DDE - Dynamic Data Exchange

Objetivo

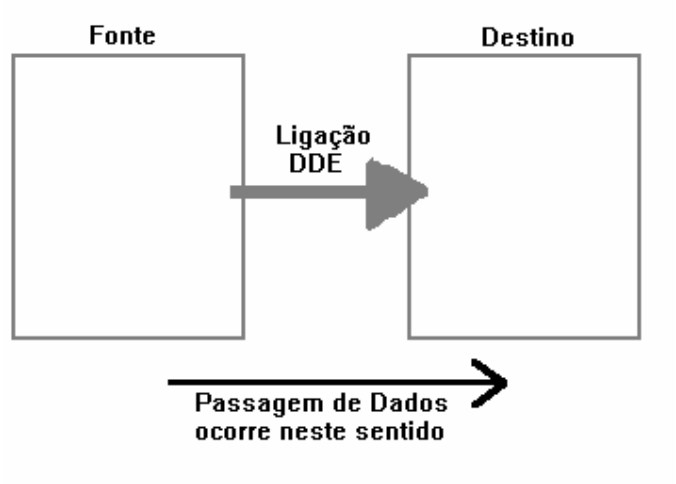
Neste módulo será mostrado a forma que o Visual Basic implementa o DDE (Dynamic Data Exchange), que é a maneira pela qual o Windows permite que aplicações diferentes comuniquem entre si.

Conceituação de DDE

DDE é um mecanismo do Windows que permite que duas aplicações comuniquem entre si, continuamente e automaticamente trocando dados.

Uma comunicação DDE é direta, ao contrário da comunicação entre aplicações quando se usa o Clipboard, que é na realidade em ponto intermediário entre duas aplicações Windows.

Aplicação Origem e Aplicação Destino



Uma Ligação DDE envolve duas partes : uma que envia dados, denominada aplicação fonte, e outra que recebe os dados, chamada de aplicação destino.

Em Visual Basic, apenas alguns controles podem servir de destino de ligações DDE. São eles : TextBoxes, PictureBoxes, e Labels.

Tipos de Ligação DDE

Existem três tipos de ligação DDE no Visual Basic : Automática, Manual e Notificada.

Na ligação automática, qualquer mudança ocorrida na aplicação fonte, é refletida imediatamente na aplicação destino.

Na Ligação manual, a a aplicação destino apenas é atualizada a pedido dela mesma, ou seja, a aplicação fonte pode sofrer inúmeras alterações sem que elas reflitam na aplicação destino.

Na ligação Notificada, cada vez que a aplicação fonte sofre uma alteração, um evento ocorre na aplicação destino, notificando-a que houveram mudanças na aplicação fonte.

Criação de Ligação DDE em Design Time

Uma ligação pode ser criada tanto em Design-time quanto em Run-time. Em Design-Time, você utiliza as facilidades de Corte e Colagem (*Cut* e *Paste*) do Windows. Ao fazer a colagem, O VB automaticamente determina todas as propriedades necessárias para fazer uma ligação. Ao mesmo tempo que a criação de uma DDE em design-time é bastante fácil, este tipo de criação lhe tira a flexibilidade de trabalho. Por exemplo, uma ligação deste tipo será sempre do tipo automática. e isto somente poderá ser alterado, se você escrever código para isto.

Para fazer uma ligação em Run-Time, você próprio determina as mesmas propriedades, que o VB determina quando a ligação é feita em design-time, só que via programação.

Exemplo - Criação de uma Ligação em *Design Time*

Para exemplificar a criação de uma ligação DDE em *design-time* usaremos o Word for Windows como aplicação fonte e o VB com aplicação destino.

Preparando o ambiente no Word for Windows

1. Inicialize o Word for Windows.
2. Digite dois parágrafos quaisquer da sua apostila e salve o documento com o nome de *Source.doc*.
3. Selecione o texto digitado e copie-o para o Clipboard do Windows.

Criando a Ligação DDE no Visual Basic

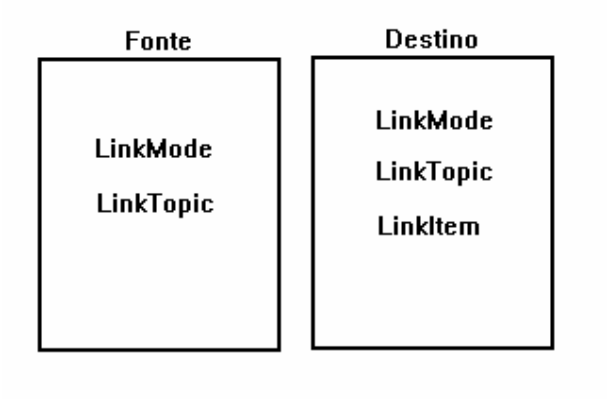
4. Inicialize o Visual Basic.
5. Crie um formulário contendo apenas um TextBox.

6. Com o TextBox selecionado, pressione o mouse sobre o comando *Paste Link* do menu Editar.

Executando a Aplicação

7. Pressione F5 ou selecione Start do Menu Run.
8. Pare a aplicação e verifique quais as propriedades da textbox são novas para você. Verifique e procure compreender o conteúdo destas variáveis.

Propriedades Utilizadas no estabelecimento de Ligações DDE



Já foi visto acima como criar ligações DDE em *design-time*. Vimos também que um TextBox possui uma série de propriedades que quando "setadas", permitem a uma aplicação VB fazer uma ligação dinâmica a um outro aplicativo qualquer do Windows. Neste tópico veremos cada uma destas propriedades e qual é o seu papel dentro deste importante mecanismo.

Aplicação Destino

Quando uma aplicação VB atua como destino de uma ligação DDE, as seguintes propriedades do controle que será destino da ligação DDE devem ter seus valores determinados : LinkMode, LinkTopic e LinkItem. A propriedade LinkTimeout também pode ter seu valor determinado, não sendo porém imprescindível para o estabelecimento da DDE.

Link Mode ### Determina o tipo da ligação estabelecida. Pode assumir os seguintes valores :

Valor	Tipo da Ligação
0	Nenhuma Ligação
1	Automática
2	Manual
3	Notificada

LinkTopic ### Determina a aplicação fonte e o tópico da conversação. O conteúdo da propriedade LinkTopic, associado ao conteúdo da propriedade LinkItem especificam inteiramente a ligação com o outro aplicativo. Normalmente o conteúdo desta propriedade é conposto por :

Nome do Aplicativo (Excel, WinWord, PBrush, etc.)
Pipe (|)
Nome do arquivo sobre o qual a ligação deve ser feita (*.xls, *.doc, *.bmp, etc.)

Exemplos:

1. Para fazer a ligação do Excel com o VB, sendo a aplicação VB o destino, a propriedade LinkTopic deverá ter o seguinte conteúdo: "Excel|arquivo.xls" (supondo que o arquivo Excel sobre o qual a ligação deve ser feita se chame 'arquivo.xls')

2. Para fazer a ligação do Word com o VB, sendo a aplicação VB o destino, a propriedade LinkTopic deverá ter o seguinte conteúdo: "WinWord|arquivo.doc" (supondo que o arquivo Word sobre o qual a ligação deve ser feita se chame 'arquivo.doc')

LinkItem ### Determina especificamente o dado passado na conversação. Em uma ligação DDE com o Excel pode ser a faixa de células, na ligação com o Word pode ser um indicador (bookmark) no texto, na ligação com outra aplicação Visual Basic pode ser o nome do controle que é fonte da ligação DDE.

