, controles **ComboBox** e controles **ListBox** com base na quantidade de dados contida no objeto. Para os controles **HScrollBar** e **VScrollBar**, entretanto, você deve especificar estes incrementos. Use **LargeChange** e **SmallChange** para configurar os incrementos de rolagem adequados à maneira como a barra de rolagem está sendo usada.

Geralmente, você configura **LargeChange** e **SmallChange** durante o tempo de criação. Você também pode reconfigurá-las no código durante o tempo de execução quando o incremento de rolagem deve alterar-se dinamicamente.

**Observação:** Você configura os intervalos máximo e mínimo dos controles **HScrollBar** e **VScrollBar** com as propriedades **Max** e **Min**.

## Propriedade LinItem

Retorna ou configura os dados passados a um controle de destino em uma conversação DDE com outro aplicativo.

### Sintaxe

`object.LinItem` [= *string*]

A sintaxe da propriedade **LinItem** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>string</i>	Uma <u>expressão de seqüência de caracteres</u> que especifica os dados a serem passados ao controle de destino.

### Comentários

Esta propriedade corresponde ao argumento *item* na sintaxe DDE padrão, com *application*, *topic* e *item* como argumentos. Para configurar esta propriedade, especifique uma unidade reconhecível de dados em um aplicativo como uma referência — por exemplo, uma referência de célula como "R1C1" no Microsoft Excel.

Use **LinItem** em combinação com a propriedade **LinTopic** para especificar o vínculo de dados completo para um controle de destino a um aplicativo de origem. Para ativar este vínculo, configure a propriedade **LinMode**.

Você configura **LinItem** somente para um controle usado como destino. Quando o formulário do Visual Basic é uma origem em uma conversação DDE, o nome de qualquer controle **Label**, **PictureBox** ou **TextBox** no formulário pode ser o argumento *item* na seqüência de caracteres *appli-*



*cation|topic|item* usada pelo destino. Por exemplo, a sintaxe abaixo representa uma referência válida do Microsoft Excel a um aplicativo do Visual Basic:  
`=VizBasicApplication|MyForm!TextBox1`

Você poderia inserir a sintaxe acima para uma célula de destino na barra de fórmulas do Microsoft Excel.

Um controle DDE pode potencialmente funcionar simultaneamente como origem e destino, provocando um loop infinito se um par origem-destino também for um par destino-origem consigo mesmo. Por exemplo, um controle **TextBox** pode ser tanto a origem (através de seu formulário pai) e destino da mesma célula no Microsoft Excel. Quando os dados em uma **TextBox** do Visual Basic forem alterados, o envio de dados ao Microsoft Excel, a célula no Microsoft Excel se altera, enviando a alteração à **TextBox**, e assim por diante, causando o loop.

Para evitar este tipo de loop, use itens relacionados, mas não idênticos, para vínculos destino-origem e origem-destino em ambas as direções entre aplicativos. Por exemplo, no Microsoft Excel, use células relacionadas (precedentes ou dependentes) para vincular uma planilha a um controle do Visual Basic, evitando o uso de um único item como origem e destino. Documente qualquer par *application|topic* que você estabeleça, se você inclui um Comando Paste Lin para uso durante o tempo de execução.

**Observação:** A configuração de um vínculo de dados permanente durante o tempo de criação com o comando **Paste Lin** no menu **Edit** também configura as propriedades **LinMode**, **LinTopic** e **LinItem**. Isto cria um vínculo que é salvo com o formulário. Cada vez que o formulário é carregado, o Visual Basic tenta restabelecer a conversação.

## Propriedade LinMode

Retorna ou configura o tipo de vínculo usado para uma conversação DDE e ativa as conexões como segue:

- Controle — Permite que um controle de destino em um formulário do Visual Basic inicie uma conversação, conforme especificado pelas propriedades **LinTopic** e **LinItem** do controle.
- Formulário — Permite que um aplicativo de destino inicie uma conversação com um formulário do Visual Basic de origem, conforme especificado pela expressão *application|topic|item* do aplicativo de destino.

### Sintaxe

*object.LinMode* [= *number*]

A sintaxe da propriedade **LinMode** tem estas três partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Um número inteiro que especifica o tipo de conexão, conforme descrito em Configurações.

### Configurações

Para controles usados como destinos em conversações DDE, as configurações de *number* são:

Constante	Configuração	Descrição
<b>VbLinNone</b>	0	(Padrão) None — Nenhuma interação DDE.
<b>VbLinAutomatic</b>	1	Automatic — O controle de destino é atualizado cada vez que os dados vinculados são alterados.
<b>VbLinManual</b>	2	Manual — O controle de destino é atualizado somente quando o método <b>LinRequest</b> é chamado.
<b>VbLinNotify</b>	3	Notify — Ocorre um evento LinNotify toda vez que os dados vinculados se alteram, mas o controle de destino somente é atualizado quando o método <b>LinRequest</b> é acionado.

Para formulários usados como fonte em conversações DDE, as configurações de *number* são:

Constante	Configuração	Descrição
-----------	--------------	-----------

<b>vbLinNone</b>	0	(Padrão) None — Nenhuma interação DDE. Nenhum aplicativo de destino pode iniciar uma conversa com a origem com o formulário de origem como tópico e nenhum aplicativo pode <u>enviar</u> dados ao formulário. Se <b>LinMode</b> é 0 (None) durante o tempo de <u>criação</u> , você não pode alterá-la para 1 (Source) durante o tempo de <u>execução</u> .
<b>vbLinSource</b>	1	Source — Permite que qualquer controle <b>Label</b> , <b>PictureBox</b> ou <b>TextBox</b> em um formulário forneça dados a qualquer aplicativo de destino que estabeleça uma conversa DDE com o formulário. Se tal vínculo existir, o Visual Basic notifica automaticamente o destino, sempre que o conteúdo de um controle tenha sido alterado. Além disso, um aplicativo de destino pode enviar dados a qualquer controle <b>Label</b> , <b>PictureBox</b> ou <b>TextBox</b> no formulário. Se <b>LinMode</b> for 1 (Source) durante o tempo de criação, você pode alterá-lo para 0 (None) e retornar durante o tempo de execução.

### Comentários

As condições abaixo também aplicam-se à propriedade **LinMode**:

- Configurar **LinMode** como um valor diferente de zero para um controle de destino faz com que o Visual Basic tente iniciar a conversa especificada nas propriedades **LinTopic** e **LinItem**. A origem atualiza o controle de destino de acordo com o tipo de vínculo especificado (automático, manual ou notificar).
- Se um aplicativo de origem termina uma conversa com um controle de destino do Visual Basic, o valor da configuração **LinMode** deste controle altera-se para 0 (None).
- Se você deixa **LinMode** de um formulário configurada como o padrão 0 (None) durante o tempo de criação, você não pode alterar **LinMode** durante o tempo de execução. Se você deseja que um formulário funcione como uma origem, você deve configurar **LinMode** como 1 (Source) durante o tempo de criação. Você pode, então, alterar o valor de **LinMode** durante o tempo de execução.

**Observação:** Configurar um vínculo de dados permanente durante o tempo de criação a um comando **Paste Lin** do menu **Edit** também configura as propriedades **LinMode**, **LinTopic** e **LinItem**. Isto cria um vínculo que é salvo junto com o formulário. Cada vez que o formulário é carregado, o Visual Basic tenta restabelecer a conversa.

## Propriedade LinTimeout

Retorna ou configura a quantidade de tempo que um controle espera por uma resposta a uma mensagem DDE.

### Sintaxe

*object*.**LinTimeout** [= *number*]

A sintaxe da propriedade **LinTimeout** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Uma <u>expressão numérica</u> que especifica o tempo de

---

espera.

### Comentários

Como padrão, a propriedade **LinTimeout** é configurada como 50 (equivalente a 5 segundos). Você pode especificar outras configurações em décimos de segundo.

O tempo de resposta DDE dos aplicativos de origem variam. Use esta propriedade para ajustar o tempo que um controle de destino aguarda por uma resposta de um aplicativo de origem. Se você usa **LinTimeout**, pode evitar a geração de um erro do Visual Basic se um determinado aplicativo de origem levar tempo demais para responder.

**Observação:** O período máximo de tempo que um controle pode aguardar é 65.535 décimos de segundo, ou cerca de 1 hora e 49 minutos. Configurar **LinTimeout** como 1 informa ao controle para aguardar o tempo máximo por uma resposta em uma conversação DDE. O usuário pode forçar o controle a interromper a espera pressionando a tecla ESC.

## Propriedade Lced

Retorna ou configura um valor indicando se um controle pode ser editado.

### Sintaxe

*object.Lced* [ = *boolean*]

A sintaxe da propriedade **Lced** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> que especifica se o controle pode ser editado, conforme descrito em Configurações.

### Configurações

As configurações de *boolean* são:

Configuração	Descrição
<b>True</b>	Controle <b>TextBox</b> — você pode rolar e realçar o texto no controle, mas você não pode editá-lo. O programa ainda pode modificar o texto alterando a propriedade <b>Text</b> . Objeto <b>Column</b> — você não pode editar os valores na coluna. Objeto <b>ComboBox</b> — você não pode digitar na caixa de texto.
<b>False</b>	Controle <b>TextBox</b> — você pode editar o texto no controle. Objeto <b>Column</b> — você pode editar os valores na coluna. Objeto <b>ComboBox</b> — você pode digitar na caixa de texto e produzir sua lista drop-down.

### Comentários

Para o objeto **Column**, a configuração padrão de **Lced** é o valor da propriedade **DataUpdatable** para o campo subjacente; entretanto, se **Column** for não acoplada ou a origem dos dados não suportar **DataUpdatable**, o padrão é **True**. Se **DataUpdatable** no campo subjacente for **False**, você não cria um erro configurando esta propriedade como **True**. Entretanto, ocorrerá um erro quando o controle tentar gravar os dados alterados no banco de dados.

Para o controle **ComboBox**, quando **Lced** for configurada como **True**, o usuário não pode alterar quaisquer dados, mas pode selecionar dados na caixa de textos e copiá-los. Esta propriedade não afeta o acesso à **ComboBox** através de programa.

## Propriedade ReadOnly

Retorna ou configura um valor que determina se um controle **FileListBox** contém arquivos com atributos Somente leitura.

### Sintaxe

*object.ReadOnly* [ = *boolean*]

---

A sintaxe da propriedade **ReadOnly** tem estas partes:

<b>Parte</b>	<b>Descrição</b>
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> que especifica se o controle exibe arquivos com atributo Somente leitura, conforme descrito em Configurações.

### **Configurações**

As configurações de *boolean* são:

<b>Configuração</b>	<b>Descrição</b>
<b>True</b>	O controle inclui arquivos Somente leitura na lista.
<b>False</b>	Nenhum arquivo Somente leitura é listado no controle.

### **Comentários**

Use a propriedade **ReadOnly** com um controle **FileListBox** para especificar se os arquivos com atributo Somente leitura são exibidos ou não na lista de arquivos.

---

## Exemplo da propriedade Align

Este exemplo utiliza um controle **PictureBox** como barra de ferramentas em um objeto **MDIForm** com um controle **CommandButton** para mover a **PictureBox**, da parte superior para a parte inferior do formulário. Para experimentar este exemplo, crie um novo **MDIForm** e configure a propriedade **MDIChild** de Form1 como **True**. Desenhe uma **PictureBox** no **MDIForm**, e coloque um **CommandButton** na **PictureBox**. Cole o código na seção Declarations do **MDIForm** e, em seguida, pressione F5. Clique no **CommandButton** para mover a **PictureBox**.

```
Private Sub Command1_Click ()
    If Picture1.Align = vbAlignTop Then
        Picture1.Align = vbAlignBottom
        ' Alinha-se à parte inferior do formulário.
    Else
        Picture1.Align = vbAlignTop
        ' Alinha-se à parte superior do formulário.
    End If
End Sub
```

## Exemplo da propriedade AutoRedraw

Este exemplo exibe, alternadamente, dois elementos gráficos em um controle **PictureBox**: um círculo preenchido permanente e linhas verticais temporárias. Clique em **PictureBox** para desenhar ou redesenhar as linhas. Redimensionar o formulário exige que o elemento gráfico temporário seja redesenhado. Para experimentar este exemplo cole o código na seção Declarations de um formulário que tenha um controle **PictureBox** chamado Picture1. Pressione F5 para executar o programa e clique no elemento gráfico, cada vez que o formulário for redimensionado.

```
Private Sub Form_Load ()
    Picture1.ScaleHeight = 100      ' Configurar a escala como 100.
    Picture1.ScaleWidth = 100
    Picture1.AutoRedraw = True      ' Ativar AutoRedraw.
    Picture1.ForeColor = 0         ' Configurar ForeColor.
    Picture1.FillColor = QBColor(9) ' Configurar FillColor.
    Picture1.FillStyle = 0         ' Configurar FillStyle.
    Picture1.Circle (50, 50), 30    ' Desenhar um círculo.
    Picture1.AutoRedraw = False     ' Desativar AutoRedraw.
End Sub

Private Sub Picture1_Click ()
    Dim I ' Declarar variável.
    Picture1.ForeColor = RGB(Rnd * 255, 0, 0) ' Seleciona cor aleatória.
    For I = 5 To 95 Step 10 ' Desenhar linhas.
        Picture1.Line (I, 0)-(I, 100)
    Next
End Sub
```

## Exemplo da propriedade AutoShowChildren

Este exemplo apresenta um objeto **MDIForm** com um formulário MDI filho, usa a propriedade **AutoShowChildren** para criar um formulário oculto como outra ocorrência do formulário MDI filho e, em seguida, cria um formulário MDI filho visível. Para experimentar este exemplo, configure a propriedade **MDIChild** como **True** em Form1 e, em seguida, crie um **MDIForm** com o comando **Add MDI Form** no menu **Project**. Copie o código para a seção Declarations do **MDIForm** e, em seguida, pressione F5 para executar o programa.

```
Private Sub MDIForm_Load()
    MDIForm1.AutoShowChildren = False      ' Configurar para ocultar os
    formulários filhos.
    Dim HideForm As New Form1 ' Declarar novo formulário.
    HideForm.Caption = "HideForm"         ' Configurar sua legenda.
    Load HideForm ' Carrega-o; ele está oculto.
    MDIForm1.AutoShowChildren = True      ' Configurar para exibir formu-
    lários filhos.
    Dim ShowForm As New Form1 ' Declarar outro novo formulário.
    ShowForm.Caption = "ShowForm"         ' Configurar sua legenda.
    Load ShowForm ' Carregue-o; ele é exibido.
```

---

End Sub

## Exemplo da propriedade ClipControls

Este exemplo mostra como a propriedade **ClipControls** afeta a regeneração de um formulário. Para experimentar este exemplo cole o código na seção Declarations de um formulário e, em seguida, pressione F5. Observe que a cor do formulário inteiro se altera, cada vez que você o redimensiona ou cobre parte dele com outro formulário ou aplicativo. Encerre o programa e configure **ClipControls** como **False** e, em seguida, execute novamente o programa. Observe que somente as partes recém-expostas do formulário são regeneradas.

```
Private Sub Form_Paint ()  
    ' Selecionar uma cor aleatória para o segundo plano.  
    BackColor = &HFFFFFF * Rnd  
End Sub
```

## Exemplo da propriedade Columns

Este exemplo ilustra como os dois tipos diferentes de controles **ListBox** funcionam quando contêm os mesmos dados. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha dois controles **ListBox**. Configure a propriedade **Columns** como 2 para List2 e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Load ()  
    Dim I ' Declara variável.  
    List1.Move 50, 50, 2000, 1750 ' Organizar caixas de listagem.  
    List2.Move 2500, 50, 3000, 1750  
    For I = 0 To Screen.FontCount - 1  
        List1.AddItem Screen.Fonts(I) ' Preencher ambas as caixas com  
        List2.AddItem Screen.Fonts(I) ' nomes de fontes de tela.  
    Next I  
End Sub
```

## Exemplo da propriedade DragIcon

Este exemplo altera a configuração da propriedade **DragIcon** toda vez que você arrasta um controle **PictureBox**. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha um controle **PictureBox**. Configure a propriedade **DragMode** = 1 e, em seguida, pressione F5 e arraste o controle **PictureBox**.

```
Private Sub Form_DragDrop (Source As Control, X As Single, Y As Single)  
    Dim Pic ' Declara variável.  
    Source.Move X, Y ' Configurar a posição do controle.  
    Pic = "ICONS\OFFICE\CRDFLE01.ICO" ' Obter o nome do arquivo de ícone.  
    If Source.DragIcon = False Then ' Se nenhuma figura estiver carregada,  
        Source.DragIcon = LoadPicture(Pic) ' Carregar a figura.  
    Else  
        Source.DragIcon = LoadPicture() ' Descarregar a figura.  
    End If  
End Sub
```

---

## Exemplo da propriedade DragMode

Este exemplo ativa e desativa a capacidade de arrastar um controle **CommandButton** cada vez que um formulário é clicado. Para experimentar este exemplo cole o código na seção **Declarations** de um formulário que contenha um **CommandButton** e, em seguida, pressione F5 e clique no formulário

```
Private Sub Form_Click ()
    ' Verificar o DragMode.
    If Command1.DragMode = vbManual Then
        ' Ativá-lo.
        Command1.DragMode = vbAutomatic
    Else
        ' ou desativá-lo.
        Command1.DragMode = vbManual
    End If
End Sub
```

## Exemplo da propriedade DrawMode

Este exemplo permite o desenho de um formulário arrastando-se o ponteiro do mouse. Cada clique de mouse configura um valor diferente para a propriedade **DrawMode**. Para experimentar este exemplo cole o código na seção **Declarations** de um formulário e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Load
    DrawWidth = 10 ' Configurar DrawWidth.
End Sub

Private Sub Form_Click ()
    Static M As Integer ' Configuração atual de DrawMode.
    ForeColor = QBColor(Int(Rnd * 15)) ' Escolher uma cor.
    M = ((M + 1) Mod 16) + 1 ' Manter DrawMode 16 ou menor.
    DrawMode = M ' Configurar DrawMode.
End Sub

Private Sub Form_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button Then ' Enquanto o botão está pressionado,
        PSet (X, Y) ' desenhe um grande ponto.
    End If
End Sub
```

## Exemplo da propriedade DrawStyle

Este exemplo desenha sete linhas no formulário, sendo que cada linha exibe uma propriedade **DrawStyle** diferente. (Se você configura **AutoRedraw = True**, o formulário acumula um novo conjunto de linhas cada vez que você o redimensiona e, em seguida, clica nele.) Para experimentar este exemplo cole o código na seção **Declarations** de um formulário e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Click ()
    Dim I ' Declarar variável.
    ScaleHeight = 8 ' Dividir a altura por 8.
    For I = 0 To 6
        DrawStyle = I ' Alterar o estilo.
        Line (0, I + 1) - (ScaleWidth, I + 1) ' Desenha uma nova linha.
    Next I
End Sub
```

---

## Exemplo da propriedade Drive

Este exemplo exibe uma lista de arquivos para a unidade de disco e diretório atuais. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha um controle **DriveListBox**, um controle **DirListBox** e um controle **FileListBox** e, em seguida, pressione F5. Use o mouse para alterar a unidade de disco e diretório.

```
Private Sub Drive1_Change ()
    Dir1.Path = Drive1.Drive ' Quando a unidade de disco for alterada,
    configure o caminho do diretório.
End Sub
Private Sub Dir1_Change ()
    File1.Path = Dir1.Path ' Quando o diretório se alterar configure o
    caminho do arquivo.
End Sub
```

## Exemplo da propriedade FileName

Este exemplo exibe uma mensagem em um controle Label quando um nome de arquivo em um controle **FileListBox** é clicado duas vezes. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha um controle **Label**, um controle **DirListBox** e um controle **FileListBox** e, em seguida, pressione F5, e clique duas vezes em qualquer nome de arquivo no controle **FileListBox**.

```
Private Sub Dir1_Change ()
    File1.Path = Dir1.Path ' Configurar o caminho de File1.
End Sub
Private Sub File1_PathChange ()
    Dir1.Path = File1.Path ' Configurar o caminho de Dir1.
End Sub
Private Sub File1_DblClick ()
    ' Exibir o nome de arquivo selecionado ao ser clicado duas vezes.
    Label1.Caption = "Your selection: " +_ File1.FileName
End Sub
```

## Exemplo da propriedade FillColor

Este exemplo constrói um círculo em seu formulário com configurações aleatórias de propriedade **FillColor** e **FillStyle** quando você clica o mouse. Para experimentar este exemplo cole o código na seção Declarations de um formulário e, em seguida pressione F5 e clique no formulário.

```
Private Sub Form_MouseDown (Button As Integer, Shift As Integer, X As
Single, Y As Single)
    FillColor = QBColor(Int(Rnd * 15)) ' Escolher FillColor aleatória.
    FillStyle = Int(Rnd * 8) ' Escolher FillStyle aleatório.
    Circle (X, Y), 250 ' Desenhar um círculo.
End Sub
```

## Exemplo da propriedade FillStyle

Este exemplo exibe um círculo em um formulário com configurações aleatórias de **FillColor** e **FillStyle** quando você clica o mouse. Para experimentar este exemplo cole o código na seção Declarations e, em seguida, pressione F5 para executar o programa.

```
Private Sub Form_MouseDown (Button As Integer, Shift As Integer, X As
Single, Y As Single)
    FillColor = QBColor(Rnd * 15) ' Escolher FillColor aleatória.
    FillStyle = Int(Rnd * 8) ' Escolher FillStyle aleatório.
    Circle (X, Y), 250 ' Desenhar um círculo.
End Sub
```



---

## Exemplo da propriedade FontCount

Este exemplo imprime uma lista de fontes de impressora em um controle **ListBox**. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha um controle **ListBox** chamado List1 e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Click ()
    Dim I ' Declarar variável.
    For I = 0 To Printer.FontCount -1      ' Determinar o número de fontes.
        List1.AddItem Printer.Fonts (I)    ' Colocar cada fonte na caixa de listagem.
    Next I
End Sub
```

## Exemplo da propriedade Fonts

Este exemplo imprime uma lista das fontes de impressora em um controle **ListBox**. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha um controle **ListBox** chamado List1. Pressione F5 para executar o programa e, em seguida, clique no formulário.

```
Private Sub Form_Click ()
    Dim I ' Declara variável.
    For I = 0 To Printer.FontCount -1      ' Determinar o número de fontes.
        List1.AddItem Printer.Fonts (I)    ' Colocar cada fonte na caixa de listagem.
    Next I
End Sub
```

## Exemplo da propriedade hDC

Este exemplo desenha um triângulo e, em seguida, usa uma função do Microsoft Windows para preenchê-lo com cor. Para experimentar este exemplo, crie um novo módulo usando o comando **Add Module** no menu **Project**. Cole a instrução **Declare** na seção Declarations do novo módulo, certificando-se de que a instrução esteja em uma linha sem quebras ou mudanças de linha. Em seguida, cole o procedimento **Sub** na seção Declarations de um formulário. Pressione F5 e clique no formulário.

```
' Declaração de uma rotina Windows. Esta instrução é
' para o módulo.
Declare Sub FloodFill Lib "GDI32" Alias "FloodFill" _
    (ByVal hDC As Long, ByVal X As Long, ByVal Y As _
    Long, ByVal crColor As Long) As Long
' Colocar o código abaixo no formulário.
Private Sub Form_Click ()
    ScaleMode = vbPixels      ' O Windows desenha em pixels.
    ForeColor = vbBlack       ' Configurar a linha de desenho como preto.
    Line (100, 50)-(300, 50) ' Desenhando um triângulo.
    Line -(200, 200)
    Line -(100, 50)
    FillStyle = vbFSSolid     ' Configurar FillStyle como uniforme.
    FillColor = RGB(128, 128, 255) ' Configurar FillColor.
    ' Chamar o Windows API para preencher.
    FloodFill hDC, 200, 100, ForeColor
End Sub
```

---

## Exemplo da propriedade HideSelection

Este exemplo permite selecionar texto em cada formulário e alternar o foco entre formulários clicando a barra de título de cada formulário. A seleção permanece visível até mesmo quando o formulário não está ativo. Para experimentar este exemplo, crie dois formulários e desenhe um controle **TextBox** em cada um deles. Configure a propriedade **MultiLine** como **True** para ambos os controles **TextBox** e configure a propriedade **HideSelection** como **False** para um dos controles **TextBox**. Cole o código na seção Declarations de ambos os módulos do formulário e, em seguida, pressione F5.

```
Private Sub Form_Load ()
    Open "README.TXT" For Input As 1          ' Carregar o arquivo na caixa
de texto.
    Text1.Text = Input$(LOF(1), 1)
    Close 1
    Form2.Visible = True ' Carregar Form2, se ainda não estiver carregado.
    ' Posicionar os formulários lado a lado.
    Form1.Move 0, 1050, Screen.Width / 2, Screen.Height
    Form2.Move Screen.Width / 2, 1050, Screen.Width / 2, Screen.Height
    ' Ampliar a caixa de texto para preencher o formulário.
    Text1.Move 0, 0, ScaleWidth, ScaleHeight
End Sub
```

