Capítulo 1

O Que é o Delphi

Diferenças entre Delphi Client/Server do Delphi for Windows
Características que compõem o Integrated Development Invironment (IDE)
Como o Delphi se encaixa na família Borland

Overview

O Delphi oferece uma nova maneira de desenvolver aplicações Client/Server para MS Windows. Ele combina a facilidade de utilização de um ambiente de desenvolvimento visual, o poder de uma linguagem compilada orientada a objetos e uma tecnologia de ponta em banco de dados. Este capítulo o introduz as várias características do Delphi e da família de produtos Borland.

Introdução ao Delphi

Introdução

O Delphi é um ambiente de desenvolvimento de aplicações, orientado a objeto, que permite o desenvolvimento de poderosas aplicações baseadas no MS Windows com o mínimo de codificação. O Delphi também oferece ferramentas de desenvolvimento, tais como templates de aplicações e forms, que lhe permitem criar e testar rápidamente o protótipo de suas aplicações. Você pode utilizar o conjunto de componentes e código gerado para transformar seus protótipos em aplicações robustas que satisfaçam suas necessidades. O Delphi também oferece ferramentas de bancos de dados que lhe permitem desenvolver aplicações Client/Server e relatórios. As ferramentas de bancos de dados permitem que você visualize seus dados dinamicamente durante o desenvolvimento para que verifique imediatamente se os resultados de suas queries estão de acordo com suas necessidades.

Edições do Software

O Software Delphi vêm em duas edições:

Delphi for Windows

O Delphi for Windows Edition é voltado aos desenvolvedores de aplicações desktop que estão interessados em desenvolver aplicações desktop de alta performance que acessem dados armazenados nos formatos dBase e Paradox. O Delphi for Windows inclui:

- Servidor InterBase Local para Windows 3.1
- ReportSmith

Delphi Client/Server

O Delphi Client/Server Editon é voltado aos desenvolvedores corporativos de aplicações Client/Server interessados em desenvolver aplicações Workgroup de alta performance.

Características do Delphi

O Delphi consiste de vários elementos, ferramentas de design e de banco de dados para auxiliá-lo a desenvolver e testar suas aplicações de forma rápida e intuitiva. A seguir descrevemos as características do Delphi:

Construtor Visual de Interface com o Usuário

O IDE permite criar visualmente aplicações Client/Server de forma rápida através da seleção de componentes na paleta.

Arquitetura Baseada em Componentes

Os componentes asseguram que as aplicações Delphi sejam robustas, reutilizáveis e de fácil manutenção.

Compilador de Código Nativo de Alta Performance

O Compilador gera código otimizado de máquina ao invés de p-code interpretado e lento, fazendo com que as aplicações sejam até vinte vezes mais rápidas.

Tow-Way Tools

A capacidade de alternar entre um form e seu código permite aos desenvolvedores trabalhar tanto na edição de texto como no modo de design visual através de total sincronização do código fonte com a representação visual.

Biblioteca de Componentes Visuais

A biblioteca de componentes visuais (VCL-Visual Component Library) consiste de objetos reutilizáveis incluindo objetos padrão de interface com o usuário, gerenciamento de dados, gráficos e multimidia, gerenciamento de arquivos e quadros de dialogo padrão. A Client/Server edition inclui o código fonte do Visual Component Library.

Arquitetura Aberta

A arquitetura do IDE permite adicionar componentes e ferramentas personalizadas e de terceiros.

Linguagem Orientada a Objetos

O Delphi utiliza o *Object Pascal*, que oferece a facilidade de programação em 4GL de alto nível com a performance e poderio de um 3GL.

Suporte à Tecnologia do Windows

O Delphi é compatível com a tendência da tecnologia Windows, incluindo suporte a OLE 2.0,DDE,VBXs e ODBC.

Deputador Gráfico

O Debugger permite encontrar e eliminar "bugs" em seu código.

Edição Estilo Brief

O Editor permite a utilização de um conjunto de símbolos para expressões. Consulte Brief Regular Expressions no Help on-line.

Ambiente Personalizável

A opção de menu Environment Options permite personalizar seu ambiente para o máximo de produtividade.

Object Browser
O Object Browser permite a visualização da hierarquia dos objetos na visual component library.

Gerenciador de Projetos
O Project Manager oferece uma visualização de todos os forms e units de um determinado projeto e oferece um mecanismo conveniente para gerenciar projetos.

Experts
Uma variedade de Experts o guiam através do desenvolvimento de tipos padrões de forms. Por exemplo, o Database form expert auxilia-o na construção de forms que exibam dados em bancos de dados locais ou remotos.

Gerador de Relatórios
O *ReportSmith*™ oferece a mais avançada ferramenta de geração de relatórios para desenvolvedores que precisem criar relatórios que acessem grandes volumes de dados.

Servidor Local Baseado em SQL
O Local InterBase Server permite desenvolvimento off-line econômico com um engine SQL de alta performance compatível com ANSI 92 que oferece acessibilidade a outros servidores, incluindo Oracle, Sybase, Informix, e InterBase em outras plataformas.

Características do Delphi Client/Server

O Delphi Client/Server Edition inclui todas as características do Delphi for e as seguintes características específicas ao ambiente Client/ Server:

Conectividade de Alta Performance
Os SQL Links oferecem acesso de alta performance aos drives nativos, conectando com bancos de dados populares, incluindo Oracle, Sybase,Informix, e InterBase.

Suporte a Team Development
O Intersolv PVCS permite que grupos de desenvolvedores trabalhem juntos com códigos fonte integrados, check-in, check-out e gerenciamento de controle de versão.

Esta característica requer Intersolv PVCS 5.1 ou posterior.

Construtor Visual de Query
O Visual Query Builder oferece uma ferramenta visual para criar facilmente queries sofisticadas e gerar automaticamente o código SQL correspondente.