LinkTimeout ### Determina o intervalo de tempo, em décimos de segundos, que a aplicação deverá gastar tentando estabelecer a ligação DDE com o outro aplicativo.

Antes de tentar qualquer ligação DDE, a propriedade LinkMode deverá receber o conteúdo 0. Desta forma, para utilizar o mecanismo DDE corretamente, as seguintes propriedades devem ser "setadas" na seguinte ordem :

1. LinkMode = 0
2. "Setar" o conteúdo de LinkTopic
3. "Setar" o conteúdo de LinkItem
4. "Setar" o conteúdo de LinkTimeout
5. "Setar" o conteúdo de LinkMode

Aplicação Fonte

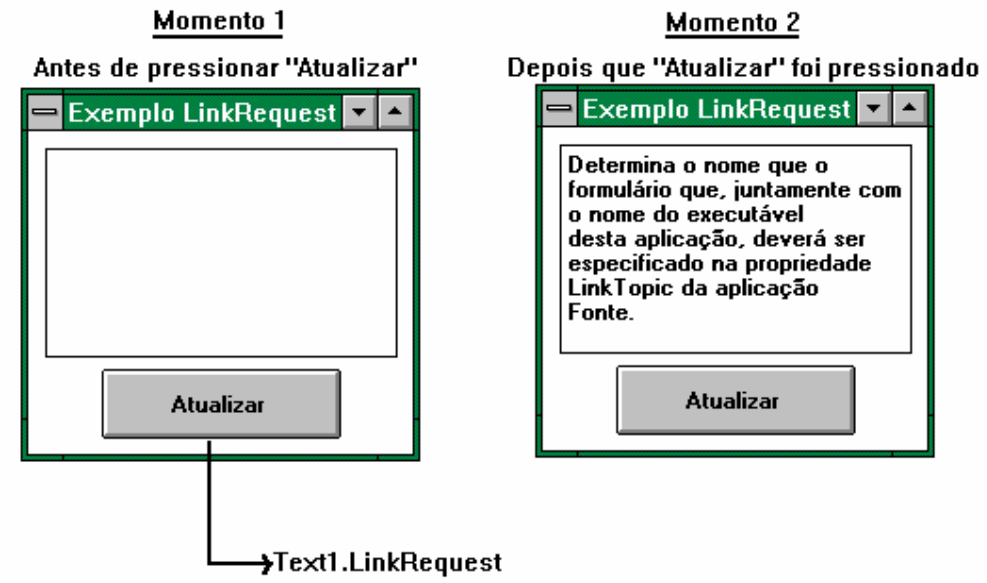
Quando uma aplicação VB atua como fonte de uma ligação DDE, as seguintes propriedades do formulário que contém o controle que será fonte da ligação DDE devem ter seus valores determinados : LinkMode, LinkTopic.

LinkMode ### Determina se um formulário VB pode atuar como Fonte em uma ligação DDE. A propriedade LinkMode do formulário pode assumir os seguintes valores :

Valor	Definição
0	Não é fonte de ligação DDE.
1	Pode ser fonte de ligação DDE.

LinkItem ### Determina o nome que o formulário, juntamente com o nome do executável desta aplicação, deverá ser especificado na propriedade LinkTopic da aplicação Fonte.

O Método LinkRequest

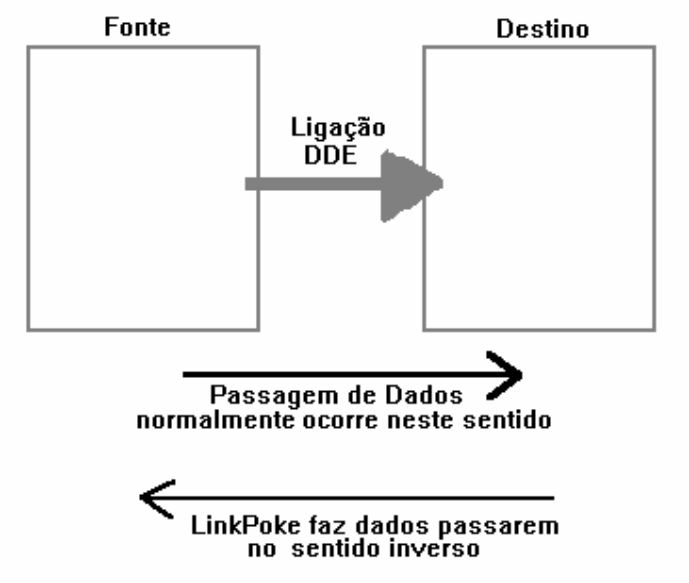


O método LinkRequest é utilizado em aplicações destino quando o tipo da ligação DDE é do tipo Manual ou do tipo Notificada. Ele requisita à aplicação fonte a atualização dos dados contidos na aplicação destino. Este método deverá ser aplicado sobre o controle destino da ligação DDE.

Sintaxe :

<nome-do-controle>.**LinkRequest**

Método LinkPoKe

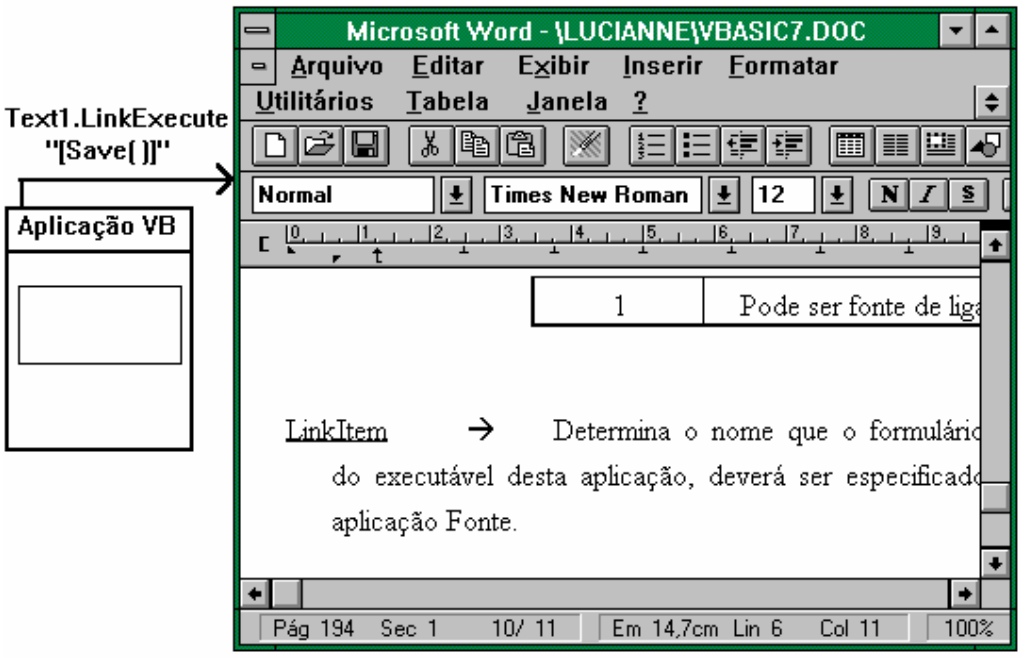


O método LinkPoke, aplicado sobre um controle de uma aplicação destino em uma ligação DDE, faz com que os dados fluam no sentido inverso da ligação, ou seja, os dados da aplicação fonte serão atualizados de acordo com o conteúdo da aplicação destino.