## Exemplo da propriedade hWnd

Este exemplo força o formulário a permanecer sempre em primeiro plano. Para experimentar este exemplo, crie um formulário (não um formulário MDI filho) e, em seguida, crie um menu para o formulário chamado **Main**. Insira nele um sub-menu chamado **Always On Top** e configure seu **Name** com mnuTopmost. Crie um novo módulo usando o comando **Add Module** no menu **Project**. Cole a instrução **Declare** na seção Declarations do novo módulo, certificando-se de que a instrução esteja em uma linha sem quebras ou mudanças de linha. Em seguida, cole o procedimento **Sub** na seção Declarations do formulário e pressione F5.

```
' Declaração de uma rotina Windows.
' Esta instrução deve ser colocada no módulo.
Declare Function SetWindowPos Lib "user32" Alias_ "SetWindowPos" (ByVal
hwnd As Long, ByVal_ hWndInsertAfter As Long, ByVal x As Long, ByVal y
As_ Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags_ As Long) As
Long

' Configurar alguns valores de constante (de WIN32API.TXT).
Const conHwndTopmost = -1
Const conHwndNoTopmost = -2
Const conSwpNoActivate = &H10
Const conSwpShowWindow = &H40

Private Sub mnuTopmost_Click ()
    ' Adicionar ou remover a marca de seleção do menu.
    mnuTopmost.Checked = Not mnuTopmost.Checked
    If mnuTopmost.Checked Then
        ' Ativar o atributo TopMost.
        SetWindowPos hWnd, conHwndTopmost, 0, 0, 0, 0, _ conSwpNoActivate
Or conSwpShowWindow
    Else
        ' Desativar o atributo TopMost.
        SetWindowPos hWnd, conHwndNoTopmost, 0, 0, 0, 0, _ 0, conSwpNoActivate
Or conSwpShowWindow
    End If
End Sub
```

Este exemplo produz automaticamente a parte de lista drop-down de um controle **ComboBox** toda vez que o **ComboBox** recebe o foco. Para experimentar este exemplo, crie um novo formulário contendo um controle **ComboBox** e um controle **OptionButton** (usado apenas para receber o foco). Crie um novo módulo usando o comando **Add Module** no menu **Project**. Cole a instrução

---

**Declare** na seção Declarations do novo módulo, certificando-se de que a instrução esteja em uma linha sem quebra ou mudança de linha. Em seguida, cole o procedimento **Sub** na seção Declarations do formulário e pressione F5. Use a tecla TAB para mover o foco de, e para a **ComboBox**.  
Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hwnd As Long, ByVal wParam As Long, ByVal lParam As Long) As Long

```
Private Sub Combol_GotFocus ()  
    Const CB_SHOWDROPDOWN = &H14F  
    Dim Tmp  
    Tmp = SendMessage(Combol.hWnd, CB_SHOWDROPDOWN, 1, ByVal 0&)  
End Sub
```

## Exemplo da propriedade Image

Este exemplo desenha um círculo no primeiro controle **PictureBox** toda vez que você clica nele. Quando você clica na segunda **PictureBox**, o elemento gráfico da primeira **PictureBox** é copiado nela. Para experimentar este exemplo, cole o código na seção Declarations de um formulário que contenha dois grandes controles **PictureBox** do mesmo tamanho. Pressione F5 para executar o programa e, em seguida, clique nos controles **PictureBox**.

```
Private Sub Form_Load ()  
    ' Configurar AutoRedraw como True.  
    Picture1.AutoRedraw = True  
End Sub  
  
Private Sub Picture1_Clic ()  
    ' Declara variáveis.  
    Dim PW, PH  
    ' Configurar FillStyle como Solid.  
    Picture1.FillStyle = vbFSSolid  
    ' Escolher a cor aleatória.  
    Picture1.FillColor = QBColor(Int(Rnd * 15))  
    PW = Picture1.ScaleWidth ' Configurar ScaleWidth.  
    PH = Picture1.ScaleHeight ' Configurar ScaleHeight.  
    ' Desenhar um círculo em local aleatório.  
    Picture1.Circle (Int(Rnd * PW), Int(Rnd * PH)), 250  
End Sub  
  
Private Sub Picture2_Clic ()  
    ' Copiar Image para Picture2.  
    Picture2.Picture = Picture1.Image  
End Sub
```

## Exemplo da propriedade ItemData

Este exemplo preenche um controle **ListBox** com nomes de funcionários e preenche a matriz de propriedades **ItemData** com os números dos funcionários usando a propriedade **NewIndex** para manter os números sincronizados com a lista classificada. Um controle **Label** exibe o nome e número de um item, quando o usuário efetua uma seleção. Para experimentar este exemplo, cole o código na seção Declarations de um formulário que contenha uma **ListBox** e um **Label**. Configure a propriedade **Sorted** para a **ListBox** como **True** e, em seguida, pressione F5 e clique na **ListBox**.

```
Private Sub Form_Load ()  
    ' Preencher List1 e a matriz ItemData com  
    ' os itens correspondentes em ordem classificada.  
    List1.AddItem "Judy Phelps"  
    List1.ItemData(List1.NewIndex) = 42310  
    List1.AddItem "Chien Lieu"  
    List1.ItemData(List1.NewIndex) = 52855  
    List1.AddItem "Mauro Sorrento"  
    List1.ItemData(List1.NewIndex) = 64932  
    List1.AddItem "Cynthia Bennet"  
    List1.ItemData(List1.NewIndex) = 39227
```

---

```
End Sub
```

```
Private Sub List1_Clic ()  
    ' Anexar o número e o nome do funcionário.  
    Msg = List1.ItemData(List1.ListIndex) & " "  
    Msg = Msg & List1.List(List1.ListIndex)  
    Label1.Caption = Msg  
End Sub
```

## Exemplo da propriedade eyPreview

Este exemplo cria um manipulador de teclado de formulário no evento eyDown. Cada uma das quatro primeiras teclas de função exibe uma mensagem diferente. Para experimentar este exemplo cole o código na seção Declarations de um formulário e, em seguida, pressione F5. Uma vez que o programa esteja em execução, pressione qualquer uma das quatro primeiras (F1 – F4) teclas de função.

```
Private Sub Form_Load ()  
    eyPreview = True  
End Sub  
  
Private Sub Form_eyDown (eyCode As Integer, Shift As Integer)  
    Select Case eyCode  
        Case vbeyF1: MsgBox "F1 is your friend."  
        Case vbeyF2: MsgBox "F2 could copy text."  
        Case vbeyF3: MsgBox "F3 could paste text."  
        Case vbeyF4: MsgBox "F4 could format text."  
    End Select  
End Sub
```

---

## Exemplo das propriedades LargeChange, SmallChange

Este exemplo usa uma barra de rolagem para mover um controle **PictureBox** pelo formulário. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha um pequeno controle **PictureBox** e um controle **HScrollBar** e, em seguida, pressione F5 e clique na barra de rolagem.

```
Private Sub Form_Load ()
    HScroll1.Max = 100      ' Configurar o valor máximo.
    HScroll1.LargeChange = 20 ' Cruzar em 5 cliques.
    HScroll1.SmallChange = 5  ' Cruzar em 20 cliques.
    Picture1.Left = 0 ' Iniciar a figura à esquerda.
    Picture1.BacColor = QBColor(3) ' Configura a cor da caixa de figura.
End Sub
Private Sub HScroll1_Change ()
    ' Mover a figura de acordo com a barra de rolagem.
    Picture1.Left = (HScroll1.Value / 100) * ScaleWidth
End Sub
```

## Exemplo das propriedades LinItem, LinMode, LinTopic

No exemplo, cada clique de mouse faz com que uma célula em uma planilha do Microsoft Excel atualize o conteúdo de um controle **TextBox** do Visual Basic. Para experimentar este exemplo, inicie o Microsoft Excel, abra uma nova planilha chamada Sheet1 e coloque alguns dados na primeira coluna. No Visual Basic, crie um formulário contendo um controle **TextBox**. Cole o código na seção Declarations e, em seguida, pressione F5 para executar o programa.

```
Private Sub Form_Click ()
    Dim CurRow As String
    Static Row ' Número de linha da planilha.
    Row = Row + 1 ' Incrementar Row.
    If Row = 1 Then ' Somente na primeira vez.
        ' Certificar-se de que o vínculo não está ativo.
        Text1.LinMode = 0
        ' Configurar o nome do aplicativo e o nome do tópico.
        Text1.LinTopic = "Excel|Sheet1"
        Text1.LinItem = "R1C1" ' Configurar LinItem.
        Text1.LinMode = 1 ' Configurar LinMode como Automatic.
    Else
        ' Atualizar a linha no item de dados.
        CurRow = "R" & Row & "C1"
        Text1.LinItem = CurRow ' Configurar LinItem.
    End If
End Sub
```

---

## Propriedade GridLineWidth

Retorna ou configura a largura em pixels das linhas de grade para um controle **MSFlexGrid**.

### Sintaxe

*object*.GridLineWidth [= *value*]

A sintaxe da propriedade **GridLineWidth** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Um número inteiro especificando a largura da linha de grade. A configuração mínima é 1 (padrão); a configuração máxima é 10.

## Propriedade LinTopic

Para um controle de destino — retorna ou configura o aplicativo de origem e o tópico (o agrupamento de dados fundamentais usados naquele aplicativo). Use **LinTopic** com a propriedade **LinItem** para especificar o vínculo de dados completo.

Para um formulário de origem — retorna ou configura o tópico ao qual o formulário de origem responde em uma conversa DDE.

### Sintaxe

*object*.LinTopic [= *value*]

A sintaxe da propriedade **LinTopic** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Uma <u>expressão de sequência de caracteres</u> especificando um elemento de sintaxe DDE.

### Comentários

A propriedade **LinTopic** consiste em uma sequência de caracteres que fornece parte das informações necessárias para configurar um vínculo de destino ou vínculo de origem. A sequência de caracteres que você usa depende se você está trabalhando com um controle de destino ou um formulário de origem. Cada sequência de caracteres corresponde a um ou mais elementos da sintaxe DDE padrão que inclui *application*, *topic* e *item*.

**Observação:** Embora a definição padrão de um vínculo DDE inclua os elementos *application*, *topic* e *item*, a sintaxe efetiva usada em aplicativos para um vínculo de destino a um aplicativo de origem pode variar ligeiramente. Por exemplo, no Microsoft Excel, você usa a sintaxe: *application|topic|item*

Dentro do Microsoft Word for Windows, você usa:

*application topic item*

(Não utilize o caractere de ligação [|] ou ponto de exclamação [!].)

Dentro de um aplicativo do Visual Basic, você usa:

*application|topic*

O ponto de exclamação para *topic* está implícito.

**Destination Control** Para configurar **LinTopic** para um controle de destino, use uma sequência de caracteres com a sintaxe *application|topic* como segue:

- *application* é o nome do aplicativo de onde os dados são solicitados, normalmente o nome de arquivo executável, sem um extensão — por exemplo, Excel (para o Microsoft Excel).
- O caractere de ligação ([, ou código de caractere 124) separa o aplicativo do tópico.
- *topic* é o dado fundamental de agrupamento usado no aplicativo de origem — por exemplo, uma planilha no Microsoft Excel.

Além disso, somente para um controle de destino, você deve configurar a propriedade **LinItem** relacionada para especificar o elemento *item* para o vínculo. Uma referência de célula como, por exemplo, R1C1, corresponde a um item em uma planilha do Microsoft Excel.

**Formulário de origem** Para configurar **LinTopic** de um formulário de origem, configure *value* como um identificador adequado para o formulário. Um aplicativo de destino usa esta sequência

de caracteres como argumento *topic* ao estabelecer um vínculo DDE com o formulário. Embora esta seqüência de caracteres seja tudo o que você precisa para configurar **LinTopic** no Visual Basic para um formulário de origem, o aplicativo de destino também precisa especificar:

- O elemento *application* que o aplicativo de destino utiliza, que é o nome de arquivo do projeto do Visual Basic sem a extensão .vbp (se você estiver executando seu aplicativo no ambiente de desenvolvimento do Visual Basic) ou o nome de arquivo do aplicativo do Visual Basic sem a extensão .exe (se você estiver executando seu aplicativo como um arquivo executável autônomo). A propriedade **EXENAME** do objeto **App** oferece esta seqüência de caracteres em seu código do Visual Basic, a menos que o nome de arquivo tenha sido alterado pelo usuário. (**EXENAME** sempre retorna o verdadeiro nome de arquivo do aplicativo em disco; DDE sempre utiliza o nome original que foi especificado na caixa de diálogo **Project Properties**)
- O elemento *item* que é utilizado pelo aplicativo de destino, que corresponde à configuração da propriedade **Name** para o controle **Label**, **PictureBox** ou **TextBox** no formulário de origem.

A sintaxe abaixo é um exemplo de uma referência válida do Microsoft Excel a um aplicativo do Visual Basic funcionando como origem:

=VizBasicApplication|Form1|TextBox1

Você poderia inserir esta referência a uma célula de destino na barra de fórmulas do Microsoft Excel.

Para ativar o vínculo de dados configurado com **LinTopic**, configure a propriedade **LinMode** como o valor não-zero adequado para especificar o tipo de vínculo desejado. Como regra geral, configure **LinMode** após você ter configurado **LinTopic**. Para um controle de destino, alterar **LinTopic** quebra um vínculo existente e encerra a conversa  o DDE. Para um formulário de origem, alterar **LinTopic** quebra todos os vínculos de destino que estejam usando este t  pico. Por estes motivos, sempre configure a propriedade **LinMode** como 0 antes de alterar **LinTopic**. Ap  s alterar **LinTopic** para um controle de destino, voc   deve configurar **LinMode** como 1 (Automatic), 2 (Manual) ou 3 (Notify) para estabelecer uma conversa  o com o novo t  pico.

**Observa  o:** Configurar um v  nculo de dados permanente durante o tempo de cria  o com o comando **Paste Lin** no menu **Edit** tamb  m configura as propriedades **LinMode**, **LinTopic** e **LinItem**. Isto cria um v  nculo que    salvo com o formul  rio. Cada vez que o formul  rio    carregado, o Visual Basic tenta restabelecer a conversa  o.

## Propriedades Max e Min (Barra de rolagem)

- **Max** — retorna ou configura a propriedade **Value** m  xima da posi  o de uma caixa de rolagem quando ela est   na extremidade inferior ou na extremidade superior. Para o controle **ProgressBar**, ele retorna ou configura seu valor m  ximo.
- **Min** — retorna ou configura a propriedade **Value** m  nima quando a caixa de rolagem est   na extremidade superior ou inferior. Para o controle **ProgressBar**, ele retorna ou configura seu valor m  nimo.

### Sintaxe

*object.Max* [= value]

*object.Min* [= value]

As sintaxes das propriedades **Max** e **Min** t  m estas partes:

Parte	Descri��o
<i>object</i>	Uma <u>express��o de objeto</u> que avalia para um objeto na lista <b>Applies To</b> .
<i>value</i>	Uma <u>express��o num��rica</u> especificando a configura��o m��xima ou m��nima da propriedade <b>Value</b> , conforme descrito em Configura��es.

### Configura  es

Para cada propriedade, voc   pode especificar um n  mero inteiro entre -32.768 e 32.767, inclusive. As configura  es padr  o s  o:

- **Max** — 32.767.
- **Min** — 0.

### Coment  rios

O ambiente operacional do Microsoft Windows automaticamente configura intervalos para barras de rolagem proporcionais ao conte  do de formul  rios, controles **ComboBox** e controles **ListBox**. Para um controle de barra de rolagem (**HScrollBar** ou **VScrollBar**), entretanto, voc   deve espe-

cificar estes intervalos. Use **Max** e **Min** para configurar um intervalo adequado para o modo como o controle de barra de rolagem é usado — por exemplo, como dispositivo de entrada ou como um indicador de velocidade ou quantidade.

De maneira típica, você configura **Max** e **Min** durante o tempo de criação. Você também pode configurá-los em código durante o tempo de execução se o intervalo de rolagem precisar ser alterado de maneira dinâmica — por exemplo, ao adicionar registros a um banco de dados que pode ser inteiramente rolado. Você configura os incrementos de rolagem máximos e mínimos para um controle de barra de rolagem através das propriedades **LargeChange** e **SmallChange**.

**Observação:** Se **Max** for configurado como menor que **Min**, o valor máximo é configurado como a posição da extremidade esquerda ou extremidade superior, respectivamente, de uma barra de rolagem horizontal ou vertical. A propriedade **Max** de um controle **ProgressBar** deve ser sempre maior que sua propriedade **Min**, e sua propriedade **Min** deve ser sempre maior que ou igual a 0.

As propriedades **Max** e **Min** definem o intervalo do controle. A propriedade **Min** do controle **ProgressBar** é 0 e, como padrão, sua propriedade **Max** é 100, representando a porcentagem de duração da operação.

## Propriedade MaxLength

Retorna ou configura um valor indicando se existe um número máximo de caracteres que podem ser inseridos no controle **TextBox** e, se for o caso, especifica o número máximo de caracteres que podem ser inseridos.

**Observação:** Em sistemas DBCS (conjunto de caracteres de byte duplo), cada caractere pode ocupar dois bytes ao invés de apenas um, o que limita o número de caracteres que podem ser inseridos.

### Sintaxe

*object.MaxLength [= value]*

A sintaxe da propriedade **MaxLength** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Um número inteiro especificando o número máximo de caracteres que um usuário pode inserir em um controle <b>TextBox</b> . O padrão para a propriedade <b>MaxLength</b> é 0, indicando nenhum máximo diferente daquele criado pelas restrições de memória no sistema do usuário para controles <b>TextBox</b> de linha única e um máximo de aproximadamente 32 para controles <b>TextBox</b> de linhas múltiplas. Qualquer número superior a 0 indica o número máximo de caracteres.

### Comentários

Use a propriedade **MaxLength** para limitar o número de caracteres que podem ser inseridos pelo usuário em uma **TextBox**.

Se o texto que excede à configuração da propriedade **MaxLength** for atribuído a uma **TextBox** a partir de código, nenhum erro ocorre; mas, somente o número máximo de caracteres é atribuído à propriedade **Text**, e os caracteres extras são truncados. A alteração desta propriedade não afeta o conteúdo atual de uma **TextBox**, mas afetará quaisquer alterações subseqüentes do conteúdo.

## Propriedade MDIChild

Retorna ou configura um valor indicando se um formulário é exibido como formulário MDI filho dentro de um formulário MDI. Somente leitura durante o tempo de execução.

### Sintaxe

*object.MDIChild*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Configurações

A configuração da propriedade **MDIChild** são:

Configuração	Descrição
--------------	-----------



---

<b>True</b>	O formulário é um formulário MDI filho e é exibido dentro do formulário MDI pai.
<b>False</b>	(Padrão) O formulário não é um formulário MDI filho.

### Comentários

Use esta propriedade ao criar um aplicativo de interface de documentos múltiplos (MDI). Durante o tempo de execução, os formulários com esta propriedade configurada como **True** são exibidos em um formulário MDI. Um formulário MDI filho pode ser maximizado, minimizado ou movido, tudo isto dentro do formulário MDI pai.

Ao trabalhar com formulários MDI filho, tenha em mente o seguinte:

- Durante o tempo de execução, quando um formulário MDI filho é maximizado, sua legenda é combinada com aquela do formulário MDI pai.
- Durante o tempo de criação, um formulário MDI filho é exibido como qualquer outro formulário, pois ele é exibido dentro de formulário pai durante o tempo de execução. O ícone de um formulário MDI filho na janela **Project** é diferente de ícones de outros tipos de formulários.
- Formulários MDI filho não podem ser de janela restrita.
- O tamanho inicial e posicionamento de formulários MDI filho são controlados pelo ambiente operacional do Microsoft Windows, a menos que você especificamente os configure no procedimento de evento Load.
- Se um formulário MDI filho é referido antes que o pai seja carregado, o formulário MDI pai é automaticamente carregado. Entretanto, se o formulário MDI pai for referido antes de carregar um formulário MDI filho, o formulário filho não é carregado.

**Observação:** Todos os formulários MDI filho têm bordas dimensionáveis, uma caixa de menu de controle e botões **Minimize** e **Maximize**, independente da configuração das propriedades **BorderStyle**, **ControlBox**, **MinButton**, e **MaxButton**.

Qualquer referência a um objeto **MDIForm**, incluindo a leitura e configuração de propriedades, provoca a carga do formulário e este se torna visível.

---

## Propriedade MouseIcon

Retorna ou configura um ícone de mouse personalizado.

### Sintaxe

*object*.MouseIcon = LoadPicture(*pathname*)

*object*.MouseIcon [= *picture*]

A sintaxe da propriedade **MouseIcon** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>pathname</i>	Uma <u>expressão de seqüência de caracteres</u> especificando o caminho e nome de arquivo do arquivo contendo o ícone personalizado.
<i>picture</i>	A propriedade <b>Picture</b> de um objeto <b>Form</b> , controle <b>PictureBox</b> ou controle <b>Image</b> .

### Comentários

A propriedade **MouseIcon** oferece um ícone personalizado que é usado quando a propriedade **MousePointer** é configurada como 99.

Embora o Visual Basic não crie ou suporte arquivos de cursores coloridos (.cur) como, por exemplo, aqueles que acompanham o Windows NT, você pode usar a propriedade **MouseIcon** para carregar arquivos de ícone ou de cursor. Os arquivos de cursor colorido como, por exemplo, aqueles que acompanham o Windows NT 3.51 são exibidos em preto e branco. Para exibir um cursor colorido, use o arquivo de ícone colorido (.ico). A propriedade **MouseIcon** oferece a seu programa um fácil acesso a cursores personalizados de qualquer tamanho, com qualquer localização desejada de ponto de acesso. O Visual Basic não carrega arquivos de cursor animado, embora as versões de 32 bits do Windows suportem estes cursores.

## Propriedade MousePointer

Retorna ou configura um valor indicando o tipo de ponteiro de mouse exibido quando o mouse está sobre uma determinada parte de um objeto durante o tempo de execução.

### Sintaxe

*object*.MousePointer [= *value*]

A sintaxe da propriedade **MousePointer** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Um número inteiro especificando o tipo de ponteiro de mouse exibido, conforme descrito em Configurações.

### Configurações

As configurações de *value* são:

Constante	Valor	Descrição
<b>vbDefault</b>	0	(Padrão) Forma determinada pelo objeto.
<b>VbArrow</b>	1	Seta.
<b>VbCrosshair</b>	2	Cruz (ponteiro em forma de cruz).
<b>Vblbeam</b>	3	Ponteiro em forma de I.
<b>VblconPointer</b>	4	Ícone (Pequeno quadrado dentro de um quadrado).
<b>VbSizePointer</b>	5	Dimensionar (seta de quatro pontas nas direções norte, sul, leste e oeste).
<b>VbSizeNESW</b>	6	Dimensionar NE SW (seta dupla apontando para nordeste e sudoeste).
<b>VbSizeNS</b>	7	Dimensionar N S (seta dupla apontando para o norte e sul).
<b>VbSizeNWSE</b>	8	Dimensionar NW SE (seta dupla apontando para noroeste e sudeste).
<b>VbSizeWE</b>	9	Dimensionar W E (seta dupla apontando para leste e oeste).

<b>VbUpArrow</b>	10	Seta para cima.
<b>VbHourglass</b>	11	Ampulheta (espera).
<b>VbNoDrop</b>	12	Não soltar.
<b>VbArrowHourglass</b>	13	Seta e ampulheta.
<b>vbArrowQuestion</b>	14	Seta e ponto de interrogação.
<b>vbSizeAll</b>	15	Dimensionar tudo.
<b>vbCustom</b>	99	Ícone personalizado especificada pela propriedade <b>MouseIcon</b> .

### Comentários

Você pode usar esta propriedade quando deseja indicar alterações no funcionamento à medida que o ponteiro do mouse passa sobre controles em um formulário ou caixa de diálogo. A configuração de ampulheta (11) é útil para indicar que o usuário deve aguardar que um processo ou operação termine.

**Observação:** Se o seu aplicativo chama DoEvents, a propriedade MousePointer pode se alterar temporariamente quando se encontra sobre um componente ActiveX.

## Propriedade MultiLine

Retorna ou configura um valor indicando se um controle TextBox pode aceitar e exibir linhas de texto múltiplas. Somente leitura durante o tempo de execução.

### Sintaxe

*object*.MultiLine

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Configurações

As configurações da propriedade **MultiLine** são:

Configuração	Descrição
<b>True</b>	Permite múltiplas linhas de texto.
<b>False</b>	(Padrão) Ignora mudanças de linha e restringe os dados a uma única linha.

### Comentários

Um controle **TextBox** de linhas múltiplas quebra as linhas de texto, à medida que o usuário digita o texto que se estende além da caixa de texto.

Você também pode adicionar barras de rolagem a controles **TextBox** maiores usando a propriedade **ScrollBars**. Se nenhuma barra de rolagem horizontal for especificada, o texto em uma **TextBox** de linhas múltiplas muda automaticamente de linha.

**Observação:** Em um formulário que não tenha nenhum botão padrão, o pressionamento de ENTER em um controle **TextBox** de linhas múltiplas move o foco para a linha seguinte. Se existir um botão padrão, você deve pressionar CTRL+ENTER para mover-se até a linha seguinte.

---

## Propriedade MultiSelect

Retorna ou configura um valor indicando se um usuário pode efetuar seleções múltiplas em um controle **FileListBox** ou **ListBox** e como as seleções múltiplas podem ser feitas. Somente leitura durante o tempo de execução.