Como o Delphi se Encaixa na Família Borland

Introdução

Os produtos Client/Server da Borland compartilham de um design e tecnologia em comum. Esta caracteristica da aos produtos Borland consistência e funcionalidade, mas ao mesmo tempo, oferece flexibilidade ao desenvolvedor para escolher a linguagem de desenvolvimento e fonte de dados. No nível desktop e LAN, os bancos de dados, linguagens e ferramentas Borland compartilham o mesmo Borland Database Engine (BDE), que podem conectar com e integrar quaisquer tipos de dados em sua organização. Estes tipos de dados incluem:

- Bancos de dados PC tais como dBase e Paradox
- Bancos de dados workgroup tais como Oracle, Sybase, Informix, e InterBase
- Bancos de dados acessados através de drives ODBC

Você pode misturar componentes Borland para construir o ambiente Client/Server que melhor atenda suas necessidades.

Componentes Chave da Família Borland

A Borland fornece uma linha completa de soluções Client/Server nas seguintes famílias de componentes chave:

Upsizing Cients

Upsizing é o processo de escalar aplicações de banco de dados baseados em PC para a arquitetura Client/Server. Como aplicações de banco de dados baseados em PC e aplicações de banco de dados Client/Server baseados em SQL cresceram em diferentes mercados, o Upsizing Client Tools satisfaz os requisitos de ambos os mercados. Através dos SQL Links, aplicações dBase e Paradox podem continuar a utilizar as ferramentas já famíliares enquanto aproveitam a vantagem de servidores de banco de dados Workgroup.

Ferramentas de Desenvolvimento de Última Geração

As ferramentas de desenvolvimento de ultima geração como Delphi e Borland C++, permitem criar aplicações Client/Server escaláveis do desktop ao nível enterprise. Estas ferramentas permitem rápida prototipação, desenvolvimento e alta performance. Estas ferramentas são orientadas a objeto, permitem reutilização de código e componentes, e também suportam o team development.

Servidores de banco de dados

A estratégia Client/Server da Borland inclui versões do InterBase Workgroup Server para MS Windows NT e Novell NetWare, bem como versões UNIX. O InterBase é um líder em tecnologia no mercado de bancos de dados relacionais e esta de acordo com o SQL 92, suportando diversas extensões da linguagem SQL.

Delphi Companion Products

Os seguintes produtos estão disponíveis para o Delphi:

- BRIEF 3.1*
- Delphi/Link for Notes
- ForeHelp*
- Turbo Assembler*
- VB Conversion Assistant

- CL Source Code
- Visual Solutions Pack*

* Também trabalha com Borland C++

Resumo do Capítulo

Pontos Chave

Após completar este capítulo, você aprendeu que:

- o Delphi é um ambiente de desenvolvimento de aplicações baseado em componentes que permitem desenvolver poderosas aplicações baseadas em MS Windows.
- dentre as características do Delphi incluem ferramentas de bancos de dados que permitem o desenvolvimento de aplicações e relatórios de bancos de dados.
- o software Delphi vêm em duas edições:
    - * Delphi for Windows
    - * Delphi Client/Server
- a Borland oferece soluções Client/Server nas seguintes famílias de componentes chave:
    - * Upsizing clients
    - *Ferramentas de desenvolvimento de ultima geração
    - *Servidores de bancos de dados
    - * Companion products

Termos e Definições

A tabela a seguir é uma referência rápida aos termos apresentados neste capítulo:

Termo	Descrição
DBE	Borland Database Engine, que pode conectar com e integrar quaisquer tipos de dados em uma organização, incluindo: • Bancos de dados PC tais como dBase e Paradox • Bancos de dados workgroup tais como Oracle, Informix, Sybase, e InterBase • Bancos de dados acessáveis através de drives ODBC
IDE	Integrated Development Environment (Ambiente de Desenvolvimento Integrado), um conjunto de elementos, ferramentas de design e de bancos de dados que auxiliam a desenvolver e testar rápida e intuitivamente em uma interface com o usuário

Capítulo 2

Um Tour pelo Ambiente de Programação do Delphi

Ao final deste capítulo, você estará apto a:

Identificar os elementos do IDE do Delphi

Identificar os elementos essenciais que formam uma aplicação

Overview

O Delphi oferece um ambiente integrado de deselvolvimento que permite desenvolver sofisticadas aplicações Windows com um mínimo de codificação. Este capítulo introduz o Intergrated Development Environment (IDE) e descreve os elementos essenciais do ambiente de programação.

Elementos do IDE do Delphi

Introdução

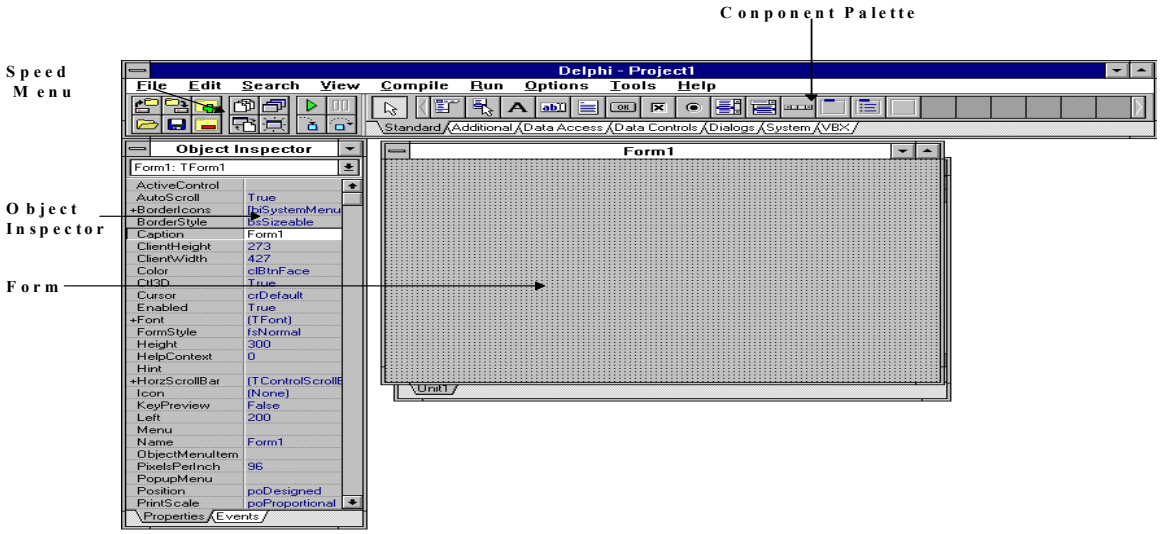
O Integrated Development Environment (IDE) do Delphi consiste de diversos elementos ou ferramentas visuais. Uma vez acostumados a estes elementos, você pode começar a desenvolver aplicações. Este capítulo discute os seguintes elementos para famíliariza-lo ao ambiente de desenvolvimento:

- Form
- Component Palette
- Object Inspector
- Code Editor
- SpeedBar
- ProjectManager
- SpeedMenus
- On-Line Help

As ferramentas são apresentadas na ordem em que seriam utilizadas para desenvolver uma aplicação.