Sintaxe:

<Nome_controle>.**LinkPoke**

Método LinkExecute



O método LinkExecute, aplicado sobre o controle destino de uma aplicação, executa um comando qualquer na aplicação fonte. Importante observar que o comando enviado deverá ser reconhecido pela aplicação fonte.

Sintaxe:

<nome-controle>.**LinkExecute**

Método LinkSend

Quando o uma aplicação VB está no papel de fonte em uma ligação DDE, se a ligação é do tipo automática, o VB atualiza a aplicação destino quando há qualquer alteração na controle que é fonte da ligação. Isto ocorre quando a fonte da ligação é diferente de desenho.

Quando a fonte da ligação é um desenho, a aplicação deve explicitamente enviar as alterações para as aplicações-destino. Para isso, utiliza-se o comando LinkSend sobre o controle fonte da ligação.

Isto ocorre para gráficos porque o envio de um desenho de uma aplicação para outra pode ser dispendioso demais em relação a recursos de máquina e tempo. Este é o porquê da opção da Microsoft por "não manter ligação automática para figuras".

Eventos associados ao DDE

Durante a criação e existência de uma ligação DDE, alguns eventos eventualmente. São eles : LinkOpen, LinkClose, LinkError e o LinkNotify.

LinkOpen ### A aplicação VB destino recebe este evento sempre que uma ligação DDE é iniciada com sucesso. A aplicação VB fonte também recebe este evento, porém este evento tem um parâmetro do sistema (Cancel), que se "setado" para True pode cancelar a ligação.

LinkClose ### Ocorre tanto na aplicação fonte como na destino, quando uma das duas finaliza a ligação.

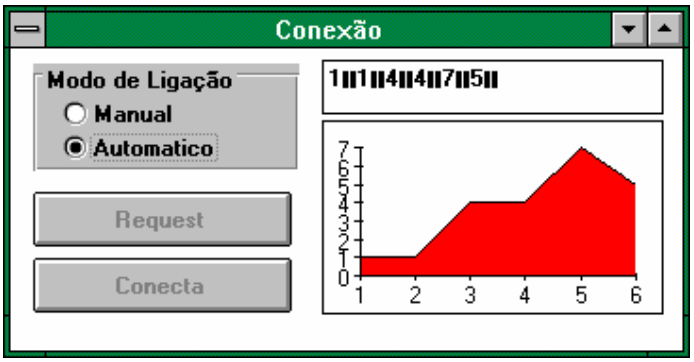
LinkError ### Ocorre na aplicação destino quando um erro acontece enquanto nenhum código está sendo executado (Por exemplo, pode faltar memória ao sistema). Erros ocorridos durante a execução de código, podem ser tratados pelo método normal de tratamento de erros (já aprendido).

LinkNotify ### Ocorre na aplicação destino, quando o conteúdo da ligação tiver sido alterado na aplicação fonte. Atenção , este evento somente ocorre se a propriedade LinkMode tiver valor 3 (Notify).

Exemplo - Criação de uma ligação em Run-Time onde Excel é fonte e VB é destino

Para exemplificar este tipo de ligação DDE, criaremos uma aplicação que mostrará o conteúdo de uma coluna de uma planilha Excel, e além disso, um gráfico dos

números apresentados. Esta ligação DDE poderá se dar de modo automático ou manual.



Criando a planilha no Excel

1. Abra o aplicativo Excel.
2. Digite seis números, um em cada linha da primeira coluna (até atingir a sexta linha).
3. Selecione os seis números e pressione o botão de criação de gráficos. Selecione então uma área na planilha onde você quiser que o gráfico apareça. Pressione "Próximo" (várias vezes) até que o gráfico esteja pronto, e então pressione OK.
4. Salve esta planilha com o nome de "source.xls".

Criando a Aplicação no Visual Basic

5. Crie uma PictureBox e nomeie-a *Pic*.
6. Crie uma TextBox e nomeie-a *TXT*.
7. Crie um Botão, desabilitado, com Caption = "Request" e Nome = "CMDRequest".
8. Crie outro botão, com caption = "Conecta" e Nome = CMDConecta.
9. Crie um Frame, desabilitado, com o nome "Frame".
10. Crie um Option Button, dentro do frame recém criado, com o nome "OPTManual", com Caption "Manual" e value *True*.
11. Crie outro Option Button dentro do Frame, desta vez com o nome "OPTAutomat" e caption "Automática".

Codificando o Botão "Conecta"

12. Associar ao evento Click deste botão as seguintes linhas de código :

```
Pic.LinkMode = 0
```

Para Excel em Português
```
Pic.LinkTopic = "Excel|source.xls Gráfico 1"
```
Para Excel em Inglês
```
Pic.LinkTopic = "Excel|source.xls Chart 1"

Pic.LinkItem = ""
Pic.LinkTimeOut = 50

Txt.LinkMode = 0
Txt.LinkTopic = "Excel|source.xls"
```

Para Excel em Português
```
Txt.LinkItem = "L1C1:L6C1"
```
Para Excel em Inglês
```
Txt.LinkItem = "R1C1:R6C1"

Txt.LinkTimeout = 50

If OPTAutomat.value = true then
   Pic.LinkMode = 1
   Txt.LinkMode = 1
Else
   Pic.LinkMode = 2
   Txt.LinkMode = 2
   CMDRequest.enabled = true
Endif

Frame.enabled = true
CMDConecta.enabled = false
```

Codificando o Option Button "Automático"

13. Associar ao evento Click deste botão as seguintes linhas de código :

```
CMDRequest.enabled = false
Pic.LinkMode = 0
Pic.LinkMode = 1
Txt.LinkMode = 0
Txt.LinkMode = 1
```

Codificando o Option Button "Manual"

14. Associar ao evento Click deste botão as seguintes linhas de código :

```
CMDRequest.enabled = true
Pic.LinkMode = 0
Pic.LinkMode = 2
Txt.LinkMode = 0
Txt.LinkMode = 2
```

Codificando o botão "Request"

15. Associar ao evento Click deste botão as seguintes linhas de código :

```
Txt.LinkRequest
Pic.LinkRequest
```

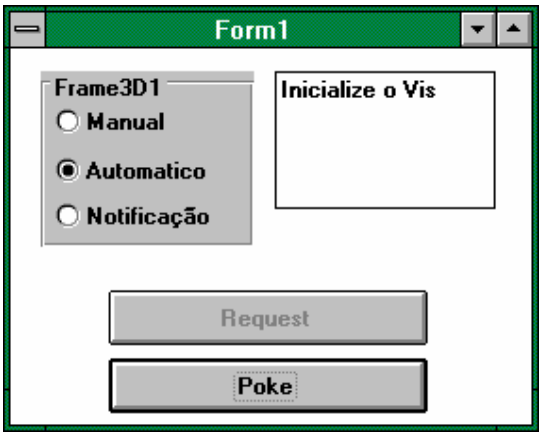
Executando a Aplicação

16. Pressione F5 para executar a aplicação. Lembre-se que o programa somente funcionará corretamente se o Excel estiver rodando, com a planilha "source.xls" aberta.