### Sintaxe

*object*.MultiSelect

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Configurações

As configurações da propriedade **MultiSelect** são:

Configuração	Descrição
0	(Padrão) Não é permitida múltipla seleção.
1	Seleção múltipla simples. Um clique de mouse ou o pressionamento da BARRA DE ESPAÇOS seleciona ou anula a seleção de um item da lista. (Teclas de direção movem o foco.)
2	Seleção múltipla ampliada. O pressionamento de SHIFT e o clique do mouse, ou o pressionamento de SHIFT e uma das teclas de direção (SETA ACIMA, SETA ABAIXO, SETA À ESQUERDA e SETA À DIREITA) ampliam a seleção, do item anteriormente selecionado até o item atual. O pressionamento de CRL e o clique do mouse selecionam ou anulam a seleção de um item da lista.

## Propriedade NewIndex

Retorna o índice do item mais recentemente adicionado a um controle **ComboBox** ou **ListBox**. Somente leitura durante o tempo de execução.

### Sintaxe

*object*.NewIndex

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Você pode usar esta propriedade com listas classificadas quando precisa de uma lista de valores que correspondem a cada item na matriz de propriedades **ItemData**. À medida que você adiciona um item a uma lista classificada, o Visual Basic insere o item na lista em ordem alfabética. Esta propriedade lhe informa onde o item foi inserido, de modo que você possa inserir um valor correspondente na propriedade **ItemData** no mesmo índice.

A propriedade **NewIndex** retorna -1 se não existirem itens na lista ou se um item tiver sido excluído, desde que o último item foi adicionado.

---

## Propriedade Orientation

Retorna ou configura um valor indicando se os documentos são impressos em modo retrato ou paisagem. Não está disponível durante o tempo de criação.

### Sintaxe

*object.Orientation* [= *value*]

A sintaxe da propriedade **Orientation** tem estas partes:

Parte	Descrição
<i>Object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>Value</i>	Um valor ou constante que determina a orientação da página, conforme descrito em Configurações.

### Configurações

As configurações de *value* são:

Constante	Valor	Descrição
<b>vbPRORPortrait</b>	1	Os documentos são impressos com a parte superior no lado estreito do papel.
<b>vbPRORLandscape</b>	2	Os documentos são impressos com sua parte superior no lado largo do papel.

### Comentários

Estas constantes são listadas na biblioteca de objetos Visual Basic (VB) no Object Browser.

**Observação:** O efeito das propriedades do objeto **Printer** depende do driver fornecido pelo fabricante da impressora. Algumas configurações de propriedades podem não ter qualquer efeito, ou diversas configurações de propriedades podem ter todas o mesmo efeito. As configurações fora do intervalo aceitável podem ou não produzir um erro. Para maiores informações, consulte a documentação do fabricante do driver específico.

## Propriedade Page

Retorna o número de página atual.

### Sintaxe

*object.Page*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

O Visual Basic mantém uma contagem de páginas que foram impressas desde que seu aplicativo iniciou ou desde a última vez em que a instrução **EndDoc** foi utilizada no objeto **Printer**. Esta contagem inicia-se em um e é incrementada em um se:

- Você usa o método **NewPage**.
- Você usa o método **Print** e o texto que deseja imprimir não se ajusta à página atual.

**Observação:** Resultados de métodos gráficos que não se ajustem à página não geram uma nova página. O resultado é recortado para ajustar-se à área de impressão da página.

---

## Propriedade PaperBin

Retorna ou configura um valor indicando que a bandeja padrão de papel na impressora de onde o papel é alimentado durante a impressão. Não está disponível durante o tempo de criação.

### Sintaxe

*object.PaperBin [= value]*

A sintaxe da propriedade **PaperBin** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Um valor ou constante especificando a bandeja padrão de papel, conforme descrito em Configurações.

### Configurações

As configurações de *value* são:

Constante	Valor	Descrição
<b>vbPRBNUpper</b>	1	Usa papel da bandeja superior.
<b>VbPRBNLower</b>	2	Usa papel da bandeja inferior.
<b>VbPRBNMiddle</b>	3	Usa papel da bandeja central.
<b>VbPRBNManual</b>	4	Aguarda a inserção manual de cada folha de papel.
<b>VbPRBNEvelope</b>	5	Usa envelopes do alimentador de envelopes.
<b>VbPRBNEnvManual</b>	6	Usa envelopes do alimentador de envelopes, mas aguarda a inserção manual.
<b>VbPRBNAuto</b>	7	(Padrão) Usa papel da bandeja padrão atual.
<b>VbPRBNTractor</b>	8	Usa papel alimentado do alimentador por tração.
<b>VbPRBNSmallFmt</b>	9	Usa papel do alimentador de papel pequeno.
<b>VbPRBNLargeFmt</b>	10	Usa papel da bandeja de papel grande.
<b>VbPRBNLargeCapacity</b>	11	Usa papel do alimentador de grande capacidade.
<b>VbPRBNCassette</b>	14	Usa papel do cartucho acoplado.

### Comentários

Estas constantes estão listada na biblioteca de objetos Visual Basic (VB) no Object Browser.

Nem todas as opções de bandeja estão disponíveis em todas as impressoras. Verifique a documentação da impressora para descrições mais específicas destas opções.

**Observação:** O efeito das propriedades do objeto **Printer** depende do driver fornecido pelo fabricante da impressora. Algumas configurações de impressora podem não ter efeito algum, ou diversas configurações de propriedades diferentes podem ter todas o mesmo efeito. As configurações fora do intervalo aceito podem ou não produzir um erro. Para maiores informações, consulte a documentação do fabricante do driver específico.

## Propriedade PaperSize

Retorna ou configura um valor indicando o tamanho do papel para a impressora atual. Não está disponível durante o tempo de criação.

### Sintaxe

*object.PaperSize* [= *value*]

A sintaxe da propriedade **PaperSize** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Um valor ou constante especificando o tamanho do papel, conforme descrito em Configurações.

### Configurações

As configurações de *value* são:

Constante	Valor	Descrição
<b>vbPRPSLetter</b>	1	Carta, 8 1/2 x 11 pol.
<b>vbPRPSLetterSmall</b>	2	Carta pequeno, 8 1/2 x 11 pol.
<b>vbPRPSTabloid</b>	3	Tablóide, 11 x 17 pol.
<b>vbPRPSLedger</b>	4	Ficha contábil, 17 x 11 pol.
<b>vbPRPSLegal</b>	5	Ofício, 8 1/2 x 14 pol.
<b>vbPRPSStatement</b>	6	Declaração, 5 1/2 x 8 1/2 pol.
<b>vbPRPSExecutive</b>	7	Executivo, 7 1/2 x 10 1/2 pol.
<b>vbPRPSA3</b>	8	A3, 297 x 420 mm
<b>vbPRPSA4</b>	9	A4, 210 x 297 mm
<b>vbPRPSA4Small</b>	10	A4 Pequeno, 210 x 297 mm
<b>vbPRPSA5</b>	11	A5, 148 x 210 mm
<b>vbPRPSB4</b>	12	B4, 250 x 354 mm
<b>vbPRPSB5</b>	13	B5, 182 x 257 mm
<b>vbPRPSFolio</b>	14	Folio, 8 1/2 x 13 pol.
<b>vbPRPSQuarto</b>	15	Quarto, 215 x 275 mm
<b>vbPRPS10x14</b>	16	10 x 14 pol.
<b>vbPRPS11x17</b>	17	11 x 17 pol.
<b>vbPRPSNote</b>	18	Nota, 8 1/2 x 11 pol.
<b>vbPRPSEnv9</b>	19	Envelope #9, 3 7/8 x 8 7/8 pol.
<b>vbPRPSEnv10</b>	20	Envelope #10, 4 1/8 x 9 1/2 pol.
<b>vbPRPSEnv11</b>	21	Envelope #11, 4 1/2 x 10 3/8 pol.
<b>vbPRPSEnv12</b>	22	Envelope #12, 4 1/2 x 11 pol.
<b>vbPRPSEnv14</b>	23	Envelope #14, 5 x 11 1/2 pol.
<b>vbPRPSCSheet</b>	24	Folha tamanho C
<b>vbPRPSDSheet</b>	25	Folha tamanho D
<b>vbPRPSESheet</b>	26	Folha tamanho E
<b>vbPRPSEnvDL</b>	27	Envelope DL, 110 x 220 mm
<b>vbPRPSEnvC3</b>	29	Envelope C3, 324 x 458 mm
<b>vbPRPSEnvC4</b>	30	Envelope C4, 229 x 324 mm
<b>vbPRPSEnvC5</b>	28	Envelope C5, 162 x 229 mm
<b>vbPRPSEnvC6</b>	31	Envelope C6, 114 x 162 mm
<b>vbPRPSEnvC65</b>	32	Envelope C65, 114 x 229 mm
<b>vbPRPSEnvB4</b>	33	Envelope B4, 250 x 353 mm
<b>vbPRPSEnvB5</b>	34	Envelope B5, 176 x 250 mm
<b>vbPRPSEnvB6</b>	35	Envelope B6, 176 x 125 mm
<b>vbPRPSEnvItaly</b>	36	Envelope, 110 x 230 mm
<b>vbPRPSEnvMonarch</b>	37	Envelope Monarch, 3 7/8 x 7 1/2 pol.
<b>vbPRPSEnvPersonal</b>	38	Envelope, 3 5/8 x 6 1/2 pol.
<b>vbPRPSFanfoldUS</b>	39	Formulário contínuo padrão americano, 14 7/8 x 11 pol.
<b>vbPRPSFanfoldStdGerman</b>	40	Formulário contínuo padrão alemão, 8 1/2 x 12 pol.
<b>vbPRPSFanfoldLglGerman</b>	41	Formulário contínuo ofício alemão, 8 1/2 x 13 pol.
<b>vbPRPSUser</b>	256	Definido pelo usuário

---

## Comentários

Estas constantes estão listadas na [biblioteca de objetos](#) Visual Basic (VB) no [Object Browser](#).

Configurar a propriedade **Height** ou **Width** de uma impressora, configura automaticamente **PaperSize** como **vbPRPSUser**.

**Observação:** O efeito das propriedades do objeto **Printer** dependem do driver fornecido pelo fabricante da impressora. Algumas configurações de propriedades podem não ter efeito algum, ou diversas configurações diferentes podem ter todas o mesmo efeito. As configurações fora do intervalo aceito podem ou não produzir um erro. Para maiores informações, consulte a documentação do fabricante para o driver específico.

## Propriedade PasswordChar

Retorna ou configura um valor indicando se os caracteres digitados por um usuário ou caracteres de espaço reservado são exibidos em um controle **TextBox**; retorna ou configura o caractere usado como espaço reservado.

### Sintaxe

*object.PasswordChar* [= *value*]

A sintaxe da propriedade **PasswordChar** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <a href="#">expressão de objeto</a> que avalia para um objeto na lista Applies To.
<i>value</i>	Uma <a href="#">expressão de sequência de caracteres</a> especificando o caractere de espaço reservado.

### Comentários

Use esta propriedade para criar um campo de senha em uma caixa de diálogo. Embora você possa usar qualquer caractere, a maioria dos aplicativos do Windows usam o asterisco (\*) (**Chr**(42)).

Esta propriedade não afeta a propriedade **Text**; a propriedade **Text** contém exatamente o que o usuário digita ou o que foi configurado a partir do código. Configure **PasswordChar** como uma sequência de caracteres de comprimento zero (""), que é o padrão, para exibir o texto efetivo. Você pode atribuir qualquer sequência de caracteres a esta propriedade, mas somente o primeiro caracteres é significativo; todos os outros são ignorados.

**Observação:** Se a propriedade **MultiLine** for configurada como **True**, a configuração da propriedade **PasswordChar** não surtirá qualquer efeito.

## Propriedade Pattern

Retorna ou configura um valor indicando os nomes de arquivo exibidos em um controle **FileListBox** durante o tempo [de execução](#).

### Sintaxe

*object.Pattern* [= *value*]

A sintaxe da propriedade **Pattern** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <a href="#">expressão de objeto</a> que avalia para um objeto na lista Applies To.
<i>value</i>	Uma <a href="#">expressão de sequência de caracteres</a> indicando uma especificação de arquivo, por exemplo "*.*" ou "*.FRM". O padrão é "*.*", que retorna uma lista de todos os arquivos. Além de usar caracteres curinga, você também pode usar padrões múltiplos separados por ponto e vírgula (;). Por exemplo, "*.exe;*.bat" retornariam uma lista de todos os <a href="#">arquivos executáveis</a> e todos os arquivos de lote MS-DOS.

### Comentários

A propriedade **Pattern** desempenha um papel fundamental na criação de recursos de manipulação e procura de arquivos de um aplicativo. Use **Pattern** em combinação com outras propriedades de controle de arquivos para oferecer ao usuário maneiras de explorar arquivos ou grupos de arquivos semelhantes. Por exemplo, em um aplicativo dedicado ao acionamento de outros pro-



---

gramas, você poderia designar que somente arquivos .exe sejam exibidos na caixa de listagem de arquivos (\*.exe). Outras propriedades-chave de controle de arquivo incluem **Drive**, **FileName** e **Path**.

Alterar o valor da propriedade **Pattern** gera um evento **PatternChange**.

## Propriedade Port

Retorna o nome da porta através da qual o documento é enviado à impressora.

### Sintaxe

*object*.Port

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

O sistema operacional determina o nome da porta, por exemplo LPT1: ou LPT2:.

**Observação:** O efeito das propriedades do objeto **Printer** dependem do driver fornecido pelo fabricante da impressora. Algumas configurações de propriedades podem não ter efeito algum, ou diversas configurações de propriedades diferente podem ter todas o mesmo efeito. As configurações fora do intervalo aceito, podem ou não produzir um erro. Para maiores informações, consulte a documentação do fabricante do driver específico.

---

## Propriedade PrevInstance

Retorna um valor indicando se uma ocorrência anterior de um aplicativo já está sendo executada.

### Sintaxe

*object*.PrevInstance

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Você pode usar esta propriedade em um procedimento de evento Load para especificar se um usuário já está executando uma ocorrência de um aplicativo. Dependendo do aplicativo, você pode querer que apenas uma ocorrência em execução no ambiente operacional do Microsoft Windows a cada vez.

## Propriedade PrintQuality

Retorna ou configura um valor indicando a resolução da impressora. Não está disponível durante o tempo de criação.

### Sintaxe

*object*.PrintQuality [= *value*]

A sintaxe da propriedade **PrintQuality** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Um valor ou constante especificando a resolução da impressora, conforme descrito em Configurações.

### Configurações

As configurações de *value* são:

Constante	Valor	Descrição
<b>vbPRPQDraft</b>	-1	Resolução rascunho
<b>vbPRPQLow</b>	-2	Baixa resolução
<b>vbPRPQMedium</b>	-3	Resolução média
<b>vbPRPQHigh</b>	-4	Alta resolução

Além dos valores negativos predefinidos, você também pode configurar *value* como um valor positivo de pontos por polegada (dpi), por exemplo, 300.

### Comentários

Estas constantes estão listadas na biblioteca de objetos Visual Basic (VB) no Object Browser.

O valor padrão depende do driver da impressora e das configurações atuais da impressora. O efeito destas configurações variam entre impressoras e entre drivers de impressora. Em algumas impressoras, algumas ou todas as configurações podem produzir o mesmo resultado.

**Observação:** O efeito das propriedades do objeto **Printer** dependem do driver fornecido pelo fabricante da impressora. Algumas configurações de propriedades podem não ter efeito algum, e diversas configurações de propriedades podem ter todas o mesmo efeito. As configurações fora do intervalo aceito podem ou não produzir um erro. Para maiores informações, consulte a documentação do fabricante do driver específico.

---

## Propriedade RowHeight

Retorna ou configura a altura de todas as linhas no controle **DBGrid**. **RowHeight** é sempre a mesma unidade de medida que o recipiente do controle **DBGrid**.

### Sintaxe

*object.RowHeight*[= *value* ]

A sintaxe da propriedade **RowHeight** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Uma <u>expressão numérica</u> especificando a altura.

### Comentários

Os usuários podem alterar **RowHeight** de qualquer linha durante o tempo de execução, posicionando o ponteiro do mouse em uma linha de grade entre linhas e arrastando.

## Propriedades ScaleHeight e ScaleWidth

Retorna ou configura o número de unidades para as medidas horizontal (**ScaleWidth**) e vertical (**ScaleHeight**) do interior de um objeto ao usar métodos gráficos ou ao posicionar controles. Para objetos **MDIForm**, não está disponível durante o tempo de criação e somente leitura durante o tempo de execução.

### Sintaxe

*object.ScaleHeight* [= *value*]

*object.ScaleWidth* [= *value*]

As sintaxes das propriedades **ScaleHeight** e **ScaleWidth** têm estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Uma <u>expressão numérica</u> especificando a medida horizontal ou vertical.

### Comentários

Você pode usar estas propriedades para criar uma escala de coordenadas para desenho ou impressão. Por exemplo, a instrução `ScaleHeight = 100` altera as unidades de medida da altura interior efetiva do formulário. Ao invés da altura ser *n* unidades atuais (twips, pixels,...), a altura será 100 unidades definidas pelo usuário. Portanto, uma distância de 50 unidades corresponde à metade da altura/largura do objeto, e uma distância de 101 unidades estará fora do objeto por 1 unidade.

Use a propriedade **ScaleMode** para definir uma escala com base em uma unidade padrão de medida, por exemplo, twips, pontos, pixels, caracteres, polegadas, milímetros ou centímetros.

A configuração destas propriedades como valores positivos efetua aumentos de coordenadas de cima para baixo e da esquerda para a direita. Configurá-los com valores negativos efetua os aumentos de coordenadas de baixo para cima e da direita para a esquerda.

Usando estas propriedades e as propriedades **ScaleLeft** e **ScaleTop** relacionadas, você pode configurar um sistema completo de coordenadas como coordenadas positivas e negativas. Todas as quatro destas propriedades Scale interagem com a propriedade **ScaleMode** das seguintes formas:

- Configurar qualquer outra propriedade Scale como qualquer valor configura automaticamente **ScaleMode** como 0. Uma **ScaleMode** de 0 é definida pelo usuário.
- Configurar **ScaleMode** como um número maior que 0 altera **ScaleHeight** e **ScaleWidth** como a nova unidade de medida e configura **ScaleLeft** e **ScaleTop** como 0. Além disso, as configurações **CurrentX** e **CurrentY** alteram-se para refletir as novas coordenadas do ponto atual.

Você também pode usar o método **Scale** para configurar as propriedades **ScaleHeight**, **ScaleWidth**, **ScaleLeft** e **ScaleTop** em uma instrução.

**Observação:** As propriedades **ScaleHeight** e **ScaleWidth** não são as mesmas que as propriedades **Height** e **Width**.

Para objetos **MDIForm**, **ScaleHeight** e **ScaleWidth** referem-se somente à área não coberta pelos controles **PictureBox** no formulário. Evite usar estas propriedades para dimensionar uma **Pictu-**

## Propriedades ScaleLeft e ScaleTop

Retornam ou configuram as coordenadas horizontal (**ScaleLeft**) e vertical (**ScaleTop**) das bordas esquerda e superior de um objeto ao usar métodos gráficos ou ao posicionar controles.

### Sintaxe

*object*.**ScaleLeft** [= *value*]

*object*.**ScaleTop** [= *value*]

As sintaxes das propriedades **ScaleLeft** e **ScaleTop** têm estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Uma <u>expressão numérica</u> especificando uma coordenada horizontal ou vertical. O padrão é 0.

### Comentários

Usando estas propriedade e as propriedades **ScaleHeight** e **ScaleWidth** relacionadas, você pode configurar um sistema completo de coordenadas com coordenadas positivas e negativas. Estas quatro propriedades Scale interagem com a propriedade **ScaleMode** das seguintes formas:

- Configurar qualquer outra propriedade Scale como qualquer valor configura automaticamente **ScaleMode** como 0. Um **ScaleMode** de 0 é definido pelo usuário.
- Configurar a propriedade **ScaleMode** como um número maior que 0 altera **ScaleHeight** e **ScaleWidth** para a nova unidade de medida, e configura **ScaleLeft** e **ScaleTop** como 0. As configurações de propriedades **CurrentX** e **CurrentY** alteram-se para refletir as novas coordenadas do ponto atual.

Você também pode usar o método **Scale** para configurar as propriedades **ScaleHeight**, **ScaleWidth**, **ScaleLeft** e **ScaleTop** em uma instrução.

**Observação:** As propriedades **ScaleLeft** e **ScaleTop** não são as mesmas que as propriedades **Left** e **Top**.

## Propriedade ScaleMode

Retorna ou configura um valor indicando a unidade de medida para coordenadas de um objeto ao usar métodos gráficos ou ao posicionar controles.

### Sintaxe

*object*.**ScaleMode** [= *value*]

A sintaxe da propriedade **ScaleMode** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Um número inteiro especificando a unidade de medida, conforme descrito em Configurações.

### Configurações

As configurações de *value* são:

Constante	Configuração	Descrição
<b>vbUser</b>	0	Indica que uma ou mais das propriedades <b>ScaleHeight</b> , <b>ScaleWidth</b> , <b>ScaleLeft</b> e <b>ScaleTop</b> estão configuradas como valores personalizados.
<b>VbTwips</b>	1	(Padrão) <u>Twip</u> (1440 twips por polegada lógica; 567 twips por centímetro lógico).
<b>VbPoints</b>	2	<u>Ponto</u> (72 pontos por polegada lógica).
<b>VbPixels</b>	3	<u>Pixel</u> (menor unidade de resolução de monitor ou impressora).
<b>vbCharacters</b>	4	Caractere (horizontal = 120 twips por unidade; vertical = 240 twips por uni-

<b>VbInches</b>	5	dade).
<b>VbMillimeters</b>	6	Polegada.
<b>VbCentimeters</b>	7	Milímetro.
		Centímetro.

### Comentários

Usando as propriedades **ScaleHeight**, **ScaleWidth**, **ScaleLeft** e **ScaleTop**, você pode criar um sistema de coordenadas personalizado com coordenadas positivas e negativas. Estas quatro propriedades Scale interagem com a propriedade **ScaleMode** das seguintes formas:

- Configurar o valor de qualquer outra propriedade Scale como qualquer valor, configura automaticamente **ScaleMode** como 0. Uma **ScaleMode** de 0 é definida pelo usuário.
- Configurar a propriedade **ScaleMode** como um número maior que 0 altera **ScaleHeight** e **ScaleWidth** para a nova unidade de medida e configura **ScaleLeft** e **ScaleTop** como 0. As configurações de propriedades **CurrentX** e **CurrentY** alteram-se para refletir as novas coordenadas do ponto atual.

## Propriedade ScrollBars

Retorna ou configura um valor indicando se um objeto tem barras de rolagem horizontal ou vertical. Somente leitura durante o tempo de execução.

### Sintaxe

*object.ScrollBars*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Configurações

Para um objeto **MDIForm**, as configurações da propriedade **ScrollBars** são:

Configuração	Descrição
<b>True</b>	(Padrão) O formulário tem uma barra de rolagem horizontal ou vertical, ou ambas.
<b>False</b>	O formulário não tem barra de rolagem.

Para um controle **TextBox** as configurações de propriedades de **ScrollBars** são:

Constante	Configuração	Descrição
<b>vbSBNone</b>	0	(Padrão) Nenhuma
<b>vbHorizontal</b>	1	Horizontal
<b>vbVertical</b>	2	Vertical
<b>vbBoth</b>	3	Ambas

### Comentários

Para um controle **TextBox** com a configuração 1 (Horizontal), 2 (Vertical), ou 3 (Ambos), você deve configurar a propriedade **MultiLine** como **True**.

Durante o tempo de execução, o ambiente operacional do Microsoft Windows implementa automaticamente uma interface de teclado padrão para permitir a navegação em controles **TextBox** com as teclas de direção (SETA ACIMA, SETA ABAIXO, SETA À ESQUERDA, e SETA À DIREITA), as teclas HOME e END, e assim por diante.

Barras de rolagem somente são exibidas em um objeto se o seu conteúdo se estende para além das bordas do objeto. Por exemplo, em um objeto **MDIForm**, se parte de um formulário filho estiver oculto atrás da borda do formulário MDI pai, será exibida uma barra de rolagem horizontal (controle **HScrollBar**). De maneira semelhante, aparecerá uma barra de rolagem vertical em um controle **TextBox** quando ele não pode exibir todas as suas linhas de texto. Se **ScrollBars** for configurada como **False**, o objeto não terá barras de rolagem, independente de seu conteúdo.

## Propriedade SelCount

Retorna o número de itens selecionados em um controle **ListBox**.

### Sintaxe

*object.SelCount*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

---

## Comentários

A propriedade **SelCount** retorna 0 se nenhum item estiver selecionado. Caso contrário, retorna o número de itens da lista atualmente selecionada. Esta propriedade é particularmente útil quando usuários podem efetuar seleções múltiplas.

## Propriedade Selected

Retorna ou configura o status da seleção de um item em um controle **FileListBox** ou **ListBox**. Esta propriedade é uma matriz de valores Boolean com o mesmo número de itens que a propriedade **List**. Não está disponível durante o tempo de criação.

### Sintaxe

*object.Selected(index) [= boolean]*

A sintaxe da propriedade **Selected** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	O número de índice do item no controle.
<i>Boolean</i>	Uma <u>expressão Booleana</u> especificando se o item está selecionado, conforme descrito em Configurações.