Aparência Inicial da Interface

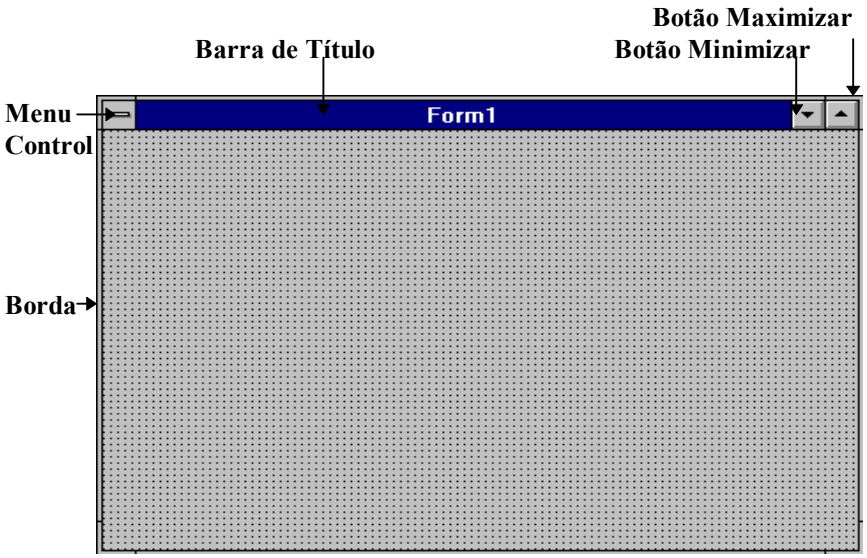
Quando você utiliza o Delphi, aparecem as seguintes janelas. Estas janelas formam a interface do Delphi. Outras janelas e elementos aparecem posteriormente conforme for desenvolvendo uma aplicação.



Forms

Introdução

Forms são a característica central das aplicações Delphi. Quando você abre o Delphi, um form se torna uma janela em sua aplicação. Quando estiver desenvolvendo a interface de usuário de sua aplicação, você adicionará itens chamados de componentes ao seu form. Um componente é um objeto Delphi, tal como um label box ou edit box. Quando você inicializa o Delphi, um form em branco Form1 aparece, como segue:



Partes de um Form

Cada form, por default, contém as seguintes partes padrões, como mostrado:

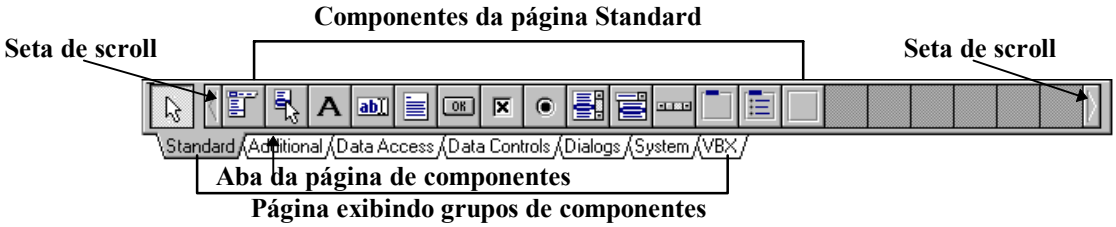
- Menu Control
- Botões de maximizar e minimizar
- Barra de título
- Bordas

Um form pode ser movido e reajustado movendo-se suas bordas.

Component Palette

Introdução

Após iniciar com um form, o próximo passo no design da interface é adicionar componentes ao form. Componentes são elementos de programa das janelas, tais como labels, edit boxes, e list boxes. Os componentes do Delphi estão localizados na Component Palette e são utilizados para construir a interface com o usuário. A Componente Palette exibe uma seleção de componentes agrupados por função ou utilização.
Um ponto chave do Delphi é que ele permite a criação de seus próprios componentes e personalização da Componente Palette. Você pode adicionar ou remover componentes da paleta, bem como criar uma nova página na paleta. Por exemplo, você pode adicionar um controle VBX de terceiros à sua paleta para um projeto em específico e depois remove-lo quando tiver sido utilizado. Você também pode criar diferentes versões da Componente Palette para diferentes projetos e permitir que diversos desenvolvedores compartilhem uma Componente Palette personalizada.
A Componente Palette aparece abaixo da barra de menu, a direita, e consiste dos seguintes botões e páginas:



Quando você passar com o ponteiro do mouse sobre cada botão, é exibido o Help Hint do componente.

Descrição das Páginas dos Componentes

Quando você clica com o mouse sobre uma aba de página, é exibido o grupo de componentes da página. A tabela a seguir descreve a função de cada grupo de componentes por página:

Página	Grupo de Componentes
Standard	Componentes padrão em uma interface MS Windows tais como botão, list box, e label.
Additional	Grupo adicional de componentes padrão, tais como SpeedButton, TabSet, e componentes Notebook.
Data Access	Componentes especializados para acesso de dados em banco de dados, tais como Table, Query e DataSource
Data Controls	Componentes especializados de banco de dados utilizados para exibir dados de bancos de dados, tais como Grid de dados, Navigator,e Edit.
Dialogs	Quadros de dialogos comuns do MS Windows que possuem uma aparência consistente para executar operações de arquivo, tais como abertura, gravação e impressão.
System	Componentes que pertencem à tecnologia do sistema Windows, tais como um timer, DDE, ou OLE.
VBX	Controle Visual Basic que acompanham o Delphi ou de outros fabricantes.
Samples	Componentes diversos, tais como ColorGrid, Calendar e SpinButton.