17. Experimente fazer alterações na planilha, e veja os resultados tanto quando a ligação DDE for de tipo automático, quanto quando a ligação for manual.

Exemplo - Criação de uma ligação em Run-Time onde Word é fonte e VB é destino

Para exemplificar este tipo de ligação DDE, criaremos uma aplicação que mostrará o conteúdo de um documento Word. Esta ligação DDE poderá se dar de modo automático ou manual. Você vai observar também, o que acontece quando tentamos criar uma ligação do tipo "Notificada", quando a aplicação fonte é o Word.



Criando o documento Word

1. Abra o aplicativo Word.
2. Digite dois parágrafos de um texto qualquer.
3. Selecione o texto inteiro e insira um indicador de nome "DDE_LINK1" (Menu *Inserir*, opção *Indicador* do Word).

4. Salve este documento com o nome de "source.doc".

Criando a Aplicação no Visual Basic

5. Crie uma TextBox e nomeie-a *TXT*.
6. Crie um botão com Caption = "Request" e Nome = "CMDRequest".
7. Crie outro botão, com caption = "Poke" e Nome = CMDPoke.
8. Crie um frame com o nome "Frame".
9. Crie um Option Button, dentro do frame recém criado, com o nome "OPTManual", e com Caption "Manual".
10. Crie outro Option Button dentro do Frame, desta vez com o nome "OPTAutomat" e caption "Automática".
11. Crie outro Option Button dentro do Frame, desta vez com o nome "OPTNotify" e caption "Notificação".

Codificando o Load do Formulário

12. Digite as seguintes linha de código :

```
dim z as integer
   Txt.LinkMode = 0
   Txt.LinkTopic = "WinWord|source.doc"
   Txt.LinkItem = "DDE_LINK1"
   Txt.LinkMode = 2
   OPTManual.value = true
```

Codificando o Option Button "Automático"

13. Associar ao evento Click deste botão as seguintes linhas de código :

```
CMDRequest.enabled = false
Txt.LinkMode = 0
Txt.LinkMode = 1
```

Codificando o Option Button "Manual"

14. Associar ao evento Click deste botão as seguintes linhas de código :

```
CMDRequest.enabled = true
Txt.LinkMode = 0
Txt.LinkMode = 2
```

Codificando o Option Button "Notificação"

15. Associar ao evento Click deste botão as seguintes linhas de código :

```
CMDRequest.enabled = true
Txt.LinkMode = 0
Txt.LinkMode = 3
```

Codificando o botão "Request"

16. Associar ao evento Click deste botão as seguintes linhas de código :

```
Txt.LinkRequest
```

Codificando o botão "Poke"

17. Associar ao evento Click deste botão as seguintes linhas de código :

```
Txt.LinkPoke
```

Executando a Aplicação

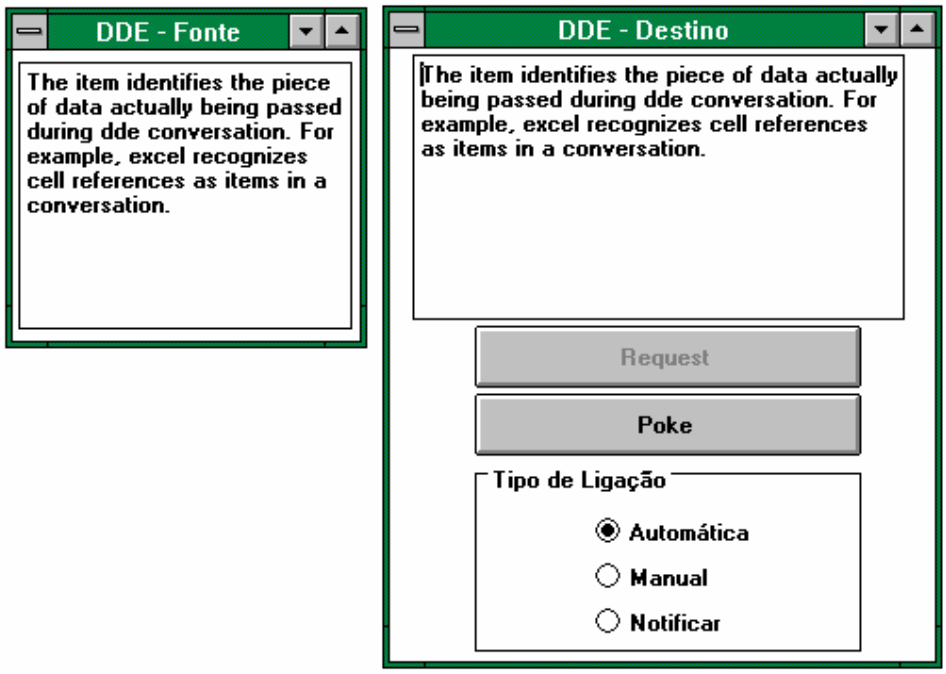
18. Pressione F5 para executar a aplicação.

19. Experimente fazer alterações no texto, e veja os resultados tanto quando a ligação DDE for de tipo automático, quanto quando a ligação for manual ou Notificada.

20. Verifique a Utilização do método LinkPoke.

Exemplo - Criação de uma ligação em Run-Time onde VB é fonte e VB é destino

Já vimos nos exemplos anteriores como o Visual Basic funciona como aplicação destino de uma ligação DDE, onde a fonte é ou o Excel ou o Word for Windows. Agora, veremos como funciona uma ligação DDE onde uma aplicação VB é fonte, e outra destino.



Criando a aplicação Visual Basic Fonte

1. Inicialize o Visual Basic.
2. Crie uma TextBox, com nome "TXT_Fonte".
3. Digite dois parágrafos de um texto qualquer na propriedade *Text* da TextBox recém criada..
4. Selecione formulário.
5. Em sua propriedade LinkMode, digite o valor 1 (Source) e em sua propriedade LinkTopic, a string "DDE_Form1".
6. Gere um executável de nome "DDE_FNT.exe".
7. Crie um grupo no Windows e Crie o item de programa "DDE_Fonte" associado ao executável recém criado.

Criando a aplicação Visual Basic Destino

8. Crie uma TextBox e nomeie-a *TXT*.
9. Crie um botão, desabilitado, com Caption = "Request" e Nome = "CMDRequest".
10. Crie outro botão, com caption = "Poke" e Nome = CMDPoke.
11. Crie um frame com o nome "Frame".
12. Crie um Option Button, dentro do frame recém criado, com o nome "OPTManual", e com Caption "Manual".

13. Crie outro Option Button dentro do Frame, de value True, desta vez com o nome "OPTAutomat" e caption "Automática".
14. Crie outro Option Button dentro do Frame, desta vez com o nome "OPTNotify" e caption "Notificar".