### Configurações

As configurações de *boolean* são:

Configuração	Descrição
<b>True</b>	O item está selecionado.
<b>False</b>	(Padrão) O item não está selecionado.

### Comentários

Esta propriedade é particularmente útil quando os usuários podem efetuar seleções múltiplas. Você pode verificar rapidamente quais itens estão selecionados em uma lista. Você também pode usar esta propriedade para selecionar ou anular a seleção de itens em uma lista a partir do código.

Se a propriedade **MultiSelect** for configurada como 0, você pode usar a propriedade **ListIndex** para obter o índice do item selecionado. Entretanto, em uma seleção múltipla, a propriedade **ListIndex** retorna o índice do item contido no retângulo de foco, esteja o item efetivamente selecionado ou não.

Se a propriedade **Style** de um controle **ListBox** for configurada como 1 (caixas de seleção), a propriedade **Selected** retorna **True** somente para aqueles itens cujas caixas de seleção estão selecionadas. A propriedade **Selected** não retornará **True** para estes itens que estão apenas ~~r~~alçados.

## Propriedades SelEndCol, SelStartCol, SelEndRow e SelStartRow

Retornam ou configuram a primeira ou última linha ou coluna de um intervalo de células. Não estão disponíveis durante o tempo de criação.

- **SelEndCol** — a última coluna selecionada à direita.
- **SelStartCol** — a primeira coluna selecionada à esquerda.
- **SelEndRow** — a última linha selecionada.
- **SelStartRow** — A primeira linha selecionada.

### Sintaxe

*object.SelEndCol [= value]*

*object.SelStartCol [= value]*

*object.SelEndRow [= value]*

*object.SelStartRow [= value]*

As sintaxes das propriedades **SelEndCol**, **SelStartCol**, **SelEndRow** e **SelStartRow** têm estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Uma <u>expressão numérica</u> especificando a primeira ou a última coluna ou linha.

---

### Comentários

Você pode usar estas propriedades para selecionar uma área específica de um controle **DBGrid** a partir de código ou retornar em código as dimensões de uma área selecionada pelo usuário.

**SelStartCol** e **SelStartRow** juntos especificam a célula no canto superior esquerdo de um intervalo selecionado. **SelEndCol** e **SelEndRow** especificam o canto inferior direito de um intervalo selecionado.

Para especificar uma célula sem mover a seleção atual, use as propriedades **Col** e **Row**.

O valor padrão para **SelStartCol** e **SelEndCol** é -1.

## Propriedades SelLength, SelStart e SelText

- **SelLength** — retorna ou configura o número de caracteres selecionados.
- **SelStart** — retorna ou configura o ponto inicial do texto selecionado; indica a posição do ponto de inserção se nenhum texto estiver realçado.
- **SelText** — retorna ou configura a seqüência de caracteres contendo o texto atualmente selecionado; consiste em uma seqüência de caracteres de comprimento zero ("") se nenhum caractere estiver realçado.

Estas propriedades não estão disponíveis durante o tempo de criação.

### Sintaxe

*object.SelLength* [= *number*]

*object.SelStart* [= *index*]

*object.SelText* [= *value*]

As sintaxes das propriedades **SelLength**, **SelStart** e **SelText** têm estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Uma <u>expressão numérica</u> especificando o número de caracteres selecionados. Para <b>SelLength</b> e <b>SelStart</b> , o intervalo de configurações vai de 0 até o comprimento do texto — o número total de caracteres na área de edição de um controle <b>ComboBox</b> ou <b>TextBox</b> .
<i>index</i>	Uma expressão numérica especificando o ponto inicial do texto selecionado, conforme descrito em Configurações.
<i>value</i>	Uma <u>expressão de seqüência de caracteres</u> contendo o texto selecionado.

### Comentários

Use estas propriedades para tarefas do tipo configurar o ponto de inserção, estabelecer um intervalo de inserção, selecionar subsequências de caracteres em um controle, ou apagar texto. Usado em conjunto com o objeto **Clipboard**, estas propriedades são úteis para operações de cópia, recorte e colagem.

Ao trabalhar com estas propriedades:

- Configurar **SelLength** como menor que 0 provoca um erro durante o tempo de execução.
- Configurar **SelStart** como maior que o comprimento do texto configura a propriedade para o comprimento do texto existente; alterar **SelStart** altera a seleção até um ponto de inserção e configura **SelLength** como 0.
- Configurar **SelText** como um novo valor, configura **SelLength** como 0 e substitui o texto selecionado pela nova seqüência de caracteres.

## Propriedade Shape

Retorna ou configura um valor indicando a aparência de um controle **Shape**.

### Sintaxe

*object.Shape* [= *value*]

A sintaxe da propriedade **Shape** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Um número inteiro especificando a aparência do controle, conforme descrito em Configurações.

### Configurações

As configurações de *value* são:

Constante	Configuração	Descrição
<b>vbShapeRectangle</b>	0	(Padrão) Retângulo



<b>vbShapeSquare</b>	1	Quadrado
<b>vbShapeOval</b>	2	Oval
<b>vbShapeCircle</b>	3	Círculo
<b>VbShapeRoundedRectangle</b>	4	Retângulo arredondado
<b>VbShapeRoundedSquare</b>	5	Quadrado arredondado

## Propriedade Shortcut

Configura um valor que especifica uma tecla de atalho para um objeto **Menu**. Não está disponível durante o tempo de execução.

### Comentários

Use esta propriedade para oferecer atalhos de teclado para comandos de menu. Você pode configurar esta propriedades usando o Menu Editor. Para uma lista de teclas de atalho que você pode usar, verifique a lista Shortcut no Menu Editor.

**Observação:** Além das teclas de atalho, você também pode designar teclas de acesso a comandos, menus e controles usando um E comercial (&) na configuração de propriedade **Caption**.

## Propriedade Stretch

Retorna ou configura um valor indicando se um elemento gráfico é redimensionado para ajustar-se ao tamanho de um controle **Image**.

### Sintaxe

*object.Stretch* [= *boolean*]

A sintaxe da propriedade **Stretch** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> especificando se o elemento gráfico redimensionado, conforme descrito em Configurações.

### Configurações

As configurações de *boolean* são:

Configuração	Descrição
<b>True</b>	O elemento gráfico é redimensionado para ajustar-se ao controle.
<b>False</b>	(Padrão) O controle é redimensionado para ajustar-se ao gráfico.

### Comentários

Se **Stretch** está configurada como **True**, o redimensionamento também redimensiona o elemento gráfico nela contida.

## Propriedade Style

Retorna ou configura um valor indicando o tipo de exibição e comportamento do controle. Somente leitura durante o tempo de execução.

### Sintaxe

*object.Style*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Configurações

As configurações de propriedades **Style** para os controles **Checkbox**, **CommandButton**, e **OptionButton** são:

Constante	Valor	Descrição
<b>vbButtonStandard</b>	0	(Padrão) Padrão. O controle é exibido como era em versões anteriores do Visual Basic. Isto é, um controle <b>CheckBox</b> é exibido como uma caixa de seleção com um rótulo junto a ele, um <b>OptionButton</b> como um botão de opção com um rótulo junto a ele, e um <b>CommandButton</b> como <b>CommandButton</b> padrão sem um elemento gráfico associado.
<b>vbButtonGraphical</b>	1	Gráfico. O controle é exibido em estilo gráfico. Isto é, um controle <b>CheckBox</b> é exibido como um botão do tipo <b>CommandButton</b> que permanece pressionado ou solto até que outro <b>OptionButton</b> em seu grupo de opções seja selecionado em um <b>CommandButton</b> é exibido como um <b>CommandButton</b> padrão que também pode ser exibido como um gráfico associado.

As configurações de propriedades **Style** para o controle **ComboBox** são:

Constante	Valor	Descrição
<b>vbComboDropDown</b>	0	(Padrão) Caixa de combinação drop-down. Inclui uma listagem drop-down em uma caixa de texto. O usuário pode selecionar na lista ou digitar na caixa de texto.
<b>vbComboSimple</b>	1	Caixa de combinação simples. Inclui uma caixa de texto e uma lista que não funciona como drop-down. O usuário pode selecionar na lista ou digitar na caixa de texto. O tamanho de uma caixa de combinação <b>Simple</b> inclui tanto a parte de lista quanto de edição. Como padrão, uma caixa de combinação <b>Simple</b> é dimensionada, de modo que nada da lista é exibido. Aumente a propriedade <b>Height</b> para exibir mais da lista.
<b>vbComboDropDownList</b>	2	Lista drop-down. Este estilo permite apenas a seleção na lista drop-down.

As configurações das propriedades **Style** para o controle **ListBox** são:

Constante	Valor	Descrição
<b>vbListBoxStandard</b>	0	(Padrão) Padrão. O controle <b>ListBox</b> é exibido como foi em versões anteriores do Visual Basic; isto é, como uma lista de itens de texto.
<b>vbListBoxCheckbox</b>	1	<b>CheckBox</b> . O controle <b>ListBox</b> é exibido com uma caixa de seleção junto a cada item de texto. Múltiplos itens na <b>ListBox</b> podem ser selecionados realçando-se a caixa de seleção ao lado dela.

### Comentários

Para o controle **ComboBox**, siga estas orientações ao decidir qual configuração escolher:

- Use a configuração 0 (Dropdown Combo) ou a configuração 1 (Simple Combo) para dar ao usuário uma lista de escolhas. Qualquer um dos estilos permite ao usuário inserir uma alternativa na caixa de texto. A configuração 0 economiza espaço no formulário pois a parte da lista se fecha quando o usuário seleciona um item.
- Use a configuração 2 (Dropdown List) para exibir uma lista fixa de alternativas de onde o usuário pode selecionar. A parte de lista se fecha quando o usuário seleciona um item.

---

## Propriedade TabStop

Retorna ou configura um valor indicando se um usuário pode usar a tecla TAB para dar o foco a um objeto.

### Sintaxe

*object.TabStop* [= *boolean*]

A sintaxe da propriedade **TabStop** tem estas três partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> especificando se o objeto é uma parada de tabulação, conforme descrito em Configurações.

### Configurações

As configurações de *boolean* são:

Configuração	Descrição
<b>True</b>	(Padrão) Designa o objeto como uma parada de tabulação.
<b>False</b>	Salta o objeto quando o usuário está usando a tecla TAB, embora o objeto ainda mantenha seu lugar na ordem de tabulação efetiva, conforme determinado pela propriedade <b>TabIndex</b> .

### Comentários

Esta propriedade lhe permite adicionar ou remover um controle da ordem de tabulação em um formulário. Por exemplo, se você está usando um controle **PictureBox** para desenhar um gráfico, configure sua propriedade **TabStop** como **False**, assim o usuário não pode passar para **PictureBox** usando a tecla TAB.

## Propriedade Title

Retorna ou configura o título do aplicativo que é exibido na **Lista de Tarefas** do Microsoft Windows. Se alterado durante o tempo de execução, as alterações não são salvas com o aplicativo.

### Sintaxe

*object.Title* [= *value*]

A sintaxe da propriedade **Title** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Uma <u>expressão de sequência de caracteres</u> especificando o título do aplicativo. O comprimento máximo de <i>value</i> é 40 caracteres. Em sistemas DBCS (conjunto de caracteres de byte duplo), isto significa o comprimento máximo é 40 bytes.

### Comentários

Esta propriedade está disponível durante o tempo de criação na caixa de diálogo para o comando **Project Properties** no menu **Project**.

## Propriedade TopIndex

Retorna ou configura um valor que especifica qual item em um controle **ComboBox**, **DirListBox**, **DriveListBox**, **FileListBox** ou **ListBox** está em exibição em primeiro plano. Não está disponível durante o tempo de criação.

### Sintaxe

*object.TopIndex* [= *value*]

A sintaxe da propriedade **TopIndex** tem estas partes:

---

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	O número do item de lista que é exibido em primeiro lugar. O padrão é 0, ou o primeiro item da lista.

#### **Comentários**

Use esta propriedade para rolar por um controle sem selecionar um item.

Se a propriedade **Columns** é configurada como 0 para o controle **ListBox**, o item é exibido em primeiro lugar, se existirem itens suficientes abaixo dele para preencher a parte visível da lista.

Se a configuração da propriedade **Columns** for maior que 0 para o controle **ListBox**, a coluna do item move-se para a extremidade esquerda sem alterar sua posição dentro da coluna.

---

## Propriedades TwipsPerPixelX, TwipsPerPixelY

Retorna o número de twips por pixel para um objeto medido horizontalmente (**TwipsPerPixelX**) ou verticalmente (**TwipsPerPixelY**).

### Sintaxe

*object*.**TwipsPerPixelX**

*object*.**TwipsPerPixelY**

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Rotinas de API Windows geralmente exigem medidas em pixels. Você pode usar estas propriedades para converter rapidamente medidas sem alterar a configuração da propriedade **ScaleMode** de um objeto.

## Propriedade WindowList

Retorna ou configura um valor que determina se um objeto **Menu** mantém uma lista das janelas MDI filho atuais em um objeto **MDIForm**. Somente leitura durante o tempo de execução.

### Sintaxe

*object*.**WindowList**

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Configurações

As configurações da propriedade **WindowList** são:

Configuração	Descrição
<b>True</b>	O objeto <b>Menu</b> mantém uma lista de janelas abertas e exibe uma marca de seleção junto à janela ativa. Os usuários podem clicar no nome da janela para ativar aquela janela.
<b>False</b>	(Padrão) O <b>Menu</b> não mantém uma lista de janelas abertas.

### Comentários

Muitos aplicativos de interface de documento múltiplo (MDI) como, por exemplo, o Microsoft Excel e o Microsoft Word for Windows, têm um menu **Windows** contendo uma lista de janelas MDI f-lhas abertas. Esta propriedade permite adicionar esta funcionalidade a seu aplicativo.

Somente um objeto **Menu** em um formulário tem sua propriedade **WindowList** configurada como **True**.

Quando você seleciona a caixa de seleção **WindowList** no Menu Editor para um objeto **Menu**, a lista de janelas MDI filho abertas para o menu que você está criando é exibida.

---

## Propriedade WindowState

Retorna ou configura um valor indicando o estado visual de uma janela de formulário durante o tempo de execução.

### Sintaxe

*object.WindowState* [= *value*]

A sintaxe da propriedade **WindowState** tem estas partes:

Parte	Descrição
<i>Object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Um número inteiro especificando o estado do objeto, conforme descrito em Configurações.

### Configurações

As configurações de *value* são:

Constante	Valor	Descrição
<b>vbNormal</b>	0	(Padrão) Normal.
<b>VbMinimized</b>	1	Minimizado (minimizado como ícone)
<b>VbMaximized</b>	2	Maximizado (ampliando até o tamanho máximo)

### Comentários

Antes que um formulário seja exibido, a propriedade **WindowState** é sempre configurada como Normal (0), independente de sua configuração inicial. Isto é refletido nas configurações de propriedades **Height**, **Left**, **ScaleHeight**, **ScaleWidth**, **Top** e **Width**. Se um formulário estiver oculto após ter sido exibido, estas propriedades refletem o estado anterior até que o formulário seja novamente exibido, independente de qualquer alteração efetuada na propriedade **WindowState** neste meio tempo.

## Propriedade WordWrap

Retorna ou configura um valor indicando se um controle **Label** com sua propriedade **AutoSize** configurada como **True** é ampliada vertical ou horizontalmente para ajustar-se ao texto especificado em sua propriedade **Caption**.

### Sintaxe

*object.WordWrap* [= *boolean*]

A sintaxe da propriedade **WordWrap** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> especificando se o <b>Label</b> se amplia para ajustar ao texto, conforme descrito em Configurações.

### Configurações

As configurações de *boolean* são:

Configuração	Descrição
<b>True</b>	O texto muda de linha; o controle <b>Label</b> se amplia ou contrai verticalmente para ajustar o texto e o tamanho à fonte. O tamanho horizontal não se altera.
<b>False</b>	(Padrão) O texto não muda de linha; o <b>Label</b> se amplia ou contrai horizontalmente para ajustar-se ao comprimento do texto, e verticalmente para ajustar-se ao tamanho da fonte e ao número de linhas.

### Comentários

Use esta propriedade para determinar como um controle **Label** exibe seu conteúdo. Por exemplo, um gráfico que se altera dinamicamente pode ter um **Label** contendo texto que também se altera.

---

Para manter um tamanho horizontal constante para o **Label** e permitir que o texto aumente e diminua, configure as propriedades **WordWrap** e **AutoSize** como **True**. Se você deseja que um controle **Label** se expanda somente horizontalmente, configure **WordWrap** como **False**. Se não quiser que **Label** mude de tamanho, configure **AutoSize** como **False**. **Observação:** Se **AutoSize** for configurada como **False**, o texto sempre muda de linha, independente do tamanho do controle **Label** ou da configuração da propriedade **WordWrap**. Isto pode obscurecer uma parte do texto porque o **Label** não se amplia em nenhuma direção.

## Propriedades X1, Y1, X2, Y2

Retornam ou configuram as coordenadas do ponto inicial (X1, Y1) e do ponto final (X2, Y2) de um controle **Line**. As coordenadas horizontais são X1 e X2; as coordenadas verticais são Y1 e Y2.

### Sintaxe

*object.X1* [= *value*]

*object.Y1* [= *value*]

*object.X2* [= *value*]

*object.Y2* [= *value*]

As sintaxes das propriedades **X1**, **Y1**, **X2**, e **Y2** têm estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Uma <u>expressão numérica</u> especificando uma coordenada.

### Comentários

Use estas propriedades para ampliar dinamicamente um controle **Line** de um ponto a outro durante o tempo de execução. Por exemplo, você pode mostrar os relacionamentos entre os item em uma lista e itens em outra lista ou conecta pontos em um mapa.

## Exemplo da propriedade MaxLength

Este exemplo utiliza um valor numérico em um controle **TextBox** para limitar o comprimento de texto em outro controle **TextBox**. Para experimentar este exemplo cole o código na seção **Declarations** de um formulário que contenha dois controles **TextBox**. Torne **Text1** bastante grande e, em seguida, pressione F5. Digite um número em **Text2** e texto em **Text1**.

```
Private Sub Text1_Change ()  
    Text1.MaxLength = Text2.Text  
End Sub
```

---

## Exemplo da propriedade MDIChild

Este exemplo cria uma segunda ocorrência de um formulário MDI filho dentro de um objeto **MDI-Form**. Para experimentar este exemplo, configure a propriedade **MDIChild** como **True** em Form1 e, em seguida, crie um objeto MDIForm com o comando **Add MDI Form** no menu **Project**. Cole o código na seção Declarations do **MDIForm** e, em seguida, pressione F5 para executar o programa.

```
Private Sub MDIForm_Load ()
    Dim NewForm As New Form1 ' Declarar novo formulário.
    NewForm.Show ' Exibir novo formulário.
End Sub
```

## Exemplo da propriedade MouseIcon

Este exemplo ilustra como a propriedade **MouseIcon** configura um ícone de mouse personalizado. Para experimentar este exemplo, crie um controle **ListBox** em um formulário e, em seguida, configure a propriedade **MultiSelect** como 1 ou 2. Durante o tempo de execução, selecione um ou mais itens. Ícones diferentes aparecerão, dependendo se você selecionou um único item ou múltiplos itens.

```
Private Sub Form_Load ()
    ' Colocar alguns itens na ListBox.
    List1.AddItem "Selection 1"
    List1.AddItem "Selection 2"
    List1.AddItem "Selection 3"
    List1.AddItem "Selection 4"
    List1.AddItem "Selection 5"
End Sub

Private Sub List1_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
    ' Configurar o ícone de mouse personalizado para itens múltiplos.
    If List1.SelCount > 1 Then
        List1.MouseIcon = LoadPicture("ICONS\COMPUTER\MOUSE04.ICO")
        List1.MousePointer = 99
    Else ' Configurar o ícone de mouse personalizado para um único item.
        List1.MouseIcon = LoadPicture("ICONS\COMPUTER\MOUSE02.ICO")
        List1.MousePointer = 99
    End If
End Sub
```

## Exemplo da propriedade MousePointer

Este exemplo transforma o ponteiro do mouse em uma ampulheta, enquanto círculos são desenhados na tela e, em seguida, altera a ampulheta de volta para um ponteiro ao final do procedimento. Para experimentar este exemplo cole o código na seção Declarations de um formulário. Pressione F5 para executar o programa e, em seguida, clique no formulário.

```
Private Sub Form_Click ()
    Dim I ' Declarar variável.
    ' Alterar o ponteiro do mouse para uma ampulheta.
    Screen.MousePointer = vbHourglass
    ' Configurar cores aleatórias e desenhar círculos no formulário.
    For I = 0 To ScaleWidth Step 50
        ForeColor = RGB(Rnd * 255, Rnd * 255, Rnd * 255)
        Circle (I, ScaleHeight * Rnd), 400
    Next
    ' Retornar o ponteiro do mouse ao normal. Screen.MousePointer = vbDefault
End Sub
```



---

## Exemplo da propriedade MultiSelect

Este exemplo preenche um controle **ListBox** com os nomes de suas fontes de tela e ilustram como a propriedade **MultiSelect** afeta o comportamento de um **ListBox**. Para experimentar este exemplo, crie dois controles **ListBox** em um controle **CommandButton** em um formulário. No primeiro **ListBox**, configure a propriedade **MultiSelect** como 1 ou 2. Durante o tempo de execução, selecione diversos itens no primeiro **ListBox** e, em seguida, clique no **CommandButton**. Todos os itens selecionados são exibidos no segundo **ListBox**. Execute o exemplo diversas vezes com configurações diferentes da propriedade **MultiSelect**. Cole o código na seção **Declarations** e, em seguida, pressione F5 para executar o programa.

```
Private Sub Form_Load ()
    Dim I ' Declara variável.
    ' Preencher a caixa de listagem com nomes de fonte de tela.
    For I = 0 To Screen.FontCount - 1
        List1.AddItem Screen.Fonts(I)
    Next I
End Sub

Private Sub Command1_Click ()
    Dim I ' Declarar variável.
    ' Limpar todos os itens da lista.
    List2.Clear
    ' Se um item estiver selecionado, adicione-o a List2.
    For I = 0 To List1.ListCount - 1
        If List1.Selected(I) Then
            List2.AddItem List1.List(I)
        End If
    Next I
End Sub
```

## Exemplo da propriedade Page

Este exemplo imprime três páginas de texto com o número de página atual na parte superior de cada página. Para experimentar este exemplo cole o código na seção **Declarations** de um formulário e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Click ()
    Dim Header, I, Y ' Declarar variáveis.
    Print "Now printing..." ' Colocar nota em formulário.
    Header = "Printing Demo-Page" ' Configurar a seqüência de caracteres de cabeçalho.
    For I = 1 To 3
        Printer.Print Header; ' Imprimir o cabeçalho.
        Printer.Print Printer.Page ' Imprimir o número da página.
        Y = Printer.CurrentY + 10 ' Configurar a posição para linha.
        ' Desenhar uma linha na página.
        Printer.Line (0, Y) - (Printer.ScaleWidth, Y) ' Desenhar linha.
        For = 1 To 50
            Printer.Print String(, " "); ' Imprimir seqüência de caracteres de espaços.
            Printer.Print "Visual Basic "; ' Imprimir texto.
            Printer.Print Printer.Page ' Imprimir o número de página.
        Next
        Printer.NewPage
    Next I
    Printer.EndDoc
End Sub
```

---

## Exemplo da propriedade PasswordChar

Este exemplo ilustra como a propriedade **PasswordChar** afeta a maneira como um controle **TextBox** exibe texto. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha um **TextBox** e, em seguida, pressione F5 e clique no formulário. Cada vez que você clica no formulário, o texto alterna entre um caractere de senha asterisco (\*) e texto normal.

```
Private Sub Form_Click ()
    If Text1.PasswordChar = "" Then
        Text1.PasswordChar = "*"
    Else
        Text1.PasswordChar = ""
    End If
End Sub
```

## Exemplo da propriedade Pattern

Este exemplo atualiza um controle **TextBox** com o novo padrão selecionado em um controle **FileListBox**. Os controles são configurados de tal maneira que quando o usuário insere um padrão no **TextBox**, por exemplo, \*.txt, este é refletido no **FileListBox**, de maneira muito semelhante à interação que você vê em uma caixa de diálogo **File Open** típica em um aplicativo do Windows. Se um caminho inteiro como, por exemplo, C:\Bin\\*.exe for inserido no controle **TextBox**, o texto é automaticamente analisado em caminho e componentes padrão pelo controle **FileListBox**. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha os seguintes controles: um **DirListBox**, um **FileListBox**, um **TextBox** e um **CommandButton**. Pressione F5 e digite um padrão de arquivo válido no **TextBox**.

```
Private Sub Form_Load ()
    Command1.Default = True ' Configurar a propriedade Default.
End Sub

Private Sub Command1_Click ()
    ' O texto é analisado nos componentes caminho e padrão.
    File1.FileName = Text1.Text
    Dir1.Path = File1.Path ' Configurar caminho do diretório.
End Sub

Private Sub File1_PatternChange ()
    Text1.Text = File1.Pattern ' Configurar texto como novo padrão.
End Sub

Private Sub Dir1_Change
    File1.Path = Dir1.Path ' Configurar caminho da caixa de listagem de arquivos.
End Sub
```

## Exemplo da propriedade Port

Este exemplo examina cada objeto **Printer** na coleção **Printers** para descobrir uma conectada a uma porta específica e torná-la a impressora padrão.