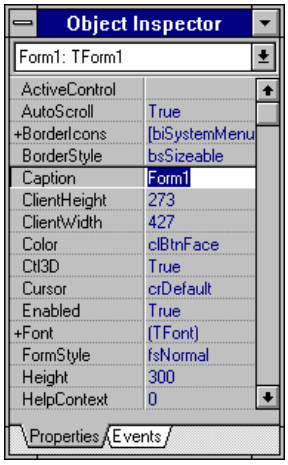
Object Inspector

Introdução

O Object Inspector trabalha com a Component Palette. Uma vez selecionado um componente da Component Palette é adicionado ao form, o Object Inspector automaticamente exibe as propriedades e eventos que podem ser utilizados como o componente. As propriedades e eventos (exibidos no formato de menu) permitem personalizar os componentes visualmente sem a necessidade de codificação.
Os menus são dinâmicos no tocante em que somente as propriedades e eventos que se aplicam aos componentes selecionados aparecerão. Se múltiplos componentes são selecionados, somente as propriedades e eventos compartilhados por todos os componentes aparecerão no Object Inspector.

Aparência do Object Inspector

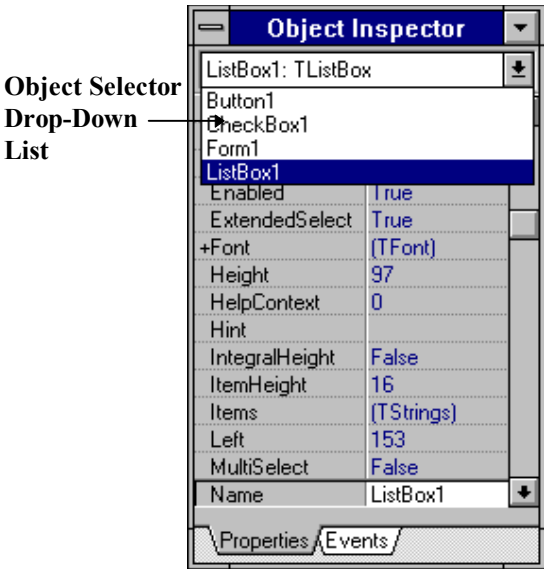
Por default, o Object Inspector aparece a esquerda da janela do Delphi, como segue:



Object Selector

A lista drop-down do Object Selector exibe o nome e o tipo de objeto de cada componente no form atual, incluindo o próprio form. Os componentes aparecem na lista quando você os adiciona ao form. A lista

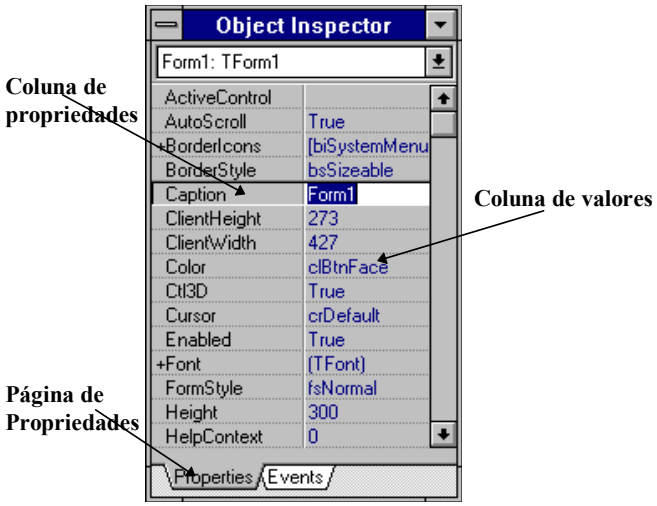
drop-down pode então ser utilizada para alternar rápidamente entre cada um dos componentes. A ilustração a seguir mostra um exemplo da lista drop-down do Object Inspector:



Quando um form ou componente é selecionado no Object Selector, as propriedades ou eventos pertencentes a ele são exibidos.

Properties Page

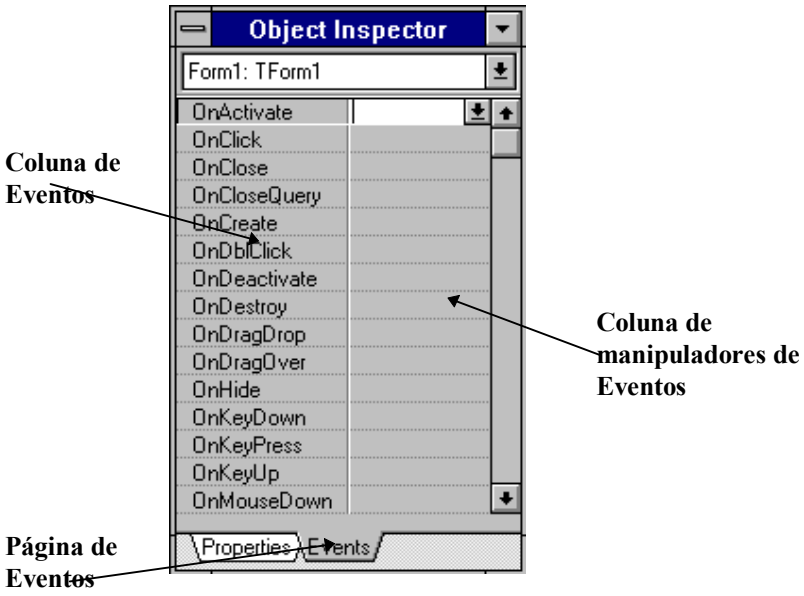
Propriedades são atributos de um componente. Elas controlam a forma como um componente aparece e se comporta na aplicação. Muitas propriedades na coluna Properties tem um valor default atribuído na coluna Values. A figura a seguir mostra um exemplo da página Properties.