Codificando o Load do Formulário

15. Digite as seguintes linha de código :

```
dim z as integer
On Error goto T_Erro
   Txt.LinkMode = 0
   Txt.LinkTopic = "DDE_FNT|DDE_Form1"
   Txt.LinkItem = "TXT_Fonte"
   Txt.LinkMode = 1
Exit Sub
T_Erro:
   z = shell( "DDE_FNT.exe", 1)
   z = doevents()
   Resume
```

Codificando o Option Button "Automático"

16. Associar ao evento Click deste botão as seguintes linhas de código :

```
CMDRequest.enabled = false
Txt.LinkMode = 0
Txt.LinkMode = 1
```

Codificando o Option Button "Manual"

17. Associar ao evento Click deste botão as seguintes linhas de código :

```
CMDRequest.enabled = true
Txt.LinkMode = 0
Txt.LinkMode = 2
```

Codificando o Option Button "Notificação"

18. Associar ao evento Click deste botão as seguintes linhas de código :

```
CMDRequest.enabled = true
Txt.LinkMode = 0
Txt.LinkMode = 3
```

Codificando o botão "Request"

19. Associar ao evento Click deste botão as seguintes linhas de código :

```
Txt.LinkRequest
```

Codificando o botão "Poke"

20. Associar ao evento Click deste botão as seguintes linhas de código :

```
Txt.LinkPoke
```

Codificando o Evento LinkNotify

21. Digitar a seguinte linha de código :

```
MSGBOX "Conteúdo da Aplicação Fonte foi alterado !!!"
```

Executando a Aplicação

22. Pressione F5 para executar a aplicação.

23. Experimente fazer alterações no texto na aplicação Fonte, e ver os resultados tanto quando a ligação DDE for de tipo automático, quanto quando a ligação for manual ou Notificada.

24. Verifique a Utilização do método LinkPoke.

Esta página foi deixada em branco intencionalmente.

OLE - Object Linking and Embedding

Objetivo

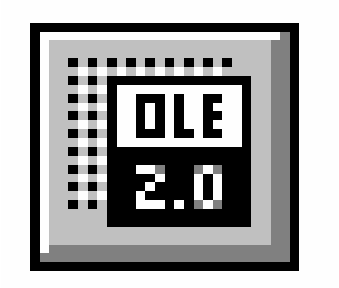
OLE é mais uma forma de "interoperabilidade" entre os aplicativos Windows. O OLE permite que um usuário, de dentro de uma aplicação Visual Basic, edite objetos criados a partir de outros aplicativos.

Conceituação de OLE

OLE é mais um método de compartilhamento de dados entre aplicações Windows. OLE permite a criação de aplicações VB, nas quais o usuário poderá acessar as melhores características de todas as ferramentas Windows. Por Exemplo, uma aplicação VB poderia conter um texto do Word, uma planilha do Excel, etc. Desta forma a aplicação está centrada nos objetos que possui e não na ferramenta (o próprio VB).

Dentro do objeto OLE, você pode, por exemplo, colocar uma planilha Excel. O usuário irá visualizar esta planilha, da mesma forma que ele a vê no Excel. Ao dar duplo-clique com o mouse sobre aquele objeto, o usuário poderá editar a planilha dentro do próprio Excel.

O Controle OLE



O controle do Visual Basic OLE, uma vez colocado no formulário permite que você associe aquela posição de tela a um aplicativo qualquer do Windows. O item ao qual o OLE é associado, é denominado objeto. Um objeto pode ser uma planilha em Excel, um documento em Word, uma figura do PaintBrush, etc..

Ao inserir um controle OLE no formulário, uma caixa de diálogo é apresentada. Nesta caixa, o usuário seleciona o aplicativo ao qual ele deseja associar o OLE, se ele deseja incorporar um objeto ou vincular o OLE a um objeto externo, etc..

Diferenças entre Ligação e Incorporação (Linking and Embedding)

O controle OLE é utilizado para associar um objeto a uma aplicação Visual Basic. Esta associação pode ser feita de duas maneiras : Ligando o objeto ao controle OLE da aplicação Visual Basic ou incorporando o objeto na aplicação.

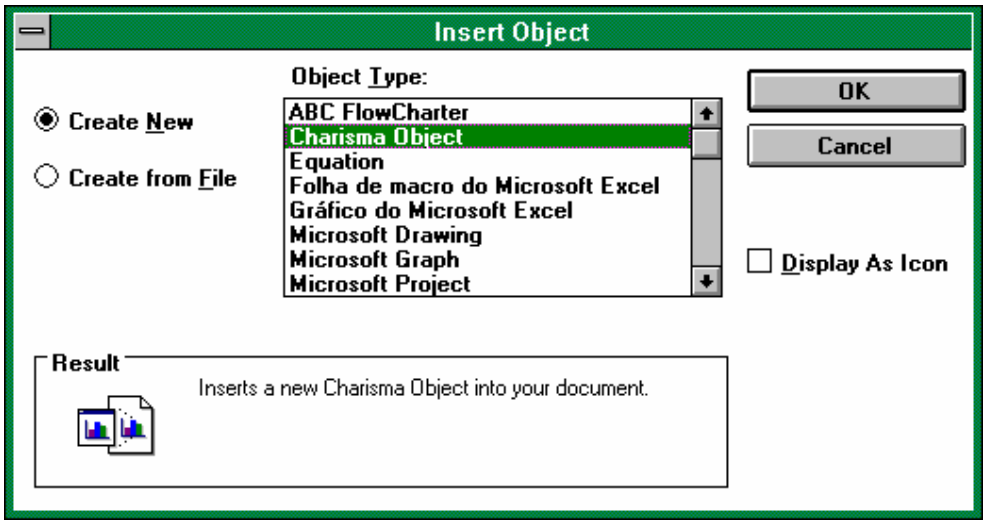
Ligação

Na ligação, o objeto não fica guardado dentro da aplicação VB. Nela, há apenas um ponteiro direcionando-a para o objeto externo. Este objeto, além de ser acessado pela sua aplicação, também pode ser acessado por qualquer outro aplicativo Windows. Por exemplo, se você tiver uma planilha Excel ligada à sua aplicação VB, esta mesma planilha também poderá estar ligada a um documento Word. Isto acontece porque o VB não guarda dentro de si a planilha, mas apenas um ponteiro para ela.

Incorporação

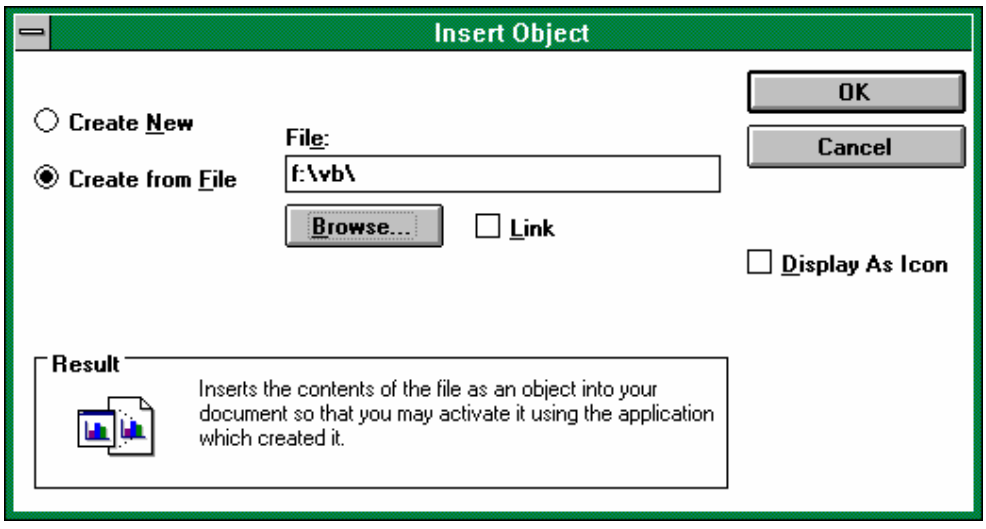
Na Incorporação, o dado fica guardado dentro da aplicação VB. Este dado somente pode ser acessado pela ferramenta que o gerou, e assim mesmo, quando chamado de dentro da aplicação VB que possui o dado.