```
Dim P As Object
For Each P In Printers
    If P.Port = "LPT2:" Or P.DeviceName Like "*LaserJet*" Then
        Set Printer = P
        Exit For
    End If
Next P
```

---

## Exemplo da propriedade RowHeight

Este exemplo configura a altura da linha atual como 500 twips quando você clica no formulário. Para experimentar este exemplo, crie um novo projeto, use a caixa de diálogo **Components** para adicionar um controle **DBGrid** à caixa de ferramentas (no menu **Project**, escolha **Components** e, em seguida, selecione **Microsoft Data Bound Grid Control**) e, em seguida, desenhe um controle **DBGrid**. Cole o código na seção Declarations do formulário, pressione F5 para executar o programa e, em seguida, selecione uma célula e clique no formulário.

```
Private Sub Form_Load ()
    DBGrid1.Rows = 5 ' Configurar colunas e linhas.
    DBGrid1.Cols = 7
End Sub

Private Sub Form_Click ()
    DBGrid1.RowHeight(DBGrid1.Row) = 500
End Sub
```

## Exemplo das propriedades ScaleHeight, ScaleWidth

Este exemplo usa as propriedades **ScaleHeight** e **ScaleWidth** para alterar as unidades de medida vertical e horizontal de um formulário. Para experimentar este exemplo, cole o código da seção Declarations de um formulário e, em seguida, pressione F5. Para ver os efeitos, clique no formulário, redimensione-o e, em seguida, clique novamente nele.

```
Private Sub Form_Click ()
    Dim Radius As Integer ' Declarar variável.
    ScaleHeight = 100 ' Configurar unidades de altura.
    ScaleWidth = 100 ' Configurar unidades de largura.
    For Radius = 5 to 50 Step 5
        FillStyle = 1
        Circle (50, 50), Radius ' Desenhar círculo.
    Next Radius
End Sub
```

## Exemplo das propriedades ScaleLeft, ScaleTop

Este exemplo cria uma grade em um controle **PictureBox** e configura coordenadas para o canto superior esquerdo como -1,-1, ao invés de 0,0. A cada 0,25 segundos, pontos são aleatoriamente plotados, do canto superior esquerdo até o canto inferior direito. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha um grande **PictureBox** e um controle **Timer** e, em seguida, pressione F5.

```
Private Sub Form_Load ()
    Timer1.Interval = 250 ' Configurar o intervalo do Timer.
    Picture1.ScaleTop = -1 ' Configurar a escala para a parte superior da grade.
    Picture1.ScaleLeft = -1 ' Configurar a escala para a parte esquerda da grade.
    Picture1.ScaleWidth = 2 ' Configurar a escala (-1 a 1).
    Picture1.ScaleHeight = 2
    Picture1.Line (-1, 0)-(1, 0) ' Desenhar linha horizontal.
    Picture1.Line (0, -1)-(0, 1) ' Desenhar linha vertical.
End Sub

Private Sub Timer1_Timer ()
    Dim I ' Declarar variável.
    ' Plotar pontos de maneira aleatória dentro de um intervalo.
    For I = -1 To 1 Step .05
        Picture1.PSet (I * Rnd, I * Rnd) ' Desenhar um ponto.
    Next I
End Sub
```

---

## Exemplo da propriedade ScaleMode

Este exemplo mostra como configurações diferentes da propriedade **ScaleMode** alteram o tamanho de um círculo. Para experimentar este exemplo, cole o código na seção Declarations de um formulário e, em seguida, pressione F5 e clique no formulário. Quando você clica no formulário, a unidade de medida se altera para a próxima configuração de **ScaleMode** e um círculo é desenhado no formulário.

```
Private Sub Form_Click ()
    ' Cicle através de cada uma das sete configurações.
    ScaleMode = ((ScaleMode + 1) Mod 7) + 1
    ' Desenhar um círculo com o raio 2 no centro do formulário.
    Circle (ScaleWidth / 2, ScaleHeight / 2), 2
End Sub
```

## Exemplo das propriedades SelLength, SelStart, SelText

Este exemplo permite ao usuário especificar algum texto a ser pesquisado e, em seguida, procura pelo texto e o seleciona se for encontrado. Para experimentar este exemplo, cole o código na seção Declarations de um formulário que contenha um amplo controle **TextBox** e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Load ()
    Text1.Text = "Two of the pea human experiences "
    Text1.Text = Text1.Text & " are good food and classical music."
End Sub

Private Sub Form_Click ()
    Dim Search, Where ' Declarar variáveis.
    ' Obter a sequência de caracteres de pesquisa do usuário.
    Search = InputBox("Enter test to be found:")
    Where = InStr(Text1.Text, Search) ' Localizar a sequência de caracteres no texto.
    If Where Then ' Se encontrar,
        Text1.SelStart = Where - 1 ' configurar o início da seleção e
        Text1.SelLength = Len(Search) ' configurar o comprimento da seleção.
    Else
        MsgBox "String not found." ' Notificar o usuário.
    End If
End Sub
```

Este exemplo mostra como o objeto **Clipboard** é usado em operações de recorte, cópia, colagem e exclusão. Para experimentar este exemplo, crie um formulário com um controle **TextBox** e use o Menu Editor para criar um menu **Edit** (para cada um dos comandos, configure a propriedade **Caption** = Cut, Copy, Paste e Delete, respectivamente; configure a propriedade **Name** = EditCut, EditCopy, EditPaste e EditDelete respectivamente.).

```
Private Sub EditCut_Click ()
    ' Limpar o conteúdo da Área de transferência.
    Clipboard.Clear
    ' Copiar o texto selecionado para a Área de transferência.
    Clipboard.SetText Screen.ActiveControl.SelText
    ' Excluir o texto selecionado.
    Screen.ActiveControl.SelText = ""
End Sub

Private Sub EditCopy_Click ()
    ' Limpar o conteúdo da Área de transferência.
    Clipboard.Clear
    ' Copiar o texto selecionado para a Área de transferência.
    Clipboard.SetText Screen.ActiveControl.SelText
End Sub

Private Sub EditPaste_Click ()
    ' Colocar texto da Área de transferência no controle ativo.
    Screen.ActiveControl.SelText = Clipboard.GetText ()
End Sub
```

```
Private Sub EditDelete_Clic ()
' Excluir o texto selecionado.
Screen.ActiveControl.SelText = ""
End Sub
```

## Exemplo da propriedade Shape

Este exemplo ilustra as seis possibilidades de formas do controle **Shape**. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha um controle **OptionButton** e um controle **Shape**. Para o **OptionButton**, configure a propriedade **Index** como 0 para criar uma matriz de controle de um elemento e, em seguida, pressione F5. Clique em cada **OptionButton** para ver uma forma diferente.

```
Private Sub Form_Load ()
Dim I ' Declarar variável.
Option1(0).Caption = "Shape #0"
For I = 1 To 5 ' Criar cinco ocorrências de Option1.
Load Option1(I)
' Configurar a localização do novo botão de opção.
Option1(I).Top = Option1(I - 1).Top + Option1(0).Height + 40
' Configurar a legenda do botão de opção.
Option1(I).Caption = "Shape #" & I
' Exibir o novo botão de opção.
Option1(I).Visible = True
Next I
End Sub

Private Sub Option1_Clic (Index As Integer)
Shape1.Shape = Index
End Sub
```

## Exemplo da propriedade Stretch

Este exemplo carrega um ícone de seta de um diretório de ícones para um controle **Image**. A seta se arrasta pelo formulário quando a propriedade **Stretch** for configurada como **True**, e salta pelo formulário quando **Stretch** for configurada como **False**. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha um controle **Image**, um controle **CheckBox** e um controle **Timer** e, em seguida, pressione F5 e clique no formulário. Certifique-se de verificar o caminho até seu diretório de ícones e alterá-lo se necessário. Para visualizar os efeitos da propriedade **Stretch**, clique no **CheckBox** e, em seguida, clique novamente no formulário.

```
Dim ImgW ' Declarar variável.
Private Sub Form_Load ()
' Carregar um ícone no controle Image.
Image1.Picture = LoadPicture("ICONS\ARROWS\ARW02RT.ICO")
Image1.Left = 0 ' Mover a imagem para o lado esquerdo.
ImgW = Image1.Width ' Salvar a largura da imagem.
Timer1.Interval = 300
Timer1.Enabled = False ' Desativar o cronômetro.
Checl.Caption = "Stretch Property"
End Sub

Private Sub Form_Clic ()
Timer1.Enabled = True ' Ativar o cronômetro.
End Sub

Private Sub Timer1_Timer ()
Static MoveIcon As Integer ' Sinalizador para mover o ícone.
If Not MoveIcon Then
Image1.Move Image1.Left + ImgW, Image1.Top, ImgW * 2
Else
' Mover a imagem e a retorna à largura original.
Image1.Move Image1.Left + ImgW, Image1.Top, ImgW
End If
```

```

' Se a imagem está fora da borda do formulário, reiniciar.
If Image1.Left > ScaleWidth Then
    Image1.Left = 0
    Timer1.Enabled = False
End If
MoveIcon = Not MoveIcon ' Reconfigurar o sinalizador.
End Sub

Private Sub Checl_Clic ()
    Image1.Stretch = Checl.Value
End Sub

```

## Exemplo da propriedade TopIndex

Este exemplo preenche um controle **ListBox** com nomes e fontes de tela e, em seguida, rola a **ListBox** quando você clica no formulário. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha um controle **ListBox** e, em seguida, pressione F5 e clique no formulário.

```

Private Sub Form_Load ()
    Dim I ' Declarar variável.
    For I = 0 To Screen.FontCount -1          ' Preencher caixa de listagem
        List1.AddItem Screen.Fonts(I)         ' nomes de fonte de tela.
    Next I
End Sub

Private Sub Form_Clic ()
    Dim X ' Declarar variável.
    X = List1.TopIndex ' Obter o índice atual.
    List1.TopIndex = List1.TopIndex + 5      ' Reconfigurar o primeiro item
da lista.
    If List1.TopIndex = X Then List1.TopIndex = 0
End Sub

```

## Exemplo da propriedade WindowList

Este exemplo cria alguns comandos de menu, ilustra o funcionamento do menu **WindowList** e mostra como permitir que seus usuários adicionem novos formulários a um aplicativo de interface de documentos múltiplos (MDI). Para experimentar este exemplo, crie um objeto **MDIForm** com o comando **Add MDI Form** no menu **Project**. No Form1, configure a propriedade **MDIChild** como **True** e crie um menu chamado **File**. Selecione a caixa **WindowList** no menu **File**. Em seu menu **File**, crie um comando **New**, configure sua propriedade **Name** como **FileMenu** e configure sua propriedade **Index** como 0 para criar uma matriz de controle. Cole o código na seção Declarations do formulário e, em seguida, pressione F5 para executar o programa. A escolha do comando **New** no menu **File** cria novos formulários MDI filhos. Seus nomes estão listados na parte inferior do menu **File**.

```

Private Sub Form_Load ()
    FileMenu(0).Caption = "&New"          ' Configurar a tecla de acesso na le-
genda.
    Load FileMenu(1) ' Criar um novo item de menu.
    FileMenu(1).Caption = "-" ' Configurar separador.
    Load FileMenu(2) ' Criar um novo item de menu.
    FileMenu(2).Caption = "E&xit"        ' Configurar legenda e tecla de aces-
so.
End Sub

Private Sub FileMenu_Clic (Index As Integer)
    Select Case Index
        Case 0 ' Selecionar o comando Novo.
            Dim NewForm As New Form1 ' Criar uma duplicata de Form1.
            ' Carregar NewForm e configurar uma legenda exclusiva.
            NewForm.Caption = "Untitled" & Forms.Count
        Case 2 ' Selecionar o comando Sair.

```

---

```
End ' Encerrar o programa.  
End Select  
End Sub
```

## Exemplo da propriedade WindowState

Este exemplo oculta uma caixa de diálogo (Form2) quando o formulário pai (Form1) está minimizado e reexibe a caixa de diálogo quando o formulário pai retorna a seu estado original ou maximizado. Para experimentar este exemplo cole o código na seção Declarations de Form1 de um aplicativo que contenha dois formulários. Pressione F5 para iniciar o exemplo. Mova Form1, de modo que você possa visualizar ambos os formulários e, em seguida, minimize ou maximize o formulário e observe o comportamento de Form2.

```
Private Sub Form_Load ()  
    Form2.Show ' Mostrar Form2.  
End Sub  
  
Private Sub Form_Resize ()  
    ' Se o formulário pai estiver minimizado...  
    If Form1.WindowState = vbMinimized Then  
        ' ...ocultar Form2.  
        Form2.Visible = False  
    ' Se o formulário pai não estiver minimizado...  
    Else  
        ' ...restaura Form2.  
        Form2.Visible = True  
    End If  
End Sub
```

## Exemplo da propriedade WordWrap

Este exemplo coloca texto em dois controle **Label** e usa a propriedade **WordWrap** para ilustrar seus diferentes comportamentos. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha dois controles **Label** e, em seguida, pressione F5 e clique no formulário para alternar a configuração da propriedade **WordWrap**.

```
Private Sub Form_Load ()  
    Dim Author1, Author2, Quote1, Quote2 ' Declarar variáveis.  
    Label1.AutoSize = True ' Configurar AutoSize.  
    Label2.AutoSize = True  
    Label1.WordWrap = True ' Configurar WordWrap.  
    Quote1 = "I couldn't wait for success, so I went on without it."  
    Author1 = " - Jonathan Winters"  
    Quote2 = "Logic is a system whereby one may go wrong with confidence."  
    Author2 = " - Charles ettering"  
    Label1.Caption = Quote1 & Chr(10) & Author1  
    Label2.Caption = Quote2 & Chr(10) & Author2  
End Sub  
  
Private Sub Form_Clic ()  
    Label1.Width = 1440 ' Configurar a largura como 1 polegada em twips.  
    Label2.Width = 1440  
    Label1.WordWrap = Not Label1.WordWrap ' Ligar e desligar a propriedade WordWrap.  
    Label2.WordWrap = Not Label2.WordWrap  
End Sub
```

---

## Exemplo das propriedades X1, Y1, X2, Y2

Este exemplo exibe uma linha animada que caminha para baixo do formulário quando você clica nele. Para experimentar este exemplo cole o código na seção Declarations de um formulário que contenha um controle **Timer** e um controle **Line** e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Load ()
    Timer1.Interval = 100 ' Configurar o intervalo de Timer.
    ' Posicionar a linha próxima ao canto superior esquerdo.
    ' Configurar as propriedades de Line1.
    With Line1
        .X1 = 100
        .Y1 = 100
        .X2 = 500
        .Y2 = 300
    End With
    Timer1.Enabled = False
End Sub

Private Sub Form_Click ()
    Timer1.Enabled = True ' Iniciar o cronômetro.
End Sub

Private Sub Timer1_Timer ()
    Static Odd ' Declarar variável.
    If Odd Then
        Line1.X2 = Line1.X2 + 250
        Line1.Y2 = Line1.Y2 + 600
    Else
        Line1.X1 = Line1.X1 + 250
        Line1.Y1 = Line1.Y1 + 600
    End If
    Odd = Not Odd ' Alternar o valor.
    ' Se a linha estiver fora do formulário, reiniciar.
    If Line1.Y1 > ScaleHeight Then
        Timer1.Enabled = False ' Esperar por outro clique.
        With Line1
            .X1 = 100
            .Y1 = 100
            .X2 = 500
            .Y2 = 300
        End With
        Odd = False
    End If
End Sub
```

## Propriedade ActiveControl

Retorna o controle que tem o foco. Quando um formulário é referido, como em `ChildForm.ActiveControl`, a **ActiveControl** especifica o controle que teria o foco, se o formulário referido estivesse ativo. Não disponível durante o tempo de criação; somente leitura durante o tempo de execução.

### Sintaxe

*object*.**ActiveControl**

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Você pode usar **ActiveControl** para acessar as propriedades de um controle ou acionar seus métodos: Por exemplo, `Screen.ActiveControl.Tag = "0"`. Um erro durante o tempo de execução ocorre se todos os controles do formulário estiverem invisíveis ou desativados.

Cada formulário pode ter um controle ativo (`Form.ActiveControl`), independente do formulário



---

estar ou não ativo. Você pode escrever código que manipula o controle ativo em cada formulário de seu aplicativo, mesmo quando o formulário não é o formulário ativo. Esta propriedade é especialmente útil em um aplicativo de interface de documento múltiplo (MDI) onde um botão em uma barra de ferramentas deve iniciar uma ação em um controle de um formulário MDI filho. Quando um usuário clica no botão **Copy** na barra de ferramentas, seu código pode referir-se a texto no controle ativo no formulário MDI filho, como em `ActiveForm.ActiveControl.SelText`.

**Observação:** Se você planeja passar `Screen.ActiveControl` a um procedimento, deve declarar o argumento naquele procedimento com a cláusula `As Control` ao invés de especificar um tipo de controle (`As TextBox` ou `As ListBox`) mesmo que `ActiveControl` sempre se refira ao mesmo tipo de controle.

## Propriedade Appearance

Retorna ou configura o estilo de pintura de controles em um **MDIForm** ou objeto **Form** durante o tempo de criação. Somente leitura durante o tempo de execução.

### Sintaxe

*object*.**Appearance**

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Configurações

As configurações da propriedade **Appearance** são:

Configuração	Descrição
0	Plano. Pinta controles e formulários sem efeitos visuais.
1	(Padrão) 3D. Pinta controles com efeitos tridimensionais.

### Comentários

Se configurado como 1 durante o tempo de criação, a propriedade **Appearance** desenha controles com efeitos tridimensionais. Se a propriedade **BorderStyle** do formulário estiver configurada como Fixed Double (**vbFixedDouble**, ou 3), a legenda e borda do formulário também são pintadas com efeitos tridimensionais. A configuração de **Appearance** como 1 também faz com que o formulário e seus controles tenham sua propriedade **BackColor** configurada como a cor selecionada para objetos 3D na guia **Appearance** da caixa de diálogo **Display Properties** do sistema operacional.

A configuração da propriedade **Appearance** como 1 para um objeto **MDIForm** afeta apenas o formulário MDI pai. Para ter efeitos tridimensionais sobre formulários MDI filho, você deve configurar a propriedade **Appearance** de cada formulário filho como 1.

---

## Propriedade Bold

Retorna ou configura o estilo de fonte do objeto **Font** como negrito ou não-negrito.

### Sintaxe

*object.Bold* [= *boolean*]

A sintaxe da propriedade **Bold** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> especificando o estilo de fonte, conforme descrito em Configurações.

### Configurações

As configurações para *boolean* são:

Configuração	Descrição
<b>True</b>	Ativa a formatação negrito.
<b>False</b>	(Padrão) Desativa a formatação negrito.

### Comentários

O objeto **Font** não está diretamente disponível durante o tempo de criação. Ao invés, você configura a propriedade **Bold** selecionando a propriedade **Font** de um controle na janela **Properties** e clicando no botão **Properties**. Na caixa **Font Style** da caixa de diálogo **Font**, selecione **Bold** ou **Bold Italic**. Durante o tempo de execução, entretanto, você configura **Bold** diretamente especificando suas configurações no objeto **Font**.

## Propriedade FontTransparent

Retorna ou configura um valor que determina se o texto e elementos gráficos de segundo plano em um objeto **Form** ou **Printer** ou um controle **PictureBox** são exibidos nos espaços ao redor de caracteres.

### Sintaxe

*object.FontTransparent* [= *boolean*]

A sintaxe da propriedade **FontTransparent** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	A <u>expressão Booleana</u> especificando o estado de texto e elementos gráficos de segundo plano, conforme descrito em Configurações.

### Configurações

As configurações para *boolean* são:

Configuração	Descrição
<b>True</b>	(Padrão) Permite que elementos gráficos e texto de segundo plano sejam exibidos ao redor dos espaços dos caracteres em uma fonte.
<b>False</b>	Mascara os elementos gráficos e texto existente ao redor dos caracteres de uma fonte.

### Comentários

Configura **FontTransparent** durante o tempo de criação usando a janela **Properties** ou durante o tempo de execução usando código. A alteração de **FontTransparent** durante o tempo de execução não afeta os elementos gráficos e o texto já desenhado em **Form**, **Printer** ou **PictureBox**.

## Propriedade Italic

Retorna ou configura o estilo de fonte do objeto **Font** para itálico ou não-italico.

### Sintaxe

*object.Italic* [= *boolean*]

A sintaxe da propriedade **Italic** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> especificando o estilo de fonte conforme descrito em Configurações.

### Configurações

As configurações para *boolean* são:

Configuração	Descrição
<b>True</b>	Ativa a formatação itálica.
<b>False</b>	(Padrão) Desativa a formatação itálica.

### Comentários

O objeto **Font** não está diretamente disponível durante o tempo de criação. Ao invés, você configura a propriedade **Italic** selecionando a propriedade **Font** de um controle na janela **Properties** e clicando no botão **Properties**. Na caixa **Font Style** da caixa de diálogo **Font**, selecione **Italic** ou **Bold Italic**. Durante o tempo de execução, entretanto, você configura **Italic** diretamente, especificando sua configuração para o objeto **Font**.

## Propriedade LBound

Retorna o valor ordinal mais baixo de um controle em uma matriz de controle.

### Sintaxe

*object.LBound*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

A propriedade **LBound** é igual ao valor da propriedade **Index** do primeiro controle na matriz. Tipicamente este valor é 0, pois o Visual Basic atribui automaticamente um valor **Index** de 0 ao primeiro controle em uma matriz de controle. Se você alterar manualmente o valor **Index** para o primeiro controle em uma matriz para algum outro valor (por exemplo, 1), **LBound** retorna o valor que você atribuiu manualmente a **Index** (neste exemplo, 1).

---

## Propriedade Size (Font)

Retorna ou configura o tamanho da fonte usada no objeto **Font**.

### Sintaxe

*object*.Size [= *number*]

A sintaxe da propriedade **Size** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Uma <u>expressão numérica</u> especificando o tamanho da fonte em pontos.

### Comentários

Use esta propriedade para formatar o texto no tamanho da fonte desejada. O tamanho da fonte padrão é determinado pelo sistema operacional. Para alterar o padrão, especifique o tamanho da fonte em pontos. O valor máximo para a propriedade **Size** é 2048 pontos.

O objeto **Font** não está diretamente disponível durante o tempo de criação. Ao invés, você configura a propriedade **Size** selecionando a propriedade **Font** na janela **Properties** e clicando no botão **Properties**. Na caixa **Size** da caixa de diálogo **Font**, selecione o tamanho desejado. Durante o tempo de execução, entretanto, configure **Size** diretamente, especificando sua configuração para o objeto **Font**.

## Propriedade StartMode

Retorna ou configura um valor que determina se um aplicativo se inicia como um projeto autônomo ou como um componente ActiveX. Somente leitura durante o tempo de execução.

### Sintaxe

*object*.StartMode

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Configurações

As configurações da propriedade **StartMode** são:

Constante	Valor	Descrição
<b>vbSModeStandalone</b>	0	(Padrão) O aplicativo é iniciado como um projeto autônomo.
<b>VbSModeAutomation</b>	1	O aplicativo é iniciado como um componente ActiveX.

### Comentários

Estas constantes estão listadas na biblioteca de objetos Visual Basic (VB) no Object Browser.

Durante o tempo de criação, você pode configurar **StartMode** na caixa de diálogo **Project Options** como 1 (**vbSModeAutomation**) para depurar um aplicativo como se ele tivesse sido iniciado como um componente ActiveX.

Uma vez que o projeto seja compilado, o valor da propriedade StartMode é determinado pela forma como o aplicativo é iniciado, não por sua configuração nominal na caixa de diálogo **Project Options**.

Quando **StartMode** está configurada como 1, e não existem classes públicas no projeto, você deve usar a instrução **End** e selecionar **End** no menu ou barra de ferramentas **Run**, para encerrar o aplicativo. Se você escolhe **Close** no menu **System**, o formulário se fecha, mas o projeto ainda estará sendo executado.

---

## Propriedade StrieThrough

Retorna ou configura o estilo de fonte do objeto **Font** como tachado ou não-tachado.

### Sintaxe

*object.StrieThrough* [= *boolean*]

A sintaxe da propriedade **StrieThrough** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> especificando o estilo de fonte, conforme descrito em Configurações.

### Configurações

As configurações para *boolean* são:

Configuração	Descrição
<b>True</b>	Ativa a formatação tachada.
<b>False</b>	(Padrão) Desativa formatação tachado.

### Comentários

O objeto **Font** não está diretamente disponível durante o tempo de criação. Ao invés, você configura a propriedade **StrieThrough** selecionando a propriedade **Font** de um controle na janela **Properties** e clicando no botão **Properties**. Na caixa de diálogo **Font**, selecione a caixa de seleção **Strieout**. Durante o tempo de execução, entretanto, você configura **StrieThrough** diretamente especificando sua configuração para o objeto **Font**.

## Propriedade TracDefault

Retorna ou configura um valor que determina se o objeto **Printer** sempre aponta para a mesma impressora ou altera a impressora para a qual ele aponta, se você altera a configuração de impressora padrão **Painel de controle** do sistema operacional. Não está disponível durante o tempo de criação.

### Sintaxe

*object.TracDefault* [= *boolean*]

A sintaxe da propriedade **TracDefault** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> especificando a impressora para a qual aponta <i>object</i> , conforme descrito em Configurações.

### Configurações

As configurações para *boolean* são:

Configuração	Descrição
<b>True</b>	(Padrão) O objeto <b>Printer</b> altera a impressora para a qual ele aponta quando você altera as configurações de impressora padrão no <b>Painel de controle</b> do sistema operacional.
<b>False</b>	O objeto <b>Printer</b> continua a apontar para a mesma impressora, embora você altere as configurações de impressora padrão no <b>Painel de controle</b> do sistema operacional.

### Comentários

A alteração da configuração da propriedade **TracDefault** enquanto um trabalho de impressão está sendo executado envia uma instrução **EndPage** implícita ao objeto **Printer**.

---

## Propriedade Type (Picture)

Retorna o formato gráfico de um objeto **Picture**. Não disponível durante o tempo de criação; somente leitura durante o tempo de execução.

### Sintaxe

*object.Type*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Valores de retorno

Os valores de retorno para a propriedade **Type** são:

Constante	Valor	Descrição
<b>vbPicTypeNone</b>	0	Picture está vazia
<b>vbPicTypeBitmap</b>	1	<u>Bitmap</u> (arquivos .bmp)
<b>vbPicTypeMetafile</b>	2	<u>Metarquivo</u> (arquivos .wmf)
<b>vbPicTypeIcon</b>	3	<u>Ícone</u> (arquivos .ico)
<b>vbPicTypeEMetafile</b>	4	Metarquivo aprimorado (arquivos .emf)

### Comentários

Estas constantes são listadas na biblioteca de objetos Visual Basic (VB) no Object Browser.

## Propriedade UBound

Retorna o valor ordinal mais alto de um controle em uma matriz de controle.

### Sintaxe

*object.UBound*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

**UBound** é igual ao valor da propriedade **Index** do último controle na matriz.

## Propriedade Underline

Retorna ou configura o estilo de fonte do objeto **Font** como sublinhado ou não-sublinhado.

### Sintaxe

*object.Underline* [= *boolean*]

A sintaxe da propriedade **Underline** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> especificando o estilo de fonte, conforme descrito em Configurações.

### Configurações

As configurações para *boolean* são:

Configuração	Descrição
<b>True</b>	Ativa a formatação sublinhada.
<b>False</b>	(Padrão) Desativa a formatação sublinhada.

### Comentários

O objeto **Font** não está diretamente disponível durante o tempo de criação. Ao invés, você pode configurar a propriedade **Underline** selecionando a propriedade **Font** de um controle na janela **Properties** e clicando no botão **Properties**. Na caixa de diálogo **Font**, selecione a caixa de seleção **Underline**. Durante o tempo de execução, entretanto, você configura **Underline** diretamente especificando sua configuração para o objeto **Font**.

---

## Propriedade UseMnemonic

Retorna ou configura um valor que especifica se um E comercial (&) incluído no texto da propriedade **Caption** do controle **Label** define uma tecla de acesso.

### Sintaxe

*object.UseMnemonic* [= *boolean*]

A propriedade **UseMnemonic** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> especificando se o controle <b>Label</b> ativa uma tecla de acesso, conforme descrito em Configurações.

### Configurações

As configurações para *boolean* são:

Configuração	Descrição
<b>True</b>	(Padrão) Qualquer E comercial aparecendo no texto da propriedade <b>Caption</b> faz com que o caractere posterior ao E comercial torne-se uma tecla de acesso. O próprio E comercial não é exibido na interface do controle <b>Label</b> .
<b>False</b>	Qualquer E comercial aparecendo no texto da propriedade <b>Caption</b> é exibido como um E comercial na interface do controle <b>Label</b> .

### Comentários

Durante o tempo de execução, pressionar ALT+ a tecla de acesso definida na propriedade **Caption** do controle **Label** move o foco para o controle seguinte ao controle **Label** na ordem de tabulação.

---

## Propriedade Weight

Retorna ou configura o peso dos caracteres que compõem um objeto **Font**. O peso refere-se à espessura dos caracteres, ou "taxa de negrito". Quanto mais alto o valor, mais negrito é o caractere.

### Sintaxe

*object.Weight* [= *number*]

A sintaxe da propriedade **Weight** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Uma <u>expressão numérica</u> especificando o peso da fonte.

### Comentários

O objeto **Font** não está diretamente disponível durante o tempo de criação. Você configura a propriedade **Weight** do objeto **Font** selecionando a propriedade **Font** do controle na janela **Properties** e clicando no botão **Properties**. Você configura implicitamente a propriedade **Weight** selecionando um item na caixa **Font Style** na caixa de diálogo **Font**. As configurações Regular e Italic têm um valor **Weight** de 400 (o padrão), e as configurações Bold e Bold Italic têm um valor **Weight** de 700. Durante o tempo de execução, entretanto, você configura **Weight** diretamente especificando sua configuração para o objeto **Font**.

Se você configura um **Weight** de um objeto **Font** como um valor diferente de 400 ou 700 durante o tempo de execução, o Visual Basic converte seu valor para 400 ou 700, dependendo de qual valor está mais próximo do valor configurado por você. Os intervalos precisos são: **Weight** > 400 e < 551 converte para 400; **Weight** > 550 converte para 700.

## Exemplo da propriedade ActiveControl

Este exemplo exibe o texto do controle ativo. Para experimentar este exemplo, cole o código na seção Declarations de um formulário que contenha controles **TextBox**, **Label** e **CommandButton** e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Click ()
    If TypeOf Screen.ActiveControl Is TextBox Then
        Label1.Caption = Screen.ActiveControl.Text
    Else
        Label1.Caption = "Button: " + Screen.ActiveControl.Caption
    End If
End Sub
```