Events Page

A página Events do Object Inspector lista os eventos que um componente pode reconhecer. No Delphi, você escreve procedures chamadas event handlers, e as utiliza na página Events para associar o nome de um event handler com o evento. Eventos são ações do usuário ou

ocorrências de sistema que o componente pode reconhecer. Um exemplo de uma ação de usuário é um clique em um botão. Um exemplo de uma ocorrência de sistema é um alarme gerado em um intervalo de tempo pré-determinado. A ilustração a seguir mostra um exemplo da página Events:



Code Editor

Introdução

No processo de desenvolvimento, após atribuir propriedades e eventos a cada componente adicionado ao form, o Code Editor é utilizado. O Code Editor é um editor de texto que exibe o código fonte que você

escreve ou que o Delphi gera para criar uma aplicação. O código fonte aparece em um arquivo chamado UNIT.PAS que é um dos tipos de arquivos que compõe um projeto do Delphi.

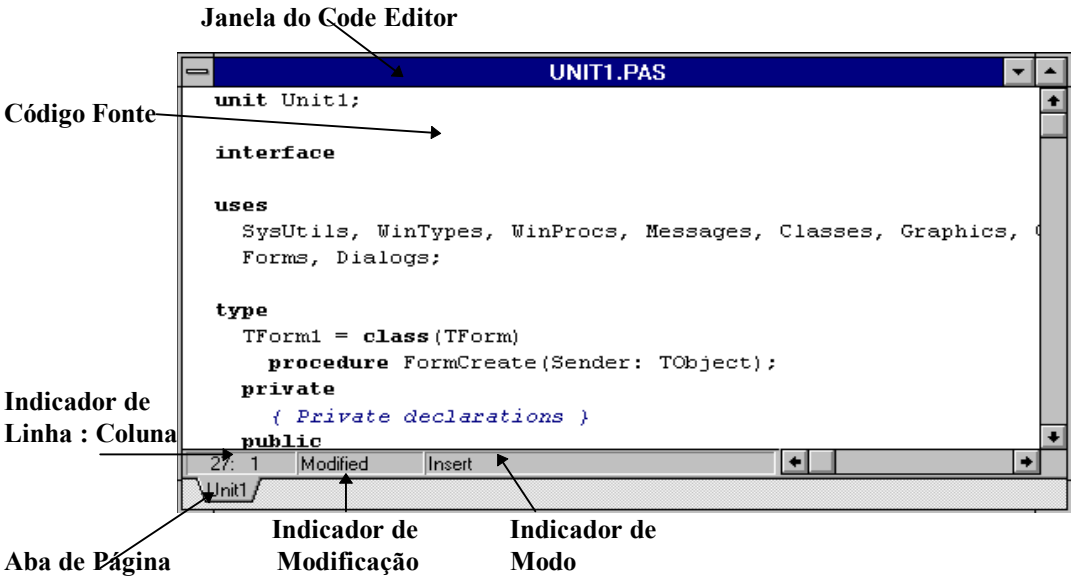
Características do Code Editor

O Code Editor oferece comandos de edição, Help sensível ao contexto, e as seguintes características de edição estilo Brief:

- Gravação e execução de macros
- Sintaxe de destaque colorida
- Undo ilimitado
- Marcação de bloco de coluna
- Toques de teclado personalizáveis

O Arquivo Unit

O Code Editor aparece no arquivo unit, que é posicionado atrás do Form1 default quando você inicializa o Delphi. Quando você da um duplo clique em um evento na página Events do Object Inspector, o Code Editor automaticamente torna-se a janela para que você possa digitar o código para manipular o evento. O Code Editor pode exibir múltiplos arquivos unit.
Uma unit é utilizada para organizar as declarações de programação do Delphi. Por exemplo, procedures utilitárias podem ser agrupadas como uma unit. Um form é organizado como uma unit.
Você também pode exibir um arquivo unit através do menu View selecionando **Toggle Form/Unit**. O arquivo default, Unit1.PAS, aparece na janela do Code Editor, como segue:



Você deve evitar o nome default de um arquivo unit. Renomeie a Unit1 para um nome mais descritivo sobre o conteúdo do arquivo. A extensão PAS não deve ser alterada.

SpeedBar

Introdução

A SpeedBar é feita de botões que economizam tempo, permitindo que você clique sobre um botão ao invés de utilizar a barra de menu para executar operações e comandos mais comuns. A SpeedBar aparece abaixo da barra de menu, à esquerda, e consiste dos seguintes botões:

Quando você passa com o ponteiro do mouse sobre cada botão, o Help Hint do botão é exibido. A SpeedBar é totalmente personalizável. Você pode adicionar ou remover botões para personalizá-la. Os botões exibidos na figura anterior é a configuração default.

Descrição dos Botões Default

A tabela a seguir descreve cada botão na SpeedBar default:

Botão	Descrição
Open Project	Abre um projeto existente
Save Project	Grava um projeto existente
Add File to Project	Adiciona um novo arquivo ao projeto
Select Unit From List	Seleciona uma unit da lista das units existentes
Select Form From list	Seleciona um form da lista de forms existentes
Run	Compila e executa sua aplicação
Pause	Interrompe momentaneamente a execução da aplicação
Open File	Abre um arquivo existente
Save File	Grava o arquivo. Arquivos unit e form são ligados. Gravando um, grava-se o outro
Remove File From Project	Remove o arquivo selecionado no Project Manager da cláusula *uses* do arquivo de projeto corrente
Toggle Between a Form and Unit	Exibe o from inativo associado com a unit ativa, ou vice-versa
New form	Cria um form em branco e uma nova unit associada para ser adicionado ao projeto
Trace into	Executa um programa, uma linha por vez e executa cada linha de uma procedure
Step over	Executa um programa, uma linha por vez e pula as procedures executando-as como uma única unit