Criando um Objeto em Design Time



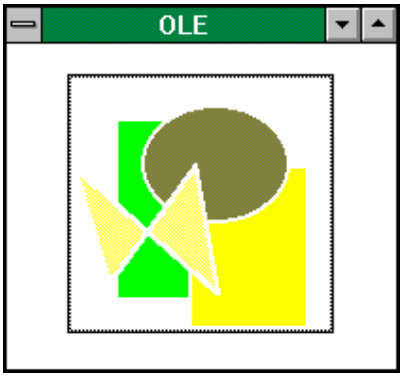
Ao inserir um controle OLE dentro de um formulário, uma caixa de diálogo é apresentada. Nesta caixa, o usuário pode selecionar : a ferramenta que gerou o objeto; se ele deseja criar o objeto a partir de um arquivo já existente ou um novo arquivo; se ele deseja incorporar o objeto ou fazer apenas uma ligação.

Importante observar que, para objetos incorporados, se o programador decidir criar a partir de um arquivo já existente, o OLE estará apenas pegando o arquivo existente como modelo, e fazendo uma cópia dos dados para dentro da aplicação VB.



Exemplo - Criando um OLE em Design-Time

Para exemplificar a criação de um controle OLE em design-time, desenvolveremos uma aplicação que conterá apenas um controle OLE cujo objeto será uma figura em PaintBrush, incorporada à aplicação Visual Basic.



Criando o controle OLE na Aplicação VB

1. Inicialize o Visual Basic.
2. Crie no formulário aberto, um controle OLE. Uma janela de Dialogo aparecerá.
3. Na janela de Dialogo, selecione o aplicativo PaintBrush.
4. Marque a opção *Create New*. Pressione o botão OK. O PaintBrush será aberto.

Criando a Figura na PaintBrush

5. No PaintBrush, desenhe uma figura qualquer.
6. Do PaintBrush, atualize a figura incorporada e saia do aplicativo.

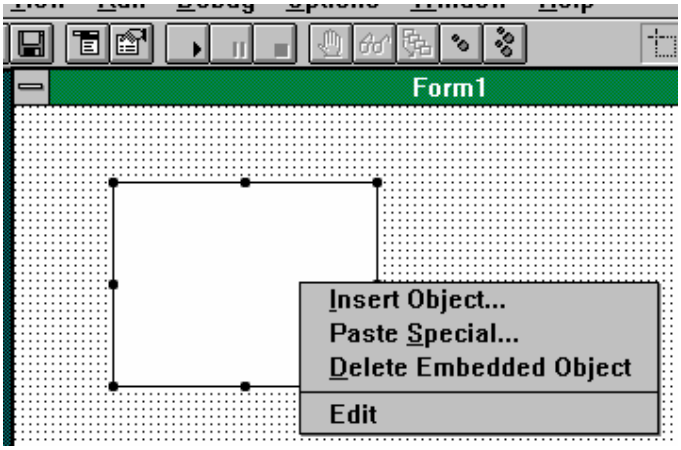
Executando a Aplicação

7. Pressione F5 para executar a aplicação.
8. Dê duplo-clique sobre a figura PaintBrush.
9. Faça alterações na figura e veja como elas refletem no controle OLE.
10. Pare a execução da aplicação e inicialize-a de novo. Veja como as alterações na figuras não foram armazenadas.

Se Houver Tempo ...

11. Crie um objeto "ligado" (não incorporado). Faça alterações nele e veja como ele reage.

Os menus Pop_Up do Controle OLE



Em design-time, o programador pode acessar comandos da OLE mais facilmente se clicar o botão direito do mouse sobre o controle OLE. Neste momento um menu é apresentado com quatro opções :

Insert Object - Mostra a Caixa de Diálogo para associação do controle OLE com um aplicativo Windows.

Paste Special - Permite a criação de um OLE, a partir do conteúdo do Clipboard.

Delete Embedded Object - Desfaz qualquer associação daquele controle com qualquer aplicativo Windows.

Edit - Abre o aplicativo ao qual o controle está associado para edição.

Propriedades para Criação de controles OLE em Run-Time

Quando se deseja criar um controle OLE em design-time, deve-se usar a caixa de diálogo "Insert Object". Para criá-lo em run-time, deve-se determinar os valores de algumas propriedades em código. Elas são : Class, OleTypeAllowed, SourceDoc, SourceItem e Action.

Class ### Identifica o aplicativo que criou o objeto. Ex.: PaintBrush.

OleTypeAllowed ### Indentifica o tipo de objeto que poderá ser criado (ligado, incorporado ou ambos).

SourceDoc ### Em objetos ligados, determina o arquivo ao qual a OLE estará ligada. Ex.: Figura.bmp.

Em objetos Incorporados, determina um arquivo para ser tomado como modelo.

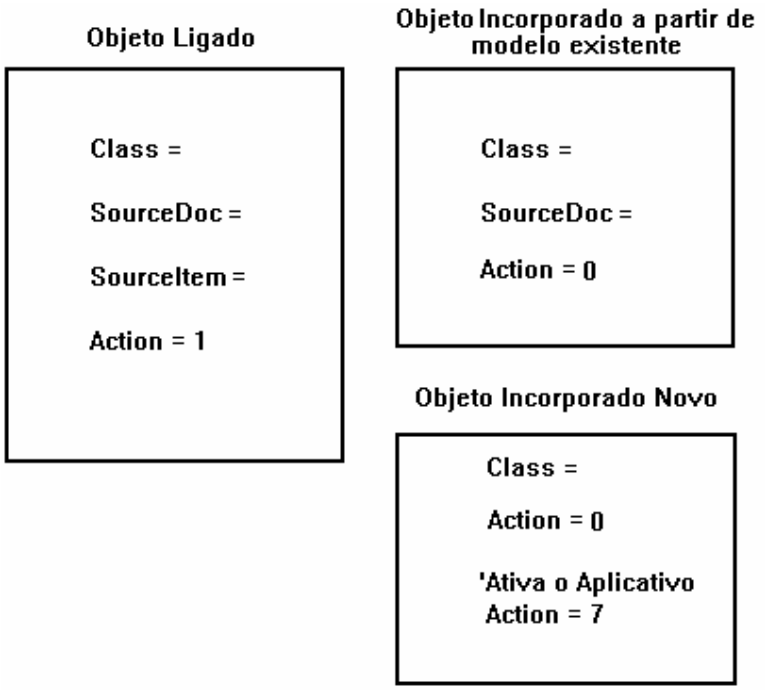
SourceItem ### Especifica o dado dentro de um arquivo ao qual o objeto estará ligado (serve apenas para Ligações). Ex.: L1C1 (linha 1 coluna 1 de uma planilha Excel).

Action ### Especifica o tipo de operação a ser executada no controle OLE.