Este exemplo mostra como você pode usar o objeto **Clipboard** em operações de recorte, cópia, colagem e exclusão usando botões em uma barra de ferramentas. Para experimentar este exemplo, coloque os controles **TextBox** e **CheckBox** em Form1 e, em seguida, crie um novo formulário MDI. Neste formulário, insira um controle **PictureBox** e, em seguida, insira um **CommandButton** na **PictureBox**. Configure a propriedade **Index** de **CommandButton** como 0 (criando uma matriz de controle). Configure a propriedade **MDIChild** de Form1 como **True**.

Para executar o exemplo, copie o código na seção Declarations de **MDIForm** e, em seguida, pressione F5. Observe que quando o **CheckBox** tem o foco, os botões não funcionam, pois **CheckBox** é agora o controle ativo, ao invés de **TextBox**.

```
Private Sub MDIForm_Load ()
    Dim I ' Declarar variável.
    Command1(0).Move 0, 0, 700, 300 ' Posicionar o botão na barra de ferramentas.
    For I = 1 To 3 ' Criar outros botões.
        Load Command1(I) ' Criar botão.
        Command1(I).Move I * 700, 0, 700, 300 ' Posicionar e dimensionar o botão.
        Command1(I).Visible = True ' Exibir o botão.
    Next I
    Command1(0).Caption = "Cut" ' Configurar legendas de botão.
    Command1(1).Caption = "Copy"
    Command1(2).Caption = "Paste"
    Command1(3).Caption = "Del"
```



---

```
End Sub
```

```
Private Sub Command1_Clic (Index As Integer)
    ' ActiveForm refere-se ao formulário ativo no formulário MDI.
    If TypeOf ActiveForm.ActiveControl Is TextBox Then
        Select Case Index
            Case 0 ' Recortar.
                ' Copiar o texto selecionado para a Área de transferência.
                Clipboard.SetText ActiveForm.ActiveControl.SelText
                ' Excluir o texto selecionado.
                ActiveForm.ActiveControl.SelText = ""
            Case 1 ' Copiar.
                ' Copiar o texto selecionado para a Área de transferência.
                Clipboard.SetText ActiveForm.ActiveControl.SelText
            Case 2 ' Colar.
                ' Colocar o texto da Área de transferência na caixa de texto.
                ActiveForm.ActiveControl.SelText = Clipboard.GetText()
            Case 3 ' Excluir.
                ' Excluir o texto selecionado.
                ActiveForm.ActiveControl.SelText = ""
        End Select
    End If
End Sub
```

## Exemplo das propriedades Bold, Italic, Size, StrieThrough, Underline e Weight

Este exemplo imprime texto em um formulário a cada clique de mouse. Para experimentar este exemplo, cole o código na seção Declarations de um formulário e, em seguida, pressione F5 e clique duas vezes no formulário.

```
Private Sub Form_Clic ()
    Font.Bold = Not Font.Bold ' Ligar e desligar negrito.
    Font.StrieThrough = Not Font.StrieThrough ' Ligar e desligar tachado.
    Font.Italic = Not Font.Italic ' Ligar e desligar itálico.
    Font.Underline = Not Font.Underline ' Ligar e desligar sublinhado.
    Font.Size = 16 ' Configurar a propriedade Size.
    If Font.Bold Then
        Print "Font weight is " & Font.Weight & " (negrito)."
    Else
        Print "Font weight is " & Font.Weight & " (não negrito)."
    End If
End Sub
```

## Exemplo da propriedade FontTransparent

Este exemplo imprime texto sobre um gráfico em um controle **PictureBox**. Coloque um **PictureBox** em um formulário, configure sua propriedade **AutoSize** como **True** e carregue sua propriedade **Picture** com um arquivo bitmap (.bmp). Para experimentar este exemplo, cole o código na seção Declarations de um formulário e, em seguida, pressione F5 e clique duas vezes no formulário.

```
Private Sub Form_Clic ()
    ' Ligar e desligar propriedade.
    Picture1.FontTransparent = Not Picture1.FontTransparent
    Picture1.Print "Demo of FontTransparent property."
End Sub
```

## Exemplo das propriedades LBound, UBound

Este exemplo imprime os valores destas duas propriedades em uma matriz de controle. Coloque um controle **OptionButton** em um formulário e configure sua propriedade **Index** como 0 (para criar uma matriz de controle). Para experimentar este exemplo, cole o código na seção Declarations de um formulário e, em seguida, pressione F5 e clique duas vezes no formulário.

```
Private Sub Form_Paint ()
```

```

Static FlagFormPainted As Integer
If FlagFormPainted <> True Then ' Quando um formulário é pintado pela
primeira vez,
    For i = 1 To 3
        Load Option1(i) ' adicionar três botões de opção à matriz.
        Option1(i).Top = Option1(i - 1).Top + 350
        Option1(i).Visible = True
    Next I
    For I = 0 to 3 ' Colocar legendas nos botões de opção.
        Option1(i).Caption = "Option #º." & CStr(i)
    Next I
    Option1(0).Value = True ' Selecionar o primeiro botão de opção.
    FlagFormPainted = True ' Terminar a pintura do formulário.
End If
End Sub
Private Sub Form_Click ()
    Print "Control array's Count property is " & Option1().Count
    Print "Control array's Lbound property is " & Option1().LBound
    Print "Control array's Ubound property is " & Option1().UBound
End Sub

```

## Exemplo da propriedade StartMode

Este exemplo mostra um possível efeito da configuração da propriedade **StartMode** como 1 (**vbSMModeAutomation**) durante o tempo de criação. Crie um projeto ActiveX EXE. Crie um novo formulário. No menu **Project**, escolha o comando **Project Properties**. Selecione a guia **Componente** e, em seguida, selecione o botão de opção do componente ActiveX no grupo **Start Mode**. Escolha **O** para fechar a caixa de diálogo **Project Properties**. Para experimentar este exemplo, cole o código na seção Declarations do formulário e, em seguida, pressione F5 e clique duas vezes no menu **Control**, à esquerda da barra de título do formulário. Se o formulário não for exibido, insira `Form1.Show` na janela **Immediate**.

```

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    If UnloadMode = vbFormControlMenu And App.StartMode = vbSMModeAutomation Then
        Msg = "Form will close but application will still be running." &
Chr(10)
        Msg = Msg + "To terminate application without a public class" &
Chr(10)
        Msg = Msg + "you must use an End statement"
        MsgBox Msg
    End If
End Sub

```

---

## Exemplo da propriedade UseMnemonic

Este exemplo lê a configuração da propriedade **UseMnemonic** de um controle **Label**. Para experimentar este exemplo, cole o código na seção Declarations de um formulário que contenha um **Label** e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Click()  
    If Label1.UseMnemonic And InStr(Label1, "&") Then  
        MsgBox "The label has an access ey character."  
    ElseIf Label1.UseMnemonic And Not InStr(Label1, "&") Then  
        MsgBox "The label supports an access ey character but doesn't have  
an ampersand."  
    Else  
        MsgBox "The label doesn't support an access ey character."  
    End If  
End Sub
```

## Exemplo das propriedades Type e Width

Este exemplo lê a configuração das propriedades **Type** e **Width** de um objeto **Picture** em um controle **PictureBox**. Para experimentar este exemplo, cole o código na seção Declarations de um formulário que contenha um **PictureBox** cuja propriedade **Picture** é configurada como um ícone e, em seguida, pressione F5 e clique no formulário.

```
Private Sub Form_Click()  
    If Picture1.Picture.Type = vbPicTypeIcon Then  
        Print "The graphic in the picture box is an icon."  
    Else  
        Print "The Picture property isn't set to an icon."  
    End If  
    Print "Width of the graphic in HiMetrics is " & Picture1.Picture.Width  
    Print "Width of picture box itself in twips is" & Picture1.Width  
End Sub
```

## Propriedade ActiveForm

Retorna o formulário que é a janela ativa. Se um objeto **MDIForm** está ativo ou é referido, ele especifica o formulário MDI filho.

### Sintaxe

*object.ActiveForm*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Use a propriedade **ActiveForm** para acessar as propriedades de um formulário ou para acionar seus métodos — por exemplo, `Screen.ActiveForm.MousePointer = 4`.

Esta propriedade é especialmente útil em um aplicativo de interface de documento múltiplo (MDI) onde um botão em uma barra de ferramentas deve iniciar uma ação ou um controle em um formulário MDI filho. Quando um usuário clica no botão **Copy** na barra de ferramentas, seu código pode referir-se ao texto no controle ativo no formulário MDI filho — por exemplo, `ActiveForm.ActiveControl.SelText`.

Quando um controle em um formulário tem o foco, este formulário é o formulário ativo na tela (`Screen.ActiveForm`). Além disso, um objeto **MDIForm** pode obter um formulário filho que é o formulário ativo dentro do conceito de formulário MDI pai (`MDIForm.ActiveForm`). A **ActiveForm** na tela não é necessariamente o mesmo que a **ActiveForm** no formulário MDI como, por exemplo, quando uma caixa de diálogo está ativa. Por este motivo, especifique o **MDIForm** com **ActiveForm** quando existe uma possibilidade de uma caixa de diálogo ser a configuração da propriedade **ActiveForm**.

**Observação:** Quando um formulário MDI filho ativo não está maximizado, as barras de título de ambos os formulários, pai e filho, aparecem ativas.

Se você planeja passar `Screen.ActiveForm` ou `MDIForm.ActiveForm` a um procedimento, deve declarar o argumento naquele procedimento com o tipo genérico (`As Form`) ao invés de um

---

tipo de formulário específico (As `MyForm`) mesmo que **ActiveForm** sempre se refira ao mesmo tipo de formulário.

A propriedade **ActiveForm** determina o valor padrão do objeto **ProjectTemplate**.

## Propriedade Caption

- **Formulário** — determina o texto exibido na barra de título do objeto **Form** ou **MDIForm**. Quando o formulário está minimizado, este texto é exibido abaixo do ícone do formulário.
- **Controle** — determina o texto exibido em, ou junto a um controle.
- **Objeto MenuLine** — determina o texto exibido para um controle **Menu** ou um objeto na coleção **MenuItems**.

Para um controle **Menu**, **Caption** é normalmente para leitura/gravação durante o tempo de execução. Mas, **Caption** é somente leitura em menus exibidos ou fornecidos pelo Visual Basic a suplementos como, por exemplo, o objeto **MenuLine**.

### Sintaxe

*object*.Caption [= *string*]

A sintaxe da propriedade **Caption** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To. Se <i>object</i> for omitido, presume-se que o formulário associado ao <u>módulo</u> do formulário ativo seja <i>object</i> .
<i>string</i>	Uma <u>expressão de sequência de caracteres</u> que é avaliada como texto exibido como legenda.

### Comentários

Quando você cria um novo objeto, sua legenda padrão é a configuração da propriedade **Name**. Esta legenda padrão inclui o nome do objeto e um número inteiro, por exemplo, Command1 ou Form1. Para um rótulo mais descritivo, configure a propriedade **Caption**.

Você pode usar a propriedade **Caption** para atribuir uma tecla de acesso a um controle. Na legenda, inclua um E comercial (&) imediatamente antes do caractere que você deseja designar como uma tecla de acesso. O caractere é sublinhado. Pressione a tecla ALT mais o caractere sublinhado para mover o foco até aquele controle. Para incluir um E comercial em uma legenda sem criar uma tecla de acesso, inclua dois Es comerciais (&&). Um único E comercial é exibido na legenda e nenhum caractere será sublinhado.

O tamanho da legenda de um controle **Label** é ilimitado. Para formulários e todos os outros controles que têm legendas, o limite é 255 caracteres.

Para exibir a legenda de um formulário, configure a propriedade **BorderStyle** como Fixed Single (1 ou **vbFixedSingle**), Sizable (2 ou **vbSizable**), ou Fixed Dialog (3 ou **vbFixedDialog**). Uma legenda longa demais para a barra de título de um formulário é truncada. Quando um formulário MDI filho é maximizado dentro de um objeto **MDIForm**, a legenda do formulário filho é incluída na legenda do formulário pai.

**Dica** Para um rótulo, configure a propriedade **AutoSize** como **True** para redimensionar automaticamente o controle, para ajustá-lo à sua legenda.

---

## Propriedade Checed

Retorna ou configura um valor que determina se uma marca de seleção é exibida junto a um item de menu.

### Sintaxe

*object*.**Checed** [= *boolean*]

A sintaxe da propriedade **Checed** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> que especifica se uma marca de seleção é exibida junto a um item de menu.

### Configurações

As configurações para *boolean* são:

Configuração	Descrição
<b>True</b>	Coloca uma marca de seleção junto a um item de menu.
<b>False</b>	(Padrão) Não coloca uma marca de seleção junto a um item de menu.

### Comentários

Durante o tempo de criação, você pode usar o Menu Editor para configurar **Checed** como **True**. Durante o tempo de execução, você pode ligar e desligar **Checed** como parte de um procedimento de evento Clic anexado a um controle **Menu**. Você também pode configurar o valor de **Checed** em um procedimento de inicialização ou no procedimento de evento Load de um formulário.

Para um controle **Menu**, **Checed** é normalmente para leitura/gravação durante o tempo de execução. Mas **Checed** é somente leitura para itens de menu que são expostos ou fornecidos pelo Visual Basic a suplementos como, por exemplo o comando **Add-In Manager** no menu **Add-Ins**.

## Propriedade Count (Coleções VB)

Retorna o número de objetos em uma coleção.

### Sintaxe

*object*.**Count**

O espaço reservado *object* é uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Você pode usar esta propriedade com uma instrução **For...Next** para executar uma operação nos formulários ou controles em uma coleção. Por exemplo, o código abaixo move todos os controles em um formulário 0,5 polegada para a direita (a configuração de propriedade **ScaleMode** é 1 ou **vbTwips**):

```
For I = 0 To Form1.Controls.Count - 1
    Form1.Controls(I).Left = Form1.Controls(I).Left + 720
Next I
```

Você também pode usar este tipo de estrutura para ativar ou desativar rapidamente todos os controles em um formulário.

Quando usado com a instrução **If TypeOf**, você pode ciclar por todos os controles e alterar, por exemplo, a configuração de propriedade **Enabled** de somente as caixas de texto ou a configuração de propriedade **BacColor** de somente os botões de opção.

## Propriedade Enabled

Retorna ou configura um valor que determina se um formulário ou controle pode responder a eventos gerados pelo usuário.

### Sintaxe

---

*object.Enabled* [= *boolean*]

A sintaxe da propriedade **Enabled** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To. Se <i>object</i> for omitido, presume-se que o formulário associado ao <u>módulo</u> do formulário ativo seja <i>object</i> .
<i>boolean</i>	Uma <u>expressão Booleana</u> que especifica se <i>object</i> pode responder a eventos gerados pelo usuário.

### Configurações

As configurações para *boolean* são:

Configuração	Descrição
<b>True</b>	(Padrão) Permite que <i>object</i> responda a eventos.
<b>False</b>	Impede que <i>object</i> responda a eventos.

### Comentários

A propriedade **Enabled** permite que formulários e controles sejam ativados ou desativados durante o tempo de execução. Por exemplo, você pode desativar objetos que não se aplicam ao estado atual do aplicativo. Você também pode desativar um controle usado puramente para finalidades de exibição como, por exemplo, uma caixa de texto que oferece informações somente leitura.

Desativar um controle **Timer** configurando **Enabled** como **False** cancela a contagem regressiva configurada pela propriedade **Interval** do controle.

Para um controle **Menu**, **Enabled** é normalmente para leitura/gravação durante o tempo de execução. Mas **Enabled** é somente leitura para itens de menu que são expostos ou fornecidos pelo Visual Basic a suplementos como, por exemplo, o comando **Add-In Manager** no menu **Add-Ins**.

## Propriedade HelpContextID

Retorna ou configura um número de contexto associado a um objeto. Usado para fornecer Ajuda vinculada ao contexto a seu aplicativo.

### Sintaxe

*object.HelpContextID* [= *number*]

A sintaxe da propriedade **HelpContextID** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To. Se <i>object</i> for omitido, presume-se que o formulário associado ao <u>módulo</u> do formulário ativo seja <i>object</i> .
<i>Number</i>	Uma <u>expressão numérica</u> que especifica o número de contexto do tópico da <b>Ajuda</b> associado a <i>object</i> .

### Configurações

As configurações para *number* são:

Configuração	Descrição
0	(Padrão) Nenhum número de contexto especificado.
> 0	Um número inteiro especificando um número de contexto válido.

### Comentários

Para Ajuda vinculada ao contexto sobre um objeto em seu aplicativo, você precisa atribuir o mesmo número de contexto tanto ao *object* quanto ao tópico da **Ajuda** associado, ao compilar seu arquivo da **Ajuda**.

Se você criou um arquivo da **Ajuda** para o ambiente operacional do Microsoft Windows para seu aplicativo e configurou a propriedade **HelpFile** do aplicativo, quando um usuário pressiona a tecla F1, o Visual Basic chama a **Ajuda** automaticamente e procura pelo tópico identificado pelo número de contexto atual.

O número de contexto atual é o valor de **HelpContextID** para o objeto que tem o foco. Se a

**HelpContextID** for configurada como 0, então o Visual Basic procura na **HelpContextID** do recipiente do objeto e, em seguida, no recipiente daquele objeto, e assim por diante. Se um número de contexto atual diferente de zero não puder ser encontrado, a tecla F1 é ignorada.

Para um controle **Menu**, **HelpContextID** é normalmente para leitura/gravação durante o tempo de execução. Mas, **HelpContextID** é somente leitura para itens de menu que são exibidos ou fornecidos pelo Visual Basic a suplementos como, por exemplo, o comando **Add-In Manager** no menu **Add-Ins**.

**Observação:** Montar um arquivo da **Ajuda** exige o Microsoft Windows Help Compiler, que é incluído no Visual Basic Professional Edition.

## Propriedade Index (Matriz de controle)

Retorna ou configura o número que identifica com exclusividade um controle em uma matriz de controle. Disponível somente se o controle fizer parte de uma matriz de controle.

### Sintaxe

*object*[(*number*)].**Index**

A sintaxe da propriedade **Index** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	A <u>expressão numérica</u> que é avaliada como um número inteiro que identifica um controle individual dentro de uma matriz de controle.

### Configurações

As configurações para *number* são:

Configuração	Descrição
Nenhum valor	(Padrão) Não faz parte de uma matriz de controle.
0 a 32.767	É parte de uma matriz. Especifica um número inteiro maior que, ou igual a 0 que identifica um controle dentro de uma matriz de controle. Todos os controles em uma matriz de controle têm a mesma propriedade <b>Name</b> . O Visual Basic atribui automaticamente o próximo número inteiro dentro da matriz de controle.

### Comentários

Visto que elementos de matriz de controle compartilham a mesma configuração da propriedade **Name**, você deve usar a propriedade **Index** no código para especificar um determinado controle na matriz. **Index** deve aparecer como um número inteiro (ou uma expressão numérica que é avaliada como um número inteiro) entre parênteses junto ao nome da matriz de controle — por exemplo, `MyButtons(3)`. Você usa a configuração da propriedade **Tag** para distinguir um controle de outro dentro de uma matriz de controle.

Quando um controle na matriz reconhece que ocorreu um evento, o Visual Basic chama o procedimento de evento da matriz de controle, e passa a configuração de **Index** aplicável como argumento adicional. Esta propriedade também é usada quando você cria controles de maneira dinâmica durante o tempo de execução com a instrução **Load**, ou os remove com a instrução **Unload**. Embora, o Visual Basic atribua como padrão o próximo valor de número inteiro como o valor de **Index** para um novo controle em uma matriz de controle, você pode ignorar este valor atribuído e ignorar números inteiros. Você também pode configurar **Index** como um número inteiro diferente de 0 para o primeiro controle da matriz. Se você se refere a um valor **Index** no código que não identifica um dos controles em uma matriz de controle, ocorre um erro de tempo de execução do Visual Basic.

**Observação:** Para remover um controle de uma matriz de controle, altere a configuração da propriedade **Name** do controle e exclua a configuração da propriedade **Index** do controle.

## Propriedade Name

- Retorna o nome usado no código para identificar um formulário, controle ou objeto de acesso a

---

dados. Somente leitura durante o tempo de execução.

- Retorna ou configura o nome de um objeto fonte.

### Sintaxe

*object*.Name

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To. Se *object* for omitido, presume-se que o formulário associado ao módulo do formulário ativo seja *object*.

### Comentários

O nome padrão para novos objetos é o tipo de objeto mais um número inteiro exclusivo. Por exemplo, o primeiro objeto **Form** novo é Form1, um novo objeto **MDIForm** é MDIForm1 e o terceiro controle **TextBox** que você cria em um formulário é Text3.

A propriedade **Name** de um objeto deve iniciar-se com uma letra, e ter no máximo 40 caracteres. Ela pode incluir números e caracteres sublinhado (\_), mas não pode incluir pontuação ou espaços. Os formulários não podem ter o mesmo nome que outro objeto público como, por exemplo, **Clipboard**, **Screen** ou **App**. Embora, a configuração da propriedade **Name** possa ser uma palavra-chave, nome de propriedade ou nome de outro objeto, isto pode criar conflitos em seu código. Você pode usar a propriedade **Name** do formulário com a instrução **Dim** durante o tempo de execução para criar outras ocorrências do formulário. Você não pode ter dois formulários com o mesmo nome durante o tempo de criação.

Você pode criar uma matriz de controle do mesmo tipo configurando a propriedade **Name** como o mesmo valor. Por exemplo, quando você configura o nome de todos os botões de opção em um grupo como MyOpt, o Visual Basic atribui valores exclusivos à propriedade **Index** de cada controle para diferenciá-los de outros na matriz. Dois controles de tipos diferentes não podem compartilhar o mesmo nome.

**Observação:** Embora o Visual Basic, com frequência, utilize a configuração da propriedade **Name** como valor padrão para as propriedades **Caption**, **LinTopic** e **Text**, alterar uma destas propriedades não afeta as outras.

## Propriedade Parent

Retorna o formulário, objeto ou coleção que contém um controle, ou outro objeto ou coleção.

### Sintaxe

*object*.Parent

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Use a propriedade **Parent** para acessar as propriedades, métodos ou controles de um pai do objeto. Por exemplo:

```
MyButton.Parent.MousePointer = 4
```

A propriedade **Parent** é útil em um aplicativo em que você passa objetos como argumentos. Por exemplo, você poderia passar uma variável de controle a um procedimento geral em um módulo e usar a propriedade **Parent** para acessar seu formulário pai.

Não existe relacionamento entre a propriedade **Parent** e a propriedade **MDIChild**. Existe, entretanto, um relacionamento pai-filho entre um objeto **MDIForm** e qualquer objeto **Form** que tenha uma propriedade **MDIChild** configurada como **True**.

## Propriedade Path

Retorna ou configura o caminho atual. Não está disponível durante o tempo de criação. Para o objeto **App**, somente leitura durante o tempo de execução.

### Sintaxe

*object*.Path [= *pathname*]

A sintaxe da propriedade **Path** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>pathname</i>	Uma <u>expressão de sequência de caracteres</u> que é avaliada como o nome do caminho.



---

## Comentários

O valor da propriedade **Path** é uma sequência de caracteres indicando um caminho, por exemplo, C:\Ob ou C:\Windows\System. Para um controle **DirListBox** ou **FileListBox**, o padrão é o caminho atual quando o controle é criado durante o tempo de execução. Para o objeto **App**, **Path** especifica o caminho do arquivo de projeto .VBP ao executar o aplicativo a partir do ambiente de desenvolvimento ou o caminho do arquivo .exe quando o aplicativo é executado como um arquivo executável.

Use esta propriedade ao criar os recursos de manipulação e procura de arquivos do aplicativo. Configurar a propriedade **Path** afeta um controle semelhante ao comando **chdir** do MS-DOS — caminhos relativos são permitidos, com ou sem uma especificação de unidade de disco. As especificação de uma unidade de disco apenas com dois pontos (:) seleciona o diretório atual naquela unidade de disco.

A propriedade **Path** também pode ser configurada como um caminho de rede qualificado sem uma conexão de unidade de disco usando a sintaxe abaixo:

\\servername\sharename\path

A sintaxe acima altera a propriedade **Drive** para uma sequência de caracteres de comprimento zero ("").

Alterar o valor de **Path** produz estes efeitos:

- Em um controle **DirListBox**, gera um evento Change.
- Em um controle **FileListBox**, gera um evento PathChange.

**Observação:** Para **DirListBox**, o valor de retorno de **Path** é diferente daquele `List(ListIndex)`, que retorna apenas a seleção.

## Propriedade Zoom

Retorna ou configura a porcentagem pela qual o resultado impresso deve ser escalonado para cima ou para baixo. Não está disponível durante o tempo de criação.

### Sintaxe

*object.Zoom* [= *number*]

A sintaxe da propriedade **Zoom** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Uma <u>expressão numérica</u> que é avaliada como uma porcentagem segundo a qual o resultado impresso deve ser escalonado. O padrão é 0, que especifica que a página impressa aparece com seu tamanho normal.

## Comentários

A configuração da propriedade **Zoom** escalona o tamanho da página física para cima ou para baixo, segundo um fator de Zoom/100, até o tamanho aparente do resultado impresso. Por exemplo, uma página tamanho carta impressa com **Zoom** configurada como 50 contém tantos dados quanto uma página do tamanho 17 por 22 polegadas pois o texto impresso e os elementos gráficos são escalonados para metade de sua altura e largura originais.

**Observação:** O efeito das propriedades do objeto **Printer** depende do driver fornecido pelo fabricante da impressora. Algumas configurações de propriedade podem não ter efeito algum, ou diversas configurações de propriedade diferentes podem ter todas o mesmo efeito. As configurações fora do intervalo aceito podem ou não produzir um erro. Para maiores informações, consulte a documentação do fabricante para o driver específico.

## Propriedade Font

Retorna o objeto **Font**.

### Sintaxe

*object.Font*

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

## Comentários

---

Use a propriedade **Font** de um objeto para identificar um objeto **Font** específico cujas propriedades você deseja usar. Por exemplo, o código abaixo altera a configuração da propriedade **Bold** de um objeto **Font** identificado pela propriedade **Font** de um objeto **TextBox**:

```
txtFirstName.Font.Bold = True
```

---

## Propriedade Container

Retorna ou configura o recipiente de um controle em um **Form**. Não está disponível durante o tempo de criação.

### Sintaxe

**Set** *object.Container* [= *container*]

A sintaxe da propriedade **Container** em estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>container</i>	Uma expressão de objeto que é avaliada como um objeto que pode servir como um recipiente para outros controles, conforme descrito em Comentários.

### Comentários

Os controles abaixo podem conter outros controles:

- Controle **Frame**
- Controle **PictureBox**.

## Propriedade Object

Retorna uma referência a uma propriedade ou método de um controle que tem o mesmo nome que uma propriedade ou método automaticamente estendido ao controle pelo Visual Basic.

### Sintaxe

*object.Object*[*.property*].*method*]

A sintaxe da propriedade **Object** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>property</i>	Propriedade do controle que é idêntica ao nome de uma propriedade fornecida pelo Visual Basic.
<i>Method</i>	Método do controle que é idêntico ao nome de um método fornecido pelo Visual Basic.

### Comentários

**Observação:** A propriedade **Object** retorna o objeto que é a base para o controle, sem as propriedades e métodos automaticamente estendidos ao controle pelo Visual Basic. Portanto, você também pode referir-se às propriedades e métodos "personalizados" do controle através da propriedade **Object**, por exemplo, `Print SSTab1.Object.Tabs`.

O Visual Basic fornece todas ou um conjunto padrão de propriedades e métodos a controles em um projeto Visual Basic. É possível para um controle ou componente ActiveX (por exemplo, o Microsoft Excel ou o Microsoft Word) definir uma propriedade ou método que tenha o mesmo nome que uma destas propriedades ou métodos padrão. Quando isto ocorre, o Visual Basic utiliza automaticamente a propriedade ou método que ele fornece, ao invés de outra com o mesmo nome definido no controle. A propriedade **Object** permite ignorar a propriedade ou método fornecido pelo Visual Basic e usar a propriedade ou método com nome idêntico definidos no controle.

Por exemplo, a propriedade **Tag** é uma propriedade fornecida a todos os controles em um projeto Visual Basic. Se um controle de um projeto tem o nome `ctlDemo`, você acessa a propriedade

**Tag** usando esta sintaxe:

```
ctlDemo.Tag
```

O Visual Basic usa automaticamente a propriedade **Tag** que ele fornece. Entretanto, se o controle define sua própria propriedade **Tag** e você deseja acessar esta propriedade, você usa a propriedade **Object** nesta sintaxe:

```
ctlDemo.Object.Tag
```

O Visual Basic estende automaticamente algumas ou todas as propriedades, métodos e eventos a controles em um projeto do Visual Basic:

### Propriedades

Align	Height	Object
-------	--------	--------

<b>Binding</b>	<b>HelpContextID</b>	<b>Parent</b>
<b>Bindings</b>	<b>Index</b>	<b>TabIndex</b>
<b>Cancel</b>	<b>Left</b>	<b>TabStop</b>
<b>Container</b>	<b>LeftNoRun</b>	<b>TagParent</b>
<b>DataChanged</b>	<b>LinItem</b>	<b>ToolTipText</b>
<b>DataField</b>	<b>LinMode</b>	<b>Top</b>
<b>DataSource</b>	<b>LinTimeout</b>	<b>TopNoRun</b>
<b>Default</b>	<b>LinTopic</b>	<b>VisibleTabStop</b>
<b>DragIcon</b>	<b>Name</b>	<b>WhatsThisHelpID</b>
<b>DragMode</b>	<b>NegotiateLinItem</b>	<b>Width</b>

#### Métodos

<b>Drag</b>	<b>LinSend</b>	<b>ShowWhatsThis</b>
<b>LinExecute</b>	<b>Move</b>	<b>Zorder</b>
<b>LinPoe</b>	<b>Refresh</b>	
<b>LinRequest</b>	<b>SetFocus</b>	

#### Eventos

<b>GotFocus</b>	<b>LinError</b>	<b>LinOpen</b>
<b>LinClose</b>	<b>LinNotify</b>	<b>LostFocus</b>

Se você usa uma propriedade ou método de um controle e não obtém o comportamento esperado, verifique se a propriedade ou método tem o mesmo nome que um destes exibidos na lista acima. Se os nomes corresponderem, verifique a documentação fornecida com o controle para ver se o comportamento corresponde ao da propriedade ou método fornecidos pelo Visual Basic. Se os comportamentos não forem idênticos, pode ser necessário usar a propriedade **Object** para acessar o recurso do controle desejado.

## Propriedade ToolTipText

Retorna ou configura uma [Dica de ferramentas](#).

### Sintaxe

*object.ToolTipText* [= *string*]

A sintaxe da propriedade **ToolTipText** tem estas partes:

<b>Parte</b>	<b>Descrição</b>
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>string</i>	Uma sequência de caracteres associada a um objeto na lista Applies To, que aparece em um pequeno retângulo abaixo do objeto quando o cursor do usuário passa sobre o objeto durante o tempo <u>de execução</u> durante cerca de um segundo.

### Comentários

Se você usa somente uma imagem para rotular um objeto, pode usar esta propriedade para explicar cada objeto com algumas palavras.

Durante o tempo de criação, você pode configurar a sequência de caracteres da propriedade **ToolTipText** na caixa de diálogo de propriedades do controle.

Para os controles **ToolBar** e **TabStrip**, você precisa configurar a propriedade **ShowTips** como True para exibir Dicas de ferramentas.

## Exemplo da propriedade ActiveForm

Este exemplo imprime a hora no formulário filho ativo em um objeto **MDIForm**. Para experimentar este exemplo, crie um **MDIForm**, desenhe um controle **PictureBox** nele e um controle **CommandButton** na **PictureBox**. No Form1, configure a propriedade **MDIChild** como **True**. Você também pode configurar **AutoRedraw** como **True** para manter o texto no formulário mesmo após cobri-lo com outro formulário. Cole o código adequado na seção Declarations de cada formulário e pressione F5.

```
' Copiar todo o código no formulário MDI.
Private Sub MDIForm_Load ()
    Dim NewForm As New Form1 ' Cria uma nova ocorrência de Form1.
    NewForm.Show
```

---

```
End Sub
```

```
Private Sub Command1_Click ()  
    ' Imprime a hora no formulário ativo.  
    ActiveForm.Print "The time is " & Format(Now, "Long Time")  
End Sub
```

Este exemplo mostra como você pode usar o objeto **Clipboard** em operações de recorte, cópia, colagem e exclusão usando botões em uma barra de ferramentas. Para experimentar este exemplo, crie um novo projeto, coloque os controles **TextBox** e **CheckBox** em Form1 e crie um novo formulário MDI. No formulário MDI, coloque um controle **PictureBox** e insira um controle **CommandButton** na **PictureBox**. Configure a propriedade **Index** do **CommandButton** como 0 (criando uma matriz de controles). Configure a propriedade **MDIChild** de Form1 como **True**. Para executar este exemplo, copie o código na seção **Declarations** do **MDIForm** e pressione F5. Observe que quando a **CheckBox** tem o foco, os botões não funcionam, já que **Checkbox** é agora o controle ativo, ao invés de **TextBox**.

```
Private Sub MDIForm_Load ()  
    Dim I ' Declarar variável.  
    Command1(0).Move 0, 0, 700, 300 ' Posicionar o botão na barra de ferramentas.  
    For I = 1 To 3 ' Criar outros botões.  
        Load Command1(I) ' Criar botão.  
        Command1(I).Move I * 700, 0, 700, 300 ' Posicionar e dimensionar botão.  
        Command1(I).Visible = True ' Exibir botão.  
    Next I  
    Command1(0).Caption = "Cut" ' Configurar botão.  
    Command1(1).Caption = "Copy"  
    Command1(2).Caption = "Paste"  
    Command1(3).Caption = "Del"  
End Sub
```

```
Private Sub Command1_Click (Index As Integer)  
    ' ActiveForm refere-se ao formulário ativo no formulário MDI.  
    If TypeOf ActiveForm.ActiveControl Is TextBox Then  
        Select Case Index  
            Case 0 ' Cut.  
                ' Copiar texto selecionado para a Área de transferência.  
                Clipboard.SetText ActiveForm.ActiveControl.SelText  
                ' Excluir o texto selecionado.  
                ActiveForm.ActiveControl.SelText = ""  
            Case 1 ' Copiar.  
                ' Copiar o texto selecionado para a Área de transferência.  
                Clipboard.SetText ActiveForm.ActiveControl.SelText  
            Case 2 ' Colar.  
                ' Colocar o texto da Área de transferência na caixa de texto.  
                ActiveForm.ActiveControl.SelText = Clipboard.GetText()  
            Case 3 ' Excluir.  
                ' Excluir o texto selecionado.  
                ActiveForm.ActiveControl.SelText = ""  
        End Select  
    End If  
End Sub
```

## Exemplo da propriedade Caption

Este exemplo altera a propriedade **Caption** de um controle **CommandButton** toda vez que o usuário clica no botão. Para experimentar este exemplo, cole o código na seção **Declarations** de um formulário contendo um **CommandButton** chamado Command1 e, em seguida, pressione F5 e clique no botão.

```
Private Sub Command1_Click ()  
    ' Verificar a legenda e altera-la Then
```

```

        Command1.Caption = "O"
    Else
        Command1.Caption = "Cliced"
    End If
End Sub

```

## Exemplo da propriedade Checed

Este exemplo exibe e remove uma marca de seleção junto a um item de menu. Para experimentar este exemplo, crie um formulário contendo um controle **Menu** que tenha um item de menu (configure ambas as propriedades **Caption** e **Name** como MyMenuItem) e, em seguida, pressione F5 e escolha o item de menu.

```

Private Sub MyMenuItem_Clic ()
    ' Ativar e desativar a marca de seleção no item de menu.
    MyMenuItem.Checked = Not MyMenuItem.Checked
End Sub

```

## Exemplo da propriedade Enabled

Este exemplo ativa um controle **CommandButton** toda vez que um controle **TextBox** tiver texto. Para experimentar este exemplo, cole o código na seção Declarations de um formulário que contenha controles **CommandButton** e **TextBox** e, em seguida, pressione F5 e digite alguma coisa na caixa de texto.

```

Private Sub Form_Load ()
    Text1.Text = "" ' Limpar a caixa de texto.
    Command1.Caption = "Save" ' Colocar a legenda em um botão.
End Sub

Private Sub Text1_Change ()
    If Text1.Text = "" Then ' Verificar se a caixa de texto está vazia.
        Command1.Enabled = False ' Desativar o botão.
    Else
        Command1.Enabled = True ' Ativar o botão.
    End If
End Sub

```

## Exemplo da propriedade HelpContextID

Este exemplo utiliza tópicos no arquivo da **Ajuda** do Visual Basic para demonstrar como especificar números de contexto para tópicos da **TextBox**. Para experimentar este exemplo, cole o código na seção Declarations de um formulário que contenha um controle **TextBox** e um controle **Frame** com um controle **OptionButton** dentro dele. Pressione F5. Uma vez que o programa estiver em execução, mova o foco para um dos controles e pressione F1.

' Os verdadeiros números de contexto do arquivo da Ajuda do Visual Basic.

```

Const winColorPalette = 21004 ' Definir constantes.
Const winToolbox = 21001
Const winCodeWindow = 21005

```

```

Private Sub Form_Load ()
    App.HelpFile = "VB.HLP"
    Frame1.HelpContextID = winColorPalette
    Text1.HelpContextID = winToolbox
    Form1.HelpContextID = winCodeWindow
End Sub

```

## Exemplo da propriedade Index

Este exemplo inicia com dois controles **OptionButton** e adiciona um novo **OptionButton** ao formulário cada vez que você clica em um controle **CommandButton**. Quando você clica em um

**OptionButton**, a propriedade **FillStyle** é configurada e um novo círculo é desenhado. Para experimentar este exemplo, cole o código na seção Declarations de um formulário que tenha dois controles **OptionButton**, um **CommandButton** e um grande controle **PictureBox**. Configure a propriedade **Name** de ambos os controles **OptionButton** como **optButton** para criar uma matriz de controles.

```
Private Sub OptButton_Clic (Index As Integer)
    Dim H, W ' Declara variáveis.
    Picture1.Cls ' Limpar a figura.
    Picture1.FillStyle = Index ' Configurar FillStyle.
    W = Picture1.ScaleWidth / 2 ' Obter o tamanho de um círculo.
    H = Picture1.ScaleHeight / 2
    Picture1.Circle (W, H), W / 2 ' Desenhar círculo.
End Sub

Private Sub Command1_Clic ()
    Static MaxIdx ' O maior indica na matriz.
    If MaxIdx = 0 Then MaxIdx = 1 ' Preconfigurar MaxIdx.
    MaxIdx = MaxIdx + 1 ' Incrementar o índice.
    If MaxIdx > 7 Then Exit Sub ' Colocar oito botões no formulário.
    Load OptButton(MaxIdx) ' Criar um novo item na matriz.
    ' Configurar a localização de novo botão de opção sob o botão anterior.
    OptButton(MaxIdx).Top = OptButton(MaxIdx - 1).Top + 360
    OptButton(MaxIdx).Visible = True ' Tornar visível o novo botão.
End Sub
```

## Exemplo da propriedade Parent

Este exemplo passa um controle de um formulário que não tem o foco para um procedimento em um módulo e então exibe o estado do controle no formulário pai. Para experimentar este exemplo, crie três formulários, Form1 contendo um controle **CommandButton**, Form2 e Form3 contendo cada um deles um controle **CheckBox**. Você também deve criar um novo módulo (clique em **Add Module** no menu **Project**). Cole o código nas seções Declarations dos respectivos formulários e módulos, e então pressione F5 para executar o programa.

```
' Inserir este código em Form1.
Private Sub Form_Load ()
    Form2.Show ' Exibir todos os formulários.
    Form3.Show
    Form2.AutoRedraw = True
    Form3.AutoRedraw = True
End Sub

Private Sub Command1_Clic ()
    ReadCheckBox Form2.Chec1 ' Chamar procedimento em outro módulo
    ReadCheckBox Form3.Chec1 ' e enviar o controle como argumento.
End Sub

' Inserir este código em Module1.
Sub ReadCheckBox (Source As Control)
    If Source.Value Then
        Source.Parent.Cls ' Limpar o formulário pai.
        Source.Parent.Print "CheckBox is On." ' Exibir no formulário pai.
    Else
        Source.Parent.Cls ' Limpar o formulário pai.
        Source.Parent.Print "CheckBox is Off." ' Exibir no formulário pai.
    End If
End Sub
```

## Exemplo da propriedade Path

Este exemplo exibe uma lista de arquivos para a unidade de disco e diretório selecionados. Para experimentar este exemplo, cole o código na seção Declarations de um formulário que contenha controles **DriveListBox**, **DirListBox** e **FileListBox**. Pressione F5. Use o mouse para alterar a

---

unidade de disco ou diretório.

```
Private Sub Drive1_Change ()
```

```
    Dir1.Path = Drive1.Drive ' Configurar o caminho do diretório.
```

```
End Sub
```

```
Private Sub Dir1_Change ()
```

```
    File1.Path = Dir1.Path ' Configurar o caminho do arquivo.
```

```
End Sub
```

## Exemplo da propriedade Container

Este exemplo demonstra a movimentação de um controle **CommandButton** de recipiente para recipiente em um objeto **Form**. Para experimentar este exemplo, cole o código na seção **Declarations** de um formulário que contenha um controle **Frame**, um controle **PictureBox** e um **CommandButton** e então pressione F5.