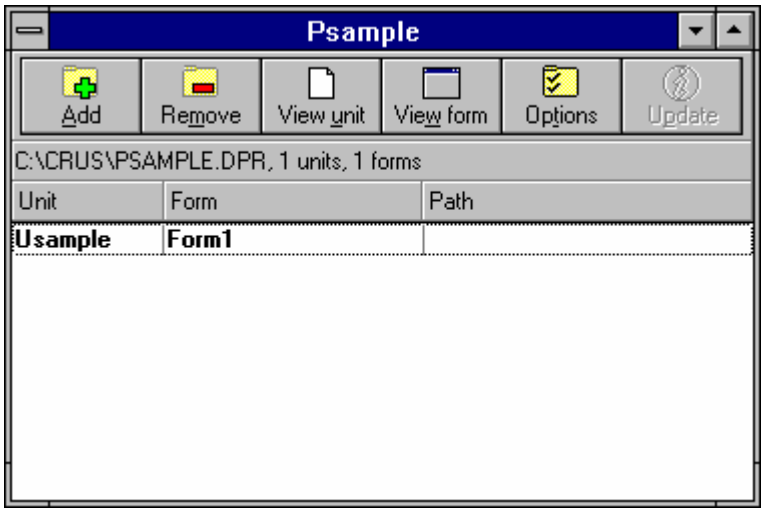
Project Manager

Introdução

No Delphi, os arquivos que compõem uma aplicação, form e units, são organizados como um projeto. O Project Manager lista cada arquivo de unit e form em sua aplicação e permite navegar entre elas.

Visualizando a Janela do Project Manager

Você pode exibir o project Manager através do menu View selecionando Project Manager. O Project Manager aparece com o nome do projeto. Se o projeto não tiver um nome, o arquivo default chamado Project1, aparece como segue:



SpeedMenus

Introdução

Um elemento importante do IDE do Delphi é o SpeedMenu. SpeedMenus são menus que oferecem acesso rápido aos comandos disponíveis no momento. Eles são sensíveis ao contexto e podem ser exibidos de duas maneiras:

- Clicando-se com o botão direito do mouse enquanto o ponteiro do mouse estiver sobre o objeto
- Pressionando-se **Alt+F10** enquanto estiver selecionado

Exemplos de SpeedMenus

SpeedMenus estão disponíveis para uma grande variedade de elementos e outros objetos no Delphi. Para uma lista completa, consulte o Help on-line. Esta seção oferece alguns exemplos de SpeedMenus para os seguintes elementos:

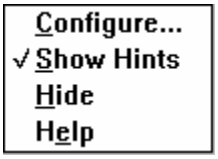
- Component Palette
- Object Inspector
- SpeedBar

SpeedMenu para a Component Palette

O SpeedMenu da Component Palette aparece quando você clica com o botão direito do mouse quando o ponteiro do mouse estiver sobre a Component Palette. Os itens do menu são :

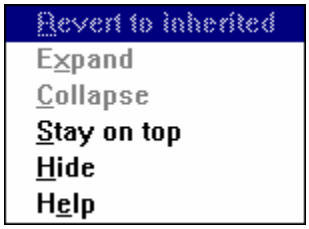
- **Configure** - Altera o conteúdo da Component Palette
- **Show Hints** - Exibe o Help Hint para cada botão
- **Help** - Oferece Help on-line para a Component Palette

Na figura a seguir, **Show Hints** está habilitado:



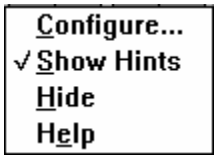
SpeedMenu para o Object Inspector

O SpeedMenu do Object Inspector aparece, como segue, quando você clicar com o botão direito do mouse quando o ponteiro do mouse estiver sobre o Object Inspector. Quando *Stay on top* estiver habilitado, o Object Inspector sempre aparece sobre a interface.



SpeedBar

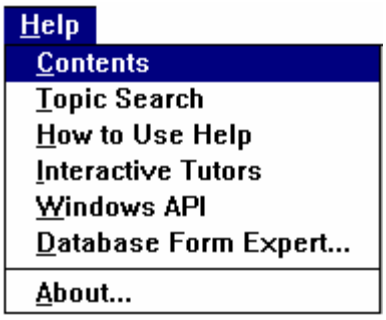
O SpeedMenu da SpeedBar aparece, como segue, quando você clicar com o botão direito do mouse quando o ponteiro do mouse estiver sobre a SpeedBar. O item de menu Configure permite adicionar ou remover icones da SpeedBar de acordo com sua necessidade.



Help On-Line

Introdução

O Help On-Line é uma grande fonte de referência sobre o Delphi. No menu Help, os itens Contents e Topic Search são comuns as aplicações MS Windows, mas documentam o produto tão bem que você pode utiliza-lo no lugar de uma cópia impressa.Contents, em particular, inclui um glossário, que é especialmente útil para uma referência rápida sobre o significado de termos.
A figura a seguir mostra o menu **Help**:



Help Sensível ao Contexto

Help sensível ao contexto está disponível para cada elemento na interface. Pressionando **F1** sobre um elemento destacado exibe a descrição do elemento. Por exemplo, se Form1 estiver destacado e você pressionar **F1**, a descrição do componente TForm (tipo de objeto) aparece, incluindo as propriedades, métodos, eventos e tarefas associadas a ele. O Help sensível ao contexto também está disponível quando o Code Editor for exibido e oferece auxílio na sintaxe da linguagem, palavras reservadas, e assim por diante.

Informação de Instrução

O Help on-line também contém uma extensiva informação instrucional. As três ferramentas seguintes, em particular, oferece suporte no desenvolvimento de uma aplicação:

- Interactive Tutors o guiam através de cada estágio no processo de desenvolvimento de uma aplicação.
- Database Form Expert permite-lhe criar forms master-detail, bem como forms com uma única tabela.
- Windows API descreve como utilizar a API do MS Windows.

Interactive Tutors

A opção Interactive Tutors é um tutorial que o ensina as operações básicas na criação de uma aplicação. O tutorial é interativo pois você pode utilizar seus próprios dados, bem como dados de exemplo que acompanham o Delphi

Consulte o tutorial para relembrar o conhecimento obtido no curso.