Propriedade Action	Significado
0	Cria um objeto incorporado
1	Cria um objeto Ligado
4	Copia o objeto para o ClipBoard
5	Copia objeto do ClipBoard
6	Atualiza objeto a partir do aplicativo que o criou.
7	Abre o aplicativo gerador do objeto.
9	Fecha o objeto OLE e termina a ligação com o aplicativo que criou o objeto.
10	Deleta o objeto OLE e desocupa memória.
11	Salva conteúdo do objeto OLE para um arquivo.
12	Lê conteúdo de um objeto OLE de um arquivo.
14	Mostra caixa de diálogo "Insert Object".
15	Mostra caixa de diálogo Paste Special.
18	Salva um objeto incorporado para um arquivo no formato OLE.

Criando Controles OLE Ligados em Run-Time

Antes de criar um objeto OLE Ligado, em Run-Time, o programador precisa definir o aplicativo a ser ligado, o nome do arquivo e opcionalmente a área do arquivo.



Criando Controles OLE Incorporados em Run-Time

Para criar um objeto OLE, incorporado, em Run-Time, a partir de um modelo já existente, o programador deverá seguir os seguintes passos :

1. Setar a propriedade Class.
2. Setar a propriedade SourceDoc.
3. Setar a propriedade Action = 0.

Para criar um novo objeto OLE, incorporado, em Run-Time, o programador deverá seguir os seguintes passos :

1. Setar a propriedade Class.
2. Setar a propriedade Action = 0, para criar o objeto.
3. Setar a propriedade Action = 7, para ativar o aplicativo para acriação do objeto.

Trabalhando com Arquivos

Dados associados a uma objeto OLE não são permanentes, isto é, quando um formulário contendo um controle OLE é fechado, todos os dados associados àquele controle são perdidos. É possível salvar dados de um objeto OLE para um arquivo através da propriedade Action. Uma vez salvos os dados, eles podem ser lidos de volta para o controle OLE a qualquer momento.

Objetos OLE podem ser salvos apenas para arquivos binários abertos.

Salvando dados em um arquivo

1. Abra um arquivo em formato binário.
2. Ajuste a propriedade FileNumber.
3. Ponha o valor 11 na propriedade Action.

Veja o exemplo abaixo, este código poderia ser associado ao click de um botão :

```
Dim NumArquivo as integer
   NumArquivo = FreeFile
   Open "arquivo.OLE" for Binary as #NumArquivo
   Ole1.FileNumber = NumArquivo
   Ole1.Action = 11
   Close #NumArquivo
```

Lendo dados de um arquivo

1. Abra um arquivo em formato binário.
2. Ajuste a propriedade FileNumber.
3. Ponha o valor 12 na propriedade Action.

Veja o exemplo abaixo, este código poderia ser associado ao click de um botão :

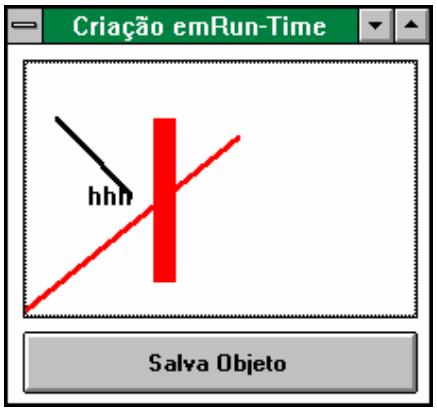
```
Dim NumArquivo as integer
   NumArquivo = FreeFile
   Open "arquivo.OLE" for Binary as #NumArquivo
   Ole1.FileNumber = NumArquivo
   Ole1.Action = 12
   Close #NumArquivo
```

Evento Updated

O evento Updated é invocado sempre que o conteúdo de um objeto é alterado. Este evento tem como parâmetro um código que indica se o contúdo do objeto foi alterado, salvo, ou se o aplicativo foi fechado.

Exemplo - Criação de um objeto OLE em run-time

Para exemplificar a criação de um objeto OLE em run-time, desenvolveremos uma aplicação contendo dois controles, um OLE e um botão. Ao inicializar a aplicação, ela verificará se já existe um arquivo de nome "FOO.ole". Em caso positivo, ela lerá o conteúdo deste arquivo para o controle OLE. Em caso negativo, o PaintBrush será aberto para que o usuário possa desenhar alguma coisa. O usuário poderá salvar suas alterações em arquivo sempre que quiser, bastando para isso pressionar o botão "Salva Objeto".



Criando os Controles no Visual Basic

1. Inicialize o Visual Basic.

2. Crie um controle OLE. Pressionar o botão de "Cancel"na Caixa de Diálogo.

3. Crie um botão com caption "Salva Objeto".

Programando o Load do Formulário

4. Digitar as seguintes linhas de código:

```
Dim numarquivo As Integer

    ' Testa se o arquivo foo.ole já existe, se existir abre, senão
    ' abre o PaintBrush
    If Trim(Dir("c:\curso\foo.ole")) <> "" Then

        ole1.Class = "PBrush"
        ole1.Action = 0

       ' A função FreeFile indica o próximo número de arquivo livre.
        numarquivo = FreeFile
        Open "c:\curso\FOO.ole" For Binary As #numarquivo
        ole1.FileNumber = numarquivo
        ole1.Action = 12
        Close #numarquivo
    Else
        ole1.Class = "PBrush"
        ole1.Action = 0
        ole1.Action = 7
    End If
```

Codificando o click do botão "Salva Objeto"

5. Digitar as seguintes linhas de código :

```
Dim numarquivo As Integer

    numarquivo = FreeFile
    Open "c:\curso\FOO.ole" For Binary As #numarquivo
    ole1.FileNumber = numarquivo
    ole1.Action = 11
    Close #numarquivo
```

Executando a Aplicação

6. Pressione F5 para inicializar a aplicação.
7. Faça alterações na figura e salve-a pelo menos uma vez.
8. Finalize a aplicação e inicialize-a novamente. Veja coma a figura está atualizada.
9. Rode a aplicação em Debug Mode, para ver o que acontece passo a passo.

Acessando DLL (Dynamic Link Library)

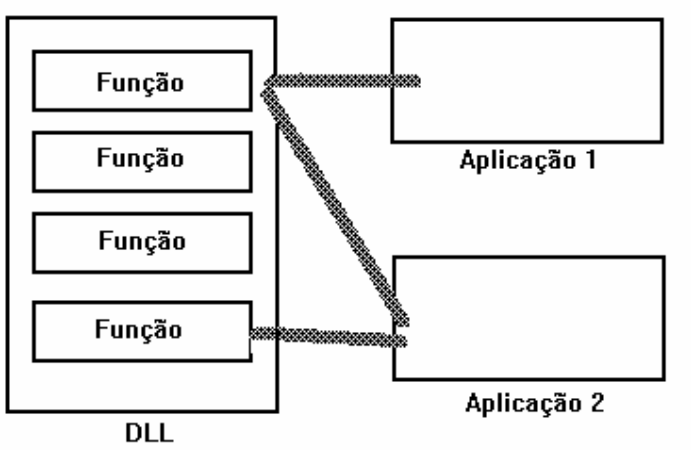
Objetivo

Apresentar ao participante as bibliotecas de funções dinâmicas do Windows. Neste capítulo, além de verificar o funcionamento deste tipo de bibliotecas, o participante irá conhecer algumas funções úteis,

Conceituação de DLL

Dynamic Link Libraries (DLLs), como o próprio nome sugere, são bibliotecas de funções que podem ser dinamicamente "ligadas" a aplicações em run-time, ao invés de associadas estaticamente em design-time.