```
Private Sub Form_Click()
```

```
    Static intX As Integer
```

```
    Select Case intX
```

```
        Case 0
```

```
            Set Command1.Container = Picture1
```

```
            Command1.Top= 0
```

```
            Command1.Left= 0
```

```
        Case 1
```

```
            Set Command1.Container = Frame1
```

```
            Command1.Top= 0
```

```
            Command1.Left= 0
```

```
        Case 2
```

```
            Set Command1.Container = Form1
```

```
            Command1.Top= 0
```

```
            Command1.Left= 0
```

```
    End Select
```

```
    intX = intX + 1
```

```
End Sub
```

## Propriedade DisabledPicture

Retorna ou configura uma referência a uma figura a ser exibida em um controle quando este é desativado (ou seja, quando sua propriedade **Enabled** estiver configurada como **False**).

### Sintaxe

*object.DisabledPicture* [= *picture*]

A sintaxe da propriedade **DisabledPicture** tem estas partes:

Parte	Descrição
<i>object</i>	Uma expressão de <u>objeto</u> que avalia para um objeto na lista <b>Applies To</b> .
<i>picture</i>	Um objeto <b>Picture</b> contendo um elemento gráfico, conforme descrito em <b>Configurações</b> .

### Configurações

As configurações para *picture* são:

Configuração	Descrição
(None)	(Padrão) Nenhuma figura.
(Bitmap, icon, metafile)	Especifica um elemento gráfico. Você pode carregar o elemento gráfico na <u>janela <b>Properties</b></u> durante o tempo de criação. Durante o tempo de execução, você também pode configurar esta propriedade usando a função <b>LoadPicture</b> em um <u>bitmap</u> , <u>ícone</u> , ou <u>metarquivo</u> , ou configurando-a na propriedade <b>Picture</b> de outro controle.

### Comentários

A propriedade **DisabledPicture** especifica um objeto de figura a ser exibido quando o controle como **CommandButton** for desativado. A propriedade **DisabledPicture** será ignorada, a não ser



que a propriedade **Style** esteja configurada como 1 (gráfica).

A figura é centralizada de maneira horizontal e vertical no controle. Se existir uma legenda, assim como uma figura, esta é centralizada acima da legenda. Se o objeto figura for grande demais para caber no controle, ele é então recortado.

Se nenhuma figura for atribuída à propriedade **DisabledPicture**, mas uma for atribuída à propriedade **Picture**, então uma versão acinzentada daquela figura será exibida quando o controle estiver desativado.

## Propriedade DownPicture

Retorna ou configura uma referência a uma figura a ser exibida em um controle quando este é clicado e na posição abaixada (pressionado).

### Sintaxe

*object*.**DownPicture** [= *picture*]

A sintaxe da propriedade **DownPicture** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>picture</i>	Um objeto <b>Picture</b> contendo um elemento gráfico, conforme descrito em Configurações.

### Configurações

As configurações para *picture* são:

Configuração	Descrição
(None)	(Padrão) Nenhuma figura.
(Bitmap, icon, metatile)	Especifica um elemento gráfico. Você pode carregar o elemento gráfico a partir da <u>janela Properties</u> durante o tempo de criação. Durante o tempo de execução, você também pode configurar esta propriedade usando a função <b>LoadPicture</b> em um <u>bitmap</u> , <u>ícone</u> , ou <u>metarquivo</u> ou configurando-o como a propriedade <b>Picture</b> de outro controle.

### Comentários

A propriedade **DownPicture** refere-se a um objeto de figura que é exibido quando o botão está no estado comprimido. A propriedade **DownPicture** é ignorada, a menos que a propriedade **Style** esteja configurada como 1 (gráfica). Observe que quando a propriedade **Style** de um controle **OptionButton** ou **CheckBox** estiver configurada como gráfica e seu botão pressionado, o segundo plano do botão estará pontilhado, mas a figura no botão não.

A figura é centralizada de maneira horizontal e vertical no botão. Se existir uma legenda incluída com a figura, esta será centralizada acima da legenda. Se nenhuma figura for atribuída a esta propriedade quando o botão estiver pressionado, então a figura atualmente atribuída à propriedade **Picture** será utilizada. Se nenhuma figura for atribuída à propriedade **Picture** ou **DownPicture**, então somente a legenda será exibida. Se o objeto de figura for grande demais para caber no botão, então ele será recortado.

## Propriedade Item

Retorna um membro específico de um objeto **Collection**, por posição ou por chave.

### Sintaxe

*object*.**Item**( *index* )

A sintaxe da propriedade **Item** tem o seguinte qualificador de objeto e parte:

Parte	Descrição
<i>object</i>	Obrigatório. Uma expressão de objeto que avalia para um objeto na lista Applies To.
<i>index</i>	Obrigatório. Uma expressão que especifica a posição de um membro da coleção. Se for uma expressão numérica, o

---

índice deve ser um número de 1 até o valor da propriedade Count da coleção. Se for uma expressão de seqüência de caracteres, o índice deve corresponder ao argumento-chave especificado quando o membro referido foi adicionado à coleção.

### **Comentários**

Se o valor fornecido como índice não corresponder a qualquer membro existente da coleção, ocorrerá um erro.

**Item** é a propriedade padrão para uma coleção. Portanto, as linhas de código abaixo são equivalentes:

```
Print MyCollection(1)
```

```
Print MyCollection.Item(1)
```

---

## Propriedade MasColor

Retorna ou configura uma cor em uma figura de botão como uma "máscara" (ou seja, transparente).

### Sintaxe

*object.MasColor* [= *color*]

A sintaxe da propriedade **MasColor** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>color</i>	Um valor ou <u>constante</u> que determina a cor a ser utilizada como máscara, conforme descrito em Configurações.

### Configurações

O Visual Basic usa o esquema de cores RGB (vermelho-verde-azul) do ambiente operacional Windows. As configurações para *color* são:

Configuração	Descrição
Cores RGB normais	As cores especificadas usando a paleta Color ou usando as funções <b>RGB</b> ou <b>QBColor</b> no código.
&H00C0C0C0	(Padrão) Cinza claro.

### Comentários

Se as cores do sistema forem alteradas, então a cor que for transparente se alterará, tornando o aspecto de sua figura imprevisível. É uma boa prática de programação usar cores que não pertençam ao sistema.

Esta propriedade somente é usada quando a propriedade **UseMasColor** for configurada como **True**, e o botão tiver uma figura no estilo bitmap atribuída a sua propriedade **Picture** (ícones e metarquivos já contêm informações de transparência).

Se a propriedade **MasColor** for alterada durante o tempo de execução, o botão se redesenhará com a nova cor funcionando como máscara.

## Propriedade UseMasColor

Retorna ou configura um valor que determina se a cor atribuída à propriedade **MasColor** é usada como uma "máscara" (ou seja, usada para criar áreas transparentes).

### Sintaxe

*object.UseMasColor* [= *boolean*]

A sintaxe da propriedade **UseMasColor** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> que especifica se a cor atribuída à propriedade <b>MasColor</b> é usada como uma máscara.

### Configurações

As configurações para *boolean* são:

Configuração	Descrição
<b>True</b>	A cor designada à propriedade <b>MasColor</b> é usada como uma máscara, criando uma área transparente em qualquer lugar que a cor estiver.
<b>False</b>	(Padrão) A cor designada à propriedade <b>MasColor</b> é ignorada e a cor permanece opaca.

## Propriedade WhatsThisHelpID

Retorna ou configura um número de contexto associado para um objeto. Use para oferecer ajuda

---

vinculada ao contexto para seu aplicativo usando a pop-up **O que é isto?** na **Ajuda** do Windows 95.

### Sintaxe

*object*.WhatsThisHelpID [= *number*]

A sintaxe da propriedade **WhatsThisHelpID** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>number</i>	Uma <u>expressão numérica</u> especificando um número de contexto da <b>Ajuda</b> , conforme descrito em Configurações.

### Configurações

As configurações para *number* são:

Configuração	Descrição
0	(Padrão) Nenhum número de contexto especificado.
>0	Um número inteiro especificando o número de contexto válido para o tópico <b>O que é isto?</b> associado ao objeto.

### Comentários

O Windows 95 usa o botão **O que é isto?** no canto superior direito da janela para iniciar a **Ajuda** do Windows e carregar um tópico identificado pela propriedade **WhatsThisHelpID**.

## Propriedade WhatsThisButton

Retorna ou configura um valor que determina se o botão **O que é isto?** aparece na barra de título de um objeto **Form**. Somente para leitura durante o tempo de execução.

### Sintaxe

*object*.WhatsThisButton

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Configurações

As configurações para a propriedade **WhatsThisButton** são:

Configuração	Descrição
<b>True</b>	Ativa a exibição do botão da <b>Ajuda O que é isto?</b> .
<b>False</b>	(Padrão) Desativa a exibição do botão da <b>Ajuda O que é isto?</b> .

### Comentários

A propriedade **WhatsThisHelp** deve ser **True** para que a propriedade **WhatsThisButton** seja **True**. Além disso, as propriedades abaixo também devem ser configuradas como mostrado:

- Propriedade **ControlBox** = **True**
- Propriedade **BorderStyle** = Fixed Single ou Sizable
- **MinButton** e **MaxButton** = **False**
- Ou –
- Propriedade **BorderStyle** = Fixed Dialog

## Propriedade WhatsThisHelp

Retorna ou configura um valor que determina se a ajuda vinculada ao contexto utiliza a pop-up **O que é isto?** fornecida pela **Ajuda** do Windows 95 ou a janela da **Ajuda** principal. Somente leitura durante o tempo de execução.

### Sintaxe

*object*.WhatsThisHelp [= *boolean*]

A sintaxe da propriedade **WhatsThisHelp** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Um valor que determina se a <b>Ajuda</b> usa a pop-up <b>O que é isto?</b> , conforme descrito em Configurações.

---

## Configurações

As configurações para *boolean* são:

Configuração	Descrição
True	O aplicativo usa uma das técnicas de acesso de <b>O que é isto?</b> para iniciar a <b>Ajuda</b> do Windows e carregar um tópico identificado pela propriedade <b>WhatsThisHelpID</b> .
False	(Padrão) O aplicativo usa a tecla F1 para iniciar a <b>Ajuda</b> do Windows e carregar o tópico identificado pela propriedade <b>HelpContextID</b> .

## Comentários

Existem três técnicas de acesso para oferecer **Ajuda O que é isto?** a um aplicativo. A propriedade **WhatsThisHelp** deve ser configurada como True para que qualquer uma destas técnicas funcione.

- Oferecer um botão **O que é isto?** na barra de título do formulário usando a propriedade **WhatsThisButton**. O ponteiro do mouse se transforma no estado **O que é isto?** (seta com ponto de interrogação). O tópico exibido é identificado pela propriedade **WhatsThisHelpID** do controle clicado pelo usuário.
- Acionar o método **WhatsThisMode** de um formulário. Isto produz o mesmo comportamento que clicar no botão **O que é isto?** sem usar um botão. Por exemplo, você pode acionar este método a partir de um comando em um menu na barra de menus de seu aplicativo.
- Chamar o método **ShowWhatsThis** para um determinado controle. O tópico exibido é identificado pela propriedade **WhatsThisHelpID** do controle.

## Método ShowWhatsThis

Exibe um tópico selecionado em um arquivo da **Ajuda** usando a pop-up **O que é isto?** oferecida pela **Ajuda** do Windows 95.

### Sintaxe

*object*.**ShowWhatsThis**

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

O método **ShowWhatsThis** é muito útil para oferecer ajuda vinculada ao contexto a partir de um menu de contexto em seu aplicativo. O método exibe o tópico identificado pela propriedade **WhatsThisHelpID** do objeto especificado na sintaxe.

## Exemplo de método ShowWhatsThis

Este exemplo exibe o tópico da **Ajuda O que é isto?** para um controle **CommandButton** selecionando um comando de menu em um menu de contexto criado para o botão. Configura a propriedade **WhatsThisHelp** do formulário como **True**. Coloca um controle **CommandButton** em um formulário, cria um menu usando o Editor Menu com um item invisível de alto nível chamado **mnuBtnContextMenu** e um submenu chamado **mnuBtnWhatsThis** com a legenda **O que é isto?**.

```
Private ThisControl As Control
```

```
Private Sub Command1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = vbRightButton Then
        Set ThisControl = Command1
        PopupMenu mnuBtnContextMenu
    End If
    Set ThisControl = Nothing
End Sub
```

```
Private Sub mnuBtnWhatsThis_Clic()
    ThisControl.ShowWhatsThis
End Sub
```

---

## Método WhatsThisMode

```
{ewc HLP95EN.DLL,DYNALIN,"Consulte também":"vbmthWhatsThisModeC;vbproBoosOnlineJumpTopic"} {ewc  
HLP95EN.DLL,DYNALIN,"Exemplo":"vbmthWhatsThisModeX":1} {ewc HLP95EN.DLL,DYNALIN,"Aplica-se  
a":"vbmthWhatsThisModeA"} {ewc HLP95EN.DLL,DYNALIN,"Especificidades":"vbmthWhatsThisModeS"}
```

Provoca a transformação do ponteiro do mouse no ponteiro **O que é isto?** e prepara o aplicativo para exibir a **Ajuda O que é isto?** sobre o objeto selecionado.

### Sintaxe

*object*.WhatsThisMode

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Executar o método **WhatsThisMode** coloca o aplicativo no mesmo estado que você obtém clicando o botão **O que é isto?** na barra de título. O ponteiro do mouse se altera para o ponteiro **O que é isto?**. Quando o usuário clica em um objeto, a propriedade **WhatsThisHelpID** do objeto clicado é usada para acionar a ajuda vinculada ao contexto. Este método é especialmente útil ao se chamar a **Ajuda** a partir de um menu na barra de menus de seu aplicativo.

## Exemplo de método WhatsThisMode

Este exemplo usa um comando em um menu para alterar o ponteiro do mouse para o ponteiro **O que é isto?** e ativar a ajuda vinculada ao contexto. Para experimentar este exemplo, crie um menu e cole o código no evento Clic de um dos controles **Menu**. Pressione F5 e clique no comando de menu para alternar o aplicativo para o estado **O que é isto?**.

```
Private Sub mnuContextHelp_Clic ()  
    Form1.WhatsThisMode  
End Sub
```

## Propriedade ShowInTasbar

Retorna ou configura um valor que determina se o objeto **Form** aparece na barra de ferramentas do Windows 95. Somente leitura durante o tempo de execução.

### Sintaxe

*object*.ShowInTasbar

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Configurações

As configurações para a propriedade **ShowInTasbar** são:

Configuração	Descrição
<b>True</b>	(Padrão) O objeto <b>Form</b> aparece na barra de tarefas.
<b>False</b>	O objeto <b>Form</b> não aparece na barra de tarefas.

### Comentários

Use a propriedade **ShowInTasbar** para impedir que caixas de diálogo em seu aplicativo apareçam na barra de tarefas.

O valor padrão para a propriedade **ShowInTasbar** assume a configuração padrão para a propriedade **BorderStyle** do objeto **Form** (Sizable). Alterar a propriedade **BorderStyle** pode alterar a configuração da propriedade **ShowInTasbar**.

## Propriedade BorderStyle (Controles ActiveX)

Retorna ou configura o estilo de borda de um objeto.

### Sintaxe

*object*.BorderStyle [= *value*]

A sintaxe da propriedade **BorderStyle** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>value</i>	Um valor ou constante que determina o estilo de

---

borda, conforme descrito em Configurações.

### Configurações

As configurações de *value* são:

Constante	Valor	Descrição
<b>ccNone</b>	0	(Padrão) Nenhuma borda ou elementos relacionados com bordas.
<b>ccFixedSingle</b>	1	Única e fixa.

**Obs.:** O prefixo cc refere-se aos controles do Windows 95. Para outros controles, os prefixos das configurações se alteram com o controle específico ou grupo de controles. Entretanto, a descrição permanece a mesma, a menos que seja indicado.

### Observações

Configurar **BorderStyle** para um controle **ProgressBar** diminui o tamanho dos blocos exibidos pelo controle.

---

## Propriedade Image (Controles ActiveX)

Retorna ou configura um valor que especifica qual objeto **ListImage** em um controle **ImageList** deve ser usado com outro objeto.

### Sintaxe

*object.Image* [= *index*]

A sintaxe da propriedade **Image** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	Um número inteiro ou sequência de caracteres exclusiva especificando o objeto <b>ListImage</b> a ser usado com <i>object</i> . O número inteiro é o valor da propriedade <b>Index</b> ; a sequência de caracteres é o valor da propriedade <b>ey</b> .

### Comentários

Antes de configurar a propriedade **Image**, você deve associar um controle **ImageList** a um controle **ToolBar**, **TreeView** ou **TabStrip** configurando a propriedade **ImageList** de cada controle como um controle **ImageList**.

Durante o tempo de criação, coloque um controle **ImageList** no formulário e carregue-o com imagens, cada uma delas sendo um objeto **ListImage** a que foi atribuído um número de índice na coleção **ListImages**. Na guia **General**, da caixa de diálogo **Property Pages** do controle, selecione a **ImageList** desejada na caixa de listagem **ImageList**, como **ImageList1**. Para objetos **Tab** e **Button**, você também pode especificar a imagem que deseja associar a estes objetos digitando o número de índice do objeto **ListImage** específico no campo **Image** da guia **Tabs** ou **Buttons**.

Durante o tempo de execução, use código como mostrado abaixo para associar uma **ImageList** a um controle e uma **ListImage** a um objeto específico:

```
Set TabStrip1.ImageList=ImageList1
TabStrip1.Tabs(1).Image=2
```

Use a propriedade **ey** para especificar um objeto **ListImage** do controle **ImageList** quando você deseja que seu código seja auto-documentado, como segue:

```
' Presumindo-se que exista um objeto ListImage com a propriedade ey =
' "fechar", usar esta imagem para um botão Toolbar.
Toolbar1.Buttons(1).Image = "fechar"
```

```
' Isto é mais fácil de ser lido que simplesmente especificar um valor
Index, como abaixo:
```

```
Toolbar1.Buttons(1).Image = 4 ' Exigir que o objeto ListImage
' com a propriedade Index = 4 seja a imagem "fechar".
```

O valor da propriedade **Index** de um objeto pode se alterar quando objetos na coleção forem reclassificados, por exemplo, quando você configura a propriedade **Sorted** como **True**. Se você esperar que a propriedade **Index** se altere dinamicamente, pode ser mais útil referir-se a objetos em uma coleção usando a propriedade **ey**.

Se não existirem imagens para uma coleção **Tabs**, o valor de *index* é -1.



---

## Propriedade **ey** (Controles ActiveX)

Retorna ou configura uma seqüência de caracteres que identifica com exclusividade um membro de uma coleção.

### Sintaxe

*object.ey* [= *string*]

A sintaxe da propriedade **ey** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>string</i>	Uma seqüência de caracteres identificando um membro de uma coleção.

### Comentários

Se a seqüência de caracteres não for exclusiva, ocorrerá um erro.

Você pode configurar a propriedade **ey** quando usar o método **Add** para adicionar um objeto a uma coleção.

O valor da propriedade **Index** de um objeto pode se alterar quando objetos na coleção forem re-ordenados como, por exemplo, quando você configura a propriedade **Sorted** como **True**. Se você esperar que a propriedade **Index** se altere dinamicamente, refira-se a objetos em uma coleção usando a propriedade **ey**.

Além disso, você pode usar a propriedade **ey** para tornar seu projeto Visual Basic "auto-documentado" atribuindo nomes significativos a objetos em uma coleção.

## Propriedade **ImageList** (Controles ActiveX)

Retorna ou configura o controle **ImageList**, se algum, que está associado a outro controle.

### Sintaxe

*object.ImageList* [= *imagelist*]

A sintaxe da propriedade **ImageList** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que é avaliada como um objeto na lista Applies To.
<i>imagelist</i>	Uma referência de objeto que especifica qual controle <b>ImageList</b> deve ser usado.

### Comentários

Para que um controle use a propriedade **ImageList**, você deve colocar um controle **ImageList** no formulário. A seguir, durante o tempo de criação, você pode configurar a propriedade **ImageList** na caixa de diálogo **Property Pages** do controle associado. Para associar uma **ImageList** a um controle durante o tempo de execução, configure a propriedade **ImageList** do controle como o controle **ImageList** que você deseja usar, como no exemplo abaixo:

```
Set TabStrip1.ImageList = ImageList1
```

---

## Método Clear (Controles ActiveX)

Remove todos os objetos em uma coleção.

### Sintaxe

*object*.Clear

O espaço reservado *object* representa uma expressão de objeto que avalia para um objeto na lista Applies To.

### Comentários

Para remover apenas um objeto de uma coleção, use o método **Remove**.

## Exemplo do método Clear (Controles ActiveX)

Este exemplo adiciona seis objetos **Panel** a um controle **StatusBar**, criando um total de sete objetos **Panel**. Um clique no formulário limpa todos os objetos **Panel** quando o número atinge sete. Se o número de objetos **Panel** for menor que sete, cada clique no formulário adicionará um novo objeto **Panel** ao controle, até que o número sete seja novamente atingido. Para experimentar este exemplo, coloque um controle **StatusBar** em um formulário e cole o código na seção Declarati-  
ons. Execute o exemplo e clique no formulário para limpar todos os objetos **Panel** e subsequen-  
temente adicionar objetos **Panel**.

```
Private Sub Form_Load()  
    Dim pnlX As Panel ' Declarar variável de objeto para objetos Panel.  
    Dim I As Integer  
  
    ' Adicionar 6 objetos Panel ao único objeto Panel padrão,  
    ' fazendo 7 objetos Panel.  
    For I = 1 to 6  
        Set pnlX = StatusBar1.Panels.Add  
    Next I  
End Sub  
  
Private Sub Form_Clic()  
    ' Se a Count da coleção for 7, limpar a coleção.  
    ' Caso contrário, adicionar um Panel e usar a propriedade Count da  
    coleção  
    ' para configurar seu Style.  
    If StatusBar1.Panels.Count = 7 Then  
        StatusBar1.Panels.Clear  
    Else  
        Dim pnlX As Panel  
        Set pnlX = StatusBar1.Panels.Add( , , "simple", 0)  
        ' A propriedade Style é enumerada de 0 a 6. Use a propriedade -1  
de Count de Panels  
        ' para configurar a propriedade Style do novo Panel.  
        ' Exibir todos os painéis, independente da largura do formulário.  
        pnlX.minwidth = TextWidth("simple")  
        pnlX.AutoSize = sbrSpring  
        pnlX.Style = StatusBar1.Panels.Count - 1  
    End If  
End Sub
```

---

## Método Remove (Controles ActiveX)

Remove um membro específico de uma coleção.

### Sintaxe

*object.Remove index*

A sintaxe do método **Remove** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>index</i>	Um número inteiro ou sequência de caracteres que identifica com exclusividade o objeto na coleção. Um número inteiro especifica o valor da propriedade <b>Index</b> ; uma sequência de caracteres especifica o valor da propriedade <b>ey</b> .

### Comentários

Para remover todos os membros de uma coleção, use o método **Clear**.

## Exemplo do método Remove (Controles ActiveX)

Este exemplo adiciona seis objetos **Panel** a um controle **StatusBar** criando um total de sete objetos **Panel**. Quando você clica no formulário, o código verifica quantos objetos **Panel** existem. Se houver apenas um objeto **Panel**, o código adiciona seis objetos **Panel**. Caso contrário, ele remove o primeiro painel. Para experimentar este exemplo, coloque um controle **StatusBar** em um formulário e cole o código na seção Declarations. Execute o exemplo e clique no formulário para remover um objeto **Panel** de cada vez e subsequente adicionar objetos **Panel**.

```
Private Sub Form_Load()  
    Dim pnlX As Panel      ' Declarar variável de objeto para objetos Pa-  
nel.  
    Dim i As Integer  
  
    ' Adicionar 6 objetos Panel ao único objeto Panel padrão,  
    ' fazendo 7 objetos Panel.  
    For i = 1 To 6  
        Set pnlX = StatusBar1.Panels.Add(, , , i)  
        pnlX.AutoSize = sbrSpring  
    Next i  
End Sub  
  
Private Sub Form_Clic()  
    ' Se Count da coleção for 1, adicionar 6 objetos Panel.  
    ' Caso contrário, remover o primeiro painel da coleção.  
    If StatusBar1.Panels.Count = 1 Then  
        Dim sbrX As Panel  
        Dim i As Integer  
        For i = 1 To 6 ' Cada painel tem seu estilo configurado por i.  
            Set sbrX = StatusBar1.Panels.Add(, , , i)  
            sbrX.AutoSize = sbrSpring  
        Next i  
    Else ' Remover o primeiro painel.  
        StatusBar1.Panels.Remove 1  
    End If  
End Sub
```

---

## Propriedade HideSelection (Controles ActiveX)

Retorna ou configura um valor que especifica se o item selecionado permanece realçado quando o controle perde o foco.

### Sintaxe

*object.HideSelection* [ = *boolean*]

A sintaxe da propriedade **HideSelection** tem estas partes:

Parte	Descrição
<i>object</i>	Uma <u>expressão de objeto</u> que avalia para um objeto na lista Applies To.
<i>boolean</i>	Uma <u>expressão Booleana</u> especificando como um controle é exibido ao perder o foco, conforme descrito em Configurações.

### Configurações

As configurações de *boolean*:

Configuração	Descrição
<b>True</b>	(Padrão) Os itens no controle não estão mais selecionados quando o controle perde o foco.
<b>False</b>	Os itens permanecem selecionados após o controle ter perdido o foco.

### Comentários

Normalmente, os itens selecionados em um controle são ocultados quando o controle perde o foco. Esta é a ação padrão da propriedade.

Se você desejar que os itens selecionados permaneçam selecionados após o controle ter perdido o foco, configure a propriedade **HideSelection** como **False**.

## Constantes de controle ActiveX

As constantes abaixo são reconhecidas pelos controles ActiveX. Como resultado, elas podem ser usadas em qualquer lugar de seu código substituindo valores efetivos.

- **Constantes BorderStyle**
- **Constantes MousePointer**

Use o **Object Browser** para visualizar as constantes intrínsecas que você pode usar com métodos e propriedades. A partir do menu **View**, escolha **Object Browser**, selecione a biblioteca de controles adequada e o objeto **Constants**. Você pode rolar as constantes que aparecem no painel **Members**.

**Obs.:** Os prefixos de constantes alteram-se com o controle ou grupo de controles específicos. Entretanto, a descrição permanece a mesma, a não ser que seja indicado.

---

## Constantes MousePointer

Constante	Valor	Descrição
<b>ccDefault</b>	0	(Padrão) Forma determinada pelo objeto.
<b>ccArrow</b>	1	Seta.
<b>ccCross</b>	2	Cruz (ponteiro em forma de cruz).
<b>ccIbeam</b>	3	Formato de I.
<b>ccIcon</b>	4	Ícone (pequeno quadrado dentro de outro quadrado).
<b>ccSize</b>	5	Tamanho (seta de quatro pontas, apontando para norte, sul, leste e oeste).
<b>ccSizeNESW</b>	6	Tamanho NE SW (seta dupla apontando para nordeste e sudoeste).
<b>ccSizeNS</b>	7	Tamanho N S (seta dupla apontando para o norte o sul).
<b>ccSizeNWSE</b>	8	Tamanho NW, SE.
<b>ccSizeEW</b>	9	Tamanho E W (seta dupla apontando para leste e oeste).
<b>ccUpArrow</b>	10	Seta acima.
<b>ccHourglass</b>	11	Ampulheta (espera).
<b>ccNoDrop</b>	12	Sem soltar.
<b>ccArrowHourglass</b>	13	Seta e ampulheta.
<b>cc ArrowQuestion</b>	14	Seta e ponto de interrogação.
<b>ccSizeAll</b>	15	Dimensionar tudo.
<b>ccCustom</b>	99	Ícone personalizado especificado pela propriedade <b>Mou- selcon</b> .

**Obs.:** O prefixo cc é reservado para controles ActiveX. Os prefixos de constantes podem alterar-se com o controle ou grupo de controles específicos. Entretanto, a descrição permanece a mesma, a menos que seja indicado.

## Constantes BorderStyle (Controles ActiveX)

Constante	Valor	Descrição
<b>ccNone</b>	0	(Padrão) Nenhuma borda ou elemento relacionado com borda.
<b>CcFixedSingle</b>	1	(Padrão para controle <b>ListView</b> ) Única Fixa. Existe uma borda de linha simples ao redor do controle.

**Obs.:** O prefixo cc é reservado para controles ActiveX. Os prefixos de constantes podem se alterar com o controle ou grupo de controles específicos. Entretanto, a descrição permanece a mesma, a não ser que seja indicado.