Database Form Expert

A opção Database Form Expert no menu Help permite criar um form que exiba dados de um banco de dados local ou remoto. Esta ferramenta é uma das muitas ferramentas Expert no Delphi. Outras ferramentas Expert incluem o Application Template Expert, Form Template Expert, Component Expert, e Dialog Box Expert. A maioria delas são acessadas através do menu File quando você cria uma aplicação ou form. As ferramentas Expert serão estudadas com mais detalhes posteriormente durante o curso.

Windows API

A opção API no menu Help oferece uma descrição da API do MS Windows. Exemplos de alguns dos tópicos incluem o seguinte:
Funções e mensagens para o MS Windows versão 3.1

- Grupos de funções
- Macros
- Saídas de impressão
- Resources

Resumo do Capítulo

Pontos Chave

Após completar este capítulo, você aprendeu:

- Uma aplicação desenvolvida no Delphi começa com um form em branco.
- Utilizar a Component Palette para adicionar componentes ao form, e depois utilizar o Object Inspector para definir propriedades e programar eventos aos componentes.
- Utilizar o Code Editor, que exibe o código fonte no arquivo unit, adicionar código a um evento handler se você adicionou um evento no Object Inspector.

Termos e Definições

A tabela a seguir é uma referência rápida aos termos explicados neste capítulo:

Termo	Definição
Application Templates	Ferramenta de design predefinidas para iniciar o desenvolvimento de uma aplicação
Arquivo Unit	Um arquivo que contém uma parte ou todo o código fonte de uma aplicação. Cada form possui um arquivo unit associado. Algumas units podem não estar associadas a um form.
Code Editor	O editor de texto que lhe permite escrever código de programação
Component Palette	A barra de botões de componentes para a construção de uma aplicação.
Componente	Um objeto Delphi utilizado para construir uma aplicação MS Windows. Um componente também é chamado de objeto de programa
Database Form Designer	Ferramenta de design para gerar um form que possa exibir dados de um banco de dados externo.
Event	Uma interação ou ação de usuário, ou uma ocorrência interna de sistema.
Event Handler	Uma procedure ou função executada sempre que ocorrer um evento.
Form	Um componente utilizado para construir uma aplicação é o primeiro a aparecer na tela. Um form pode conter outros componentes. O form se torna uma janela na aplicação.
Object Inspector	A janela que exibe propriedades e eventos utilizados para definir ou ver as propriedades de um componente.
Projeto	Todos os arquivos que contém uma aplicação.

Propriedade	Um atributo descritivo atribuído a um componente para definir sua aparência e como trabalha.
SpeedBar	Uma barra de botões onde cada uma é relacionado a um comando utilizado com freqüência. A SpeedBar permite rápido acesso a estas operações, que são itens de menu.
SpeedMenus	Menus que se aplicam ao elemento selecionado no IDE do Delphi. Eles são exibidos clicando-se com o botão direito do mouse sobre o elemento.
Templates de Form	Forms predefinidos utilizados para desenvolver uma interface com o usuário.

Capítulo 3

Desenvolvendo uma Aplicação

Ao final deste capítulo, você deve estar apto a :

- Desenvolver uma aplicação
- Utilizar o Project Manager
- Utilizar o Intergrated Debugger

Overview

O Delphi é uma poderosa linguagem de programação orientada a objeto com um avançado ambiente de desenvolvimento visual. Estas características, quando combinadas com a arquitetura de bancos de dados Borland, permite criar rápidamente aplicações Client/Server. Neste capítulo , você aprenderá sobre o processo de construção de uma aplicação Delphi, criar uma aplicação de exemplo e explorar os conceitos de gerenciamento de projetos.

Descrição do processo

Introdução

Uma característica do Delphi é que ele permite construir aplicações rápidamente. Esta seção descreve o processo de desenvolvimento de aplicações Delphi, como segue:

- Criando um Projeto
- Adicionando um Form ao Projeto
- Adicionando Componentes ao Form
- Definindo Propriedades dos Componentes
- Adicionando Event Handlers
- Compilando, Executando e Depurando a Aplicação

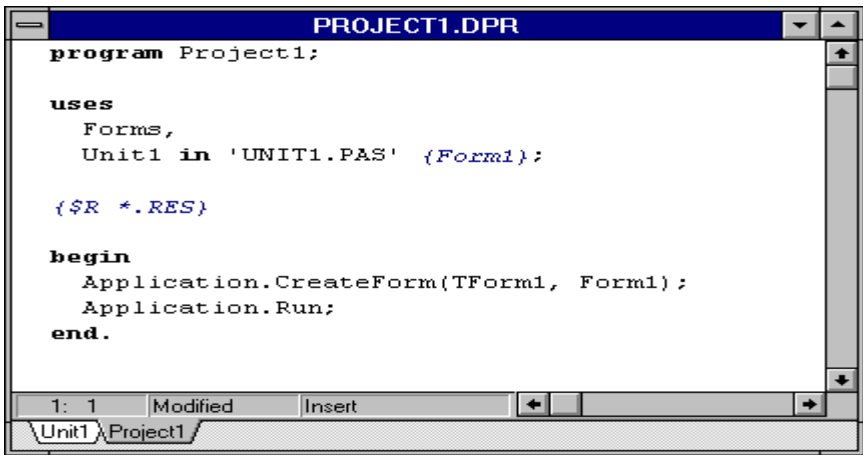
Estágio 1 - Criando um Projeto

O primeiro passo no desenvolvimento de uma aplicação no Delphi é criara um projeto. Aplicações no Delphi são gerenciadas como projetos. Criar um novo projeto gera um arquivo de projeto. O arquivo de projeto controla uma aplicação Delphi construindo os vários forms, executando a aplicação e exibindo o form principal da aplicação.

Arquivos de projeto contém código fonte Object Pascal gerado pelo Delphi que se torna parte do executável da aplicação quando for compilado e "Linkado".

Você pode começar um novo projeto inicializando o Delphi. Sempre que você inicializar o Delphi, um novo projeto é aberto. Se o Delphi já estiver aberto, você pode abrir um novo projeto através do menu **File**, Selecionando **New Project**.