Por serem apenas ligadas dinamicamente em run-time, as funções contidas nestas bibliotecas podem ser compartilhadas por várias aplicações diferentes, e quando atualizadas (pelo fabricante da DLL em questão), o processo é feito independentemente das aplicações as quais ela está ligada.



Uma aplicação VB pode não só chamar DLLs do Windows (GDI, Kernel, User), como também bibliotecas como o VBRun300.DLL, bibliotecas de outros aplicativos, e bibliotecas próprias, escritas em C.

Informações sobre as funções contidas em cada biblioteca, podem ser encontradas na documentação do aplicativo que trouxe a biblioteca.

Existe Help on-line para as bibliotecas do Windows (podem ser encontrados como ícones no grupo de programas do Visual Basic 3.0) :

Win 3.1 API Help ### Contém a lista de todas as funçoes existentes nas bibliotecas do Windows. Neste Help, há também a declaração para cada uma destas funções.

Win SDK Help ### Contém a descrição de cada uma das funções.

Como utilizar uma Biblioteca DLL

Para utilizar uma biblioteca DLL, o programador deve seguir os seguintes passos :

1. Escrever o comando de declaração da função desejada.
2. Fazer a chamada à função no meio do código.
3. Salvar a aplicação e testar o funcionamento da função.

Declarando uma DLL

Para fazer a declaração de uma biblioteca DLL, o programador se utiliza o comando *Declare*, que deve ser colocado dentro do *General Declarations* de um formulário ou módulo de código.

Os parâmetros necessários para a declaração da função devem ser procurados na documentação da DLL.

Exemplo :

```
Declare Sub InvertRect Lib "User" (ByVal hDC as
   integer, aRect as Rectangle)

Declare Function GetSystemMetrics Lib "User" (ByVal n   as
Integer) as Integer
```

Chamando uma função de uma DLL

Uma vez declarada a função ou rotina, você pode chamá-la de seu código como se fosse uma outra subrotina qualquer.

Tenha cuidado, pois se algum parâmetro incorreto for passado, a sua aplicação pode abortar. Salve seu trabalho com freqüencia !!!

Exemplo :

```
Sub Form_Load ()
Const SM_MOUSEPRESENT = 19
   show
   If GetSystemMetrics(SM_MOUSEPRESENT) Then Print "Mouse
   Installed"
End Sub
```

Tipos de Dados

As funções contidas em bibliotecas DLL normalmente são escritas em C. Existem alguns tipos de dados que existem em C que não existem em VB, ou vice-versa. Quando passamos parâmetros para funções de DLLs, precisamos nos certificar que estamos passando o valor correto. Veja a tabela abaixo, que contém os tipos de dados correspondentes em C e em VB :

Tamanho	C	Visual Basic
1 byte	char	leia texto abaixo
2 byte	int	integer
4bytes	long	long
4bytes	float	single
8 bytes	double	double
8 bytes	não existe	currency

O Visual Basic não contém um tipo de dados *Char*. Quando uma função de uma DLL exigir como parâmetro este tipo de dado, o código ASCII do caracter deverá ser passado numa variável inteiro.

A linguagem C não suporta o tipo currency. Quando um valor deste tipo tiver que ser passado, converta-o antes (dentro do VB), para o tipo de dado Double.

Qualquer função de DLL's pode ser chamada de dentro do Visual Basic, exceto aquelas que recebem como parâmetros ponteiros.

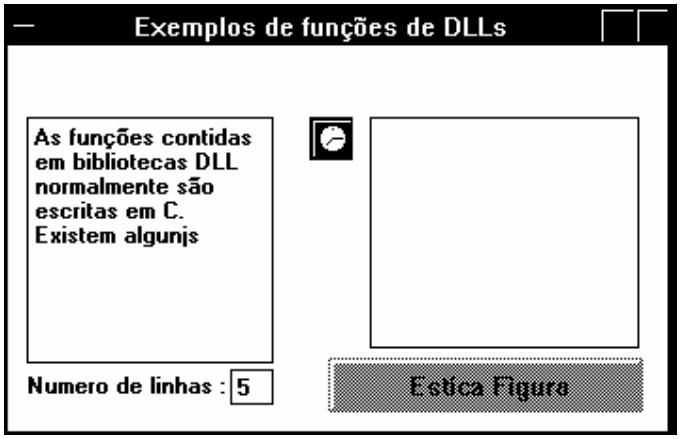
Handles de Controles

Funções de DLLs fazem uso extensivo de *handles* de controles. Um handle é um valor inteiro definido pelo sistema operacional, que serve para referenciar objetos como controles e formulários.

A propriedade **hWnd** de controles e formulários, disponível apenas em run-time, retorna o valor do handle para aquele formulário ou controle especificamente.

Exemplo - Utilização de Funções de Bibliotecas DLL

Para exemplificar o uso de funções de bibliotecas DLL, criaremos uma aplicação que usará três funções : Uma que faz o título da janela piscar, outra que permite a contagem do número de linhas dentro de um TextBox,e outra que estica um bitmap.



Criando os controles

1. Inicialize o Visual Basic.

2. No formulário existente, crie os seguintes controles :

Controle	Name	Caption
TextBox	TXT1	
TextBox	TXT2	
Picture	Pic1	
Picture	Pic2	
Command Button	cmd1	Estica Figura
Label		Número de Linhas

3. Associe a Pic1, um bitmap qualquer do diretório \VB\Bitmaps.

4. Crie um timer, habilitado, com intervalo de 500 milésimos de segundos.

Programando os eventos

5. Na seção General Declarations, escreva as seguintes linhas de código ou copie as funções do Help Win API.

```
Declare Function flashwindow Lib "User" (ByVal hWnd As Integer, ByVal
   binvert As Integer) As Integer

Declare Function sendmessage Lib "User" (ByVal hWnd As Integer, ByVal
   wmsg As Integer, ByVal wparam As Integer, lparam As Any) As Long

Declare Function StretchBlt% Lib "GDI" (ByVal hDC%, ByVal X%, ByVal
   Y%, ByVal nWidth%, ByVal nHeight%, ByVal hSrcDC%, ByVal XSrc%,
   ByVal YSrc%, ByVal nSrcWidth%, ByVal nSrcHeight%, ByVal dwRop&)
```

6. No evento timer do timer, escreva as seguintes linhas de código :

```
Dim X As Integer

    X = flashwindow(form1.hWnd, 1)
```

7. No evento Change de TXT1, escreva as seguintes linhas de código :

```
Const GetLineCount = &H40A
    Txt2.Text = sendmessage(txt1.hWnd, GetLineCount, 0, ByVal 0&)
```

8. No click do botão "Estica Figura", escrever as seguintes linhas de código :

```
Const pixel = 3
Const srccopy = &HCC0020
Dim rc

    pic1.ScaleMode = 3
    pic2.ScaleMode = 3

    rc = StretchBlt(picture2.hDC, 0, 0, pic2.ScaleWidth,
      pic2.ScaleHeight,    pic1.hDC,    0,    0,    pic1.ScaleWidth,
      pic1.ScaleHeight, srccopy)
```

Executando a Aplicação

9. Pressione F5. Experimente escrever texto na TXT1, e pressione o botão "Esticar Figura". Entre no Help on-line para conhecer novas funções.