O Delphi cria um arquivo de projeto default chamado PROJECT1.DPR, que o Delphi mantém durante o desenvolvimento da aplicação. Conforme o projeto for alterado, tal como adicionando novos forms, o Delphi atualiza o arquivo de projeto. O arquivo de projeto se parece com a figura a seguir:

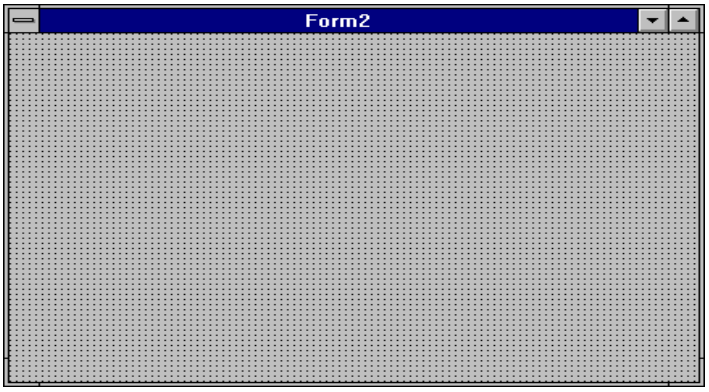


Estágio 2 - Adicionando um Form ao Projeto

Como discutido anteriormente, os forms são a fundação das aplicações Delphi e fornece uma tela onde você pode criar a interface com o usuário de seu programa. Um projeto ou aplicação geralmente possui múltiplos forms. Adicionar um form ao projeto cria os seguintes arquivos adicionais:

- Um arquivo form com extensão .DFM contendo informações de resources para a construção do form.
- Um arquivo unit com extensão .PAS contendo código Object Pascal.

Todo form em uma aplicação possui estes dois arquivos associados a ele. Conforme for adicionando novos forms, o arquivo de projeto é atualizado automaticamente. O exemplo a seguir mostra o Form2, a ser adicionado ao projeto.

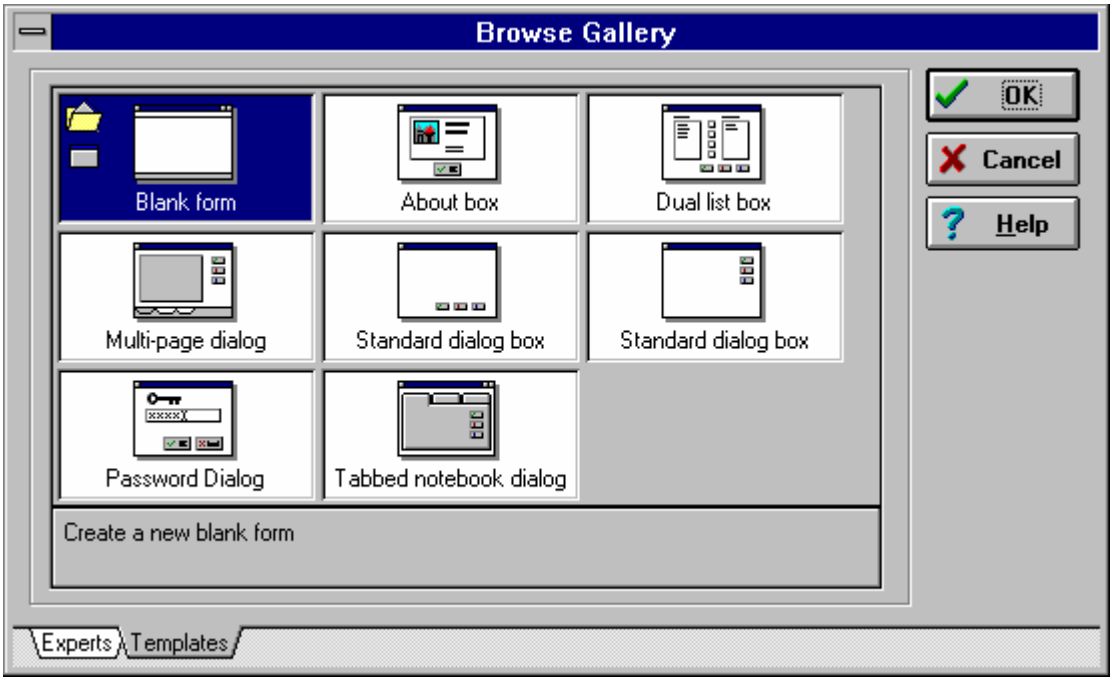


Quando você cria um projeto, um form (Form1) é criado por default. O nome de arquivo deste form é UNIT1.DFM.

Para Adicionar um Form ao Projeto

Execute os seguintes passos para adicionar um form um ao projeto:

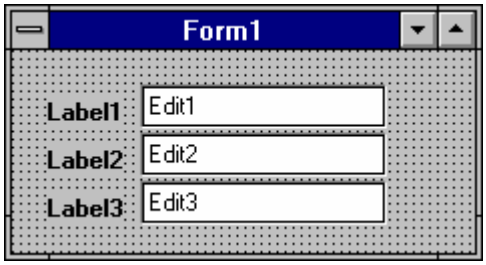
Passo	Ação
1	Para adicionar um ou mais forms ao projeto, no menu **File**, selecione **New Form**. Dependendo das configurações de Gallery na página Preferences do quadro de dialogo Environment Options, pode ocorrer: - Um form em branco aparece e é adicionado ao projeto - O quadro Browse Gallery aparece, como segue:



Passo	Ação
2	Aparecendo o quadro de dialogo Browse Gallery, selecione o tipo de form que você deseja adicionar ao projeto e de um clique em **OK**.

Estágio 3 - Adicionando Componentes ao Form

Como discutido anteriormente, componentes são objetos visuais de programa que você manipula durante o design. Os componentes disponíveis no momento estão na Component Palette. Após inserir um componente no form, você pode move- lo, edita-lo e reajusta-lo de acordo com suas necessidades. O exemplo a seguir mostra três componentes Label e três componentes Edit em um form:



Passos para Adicionar um Componente em um Form

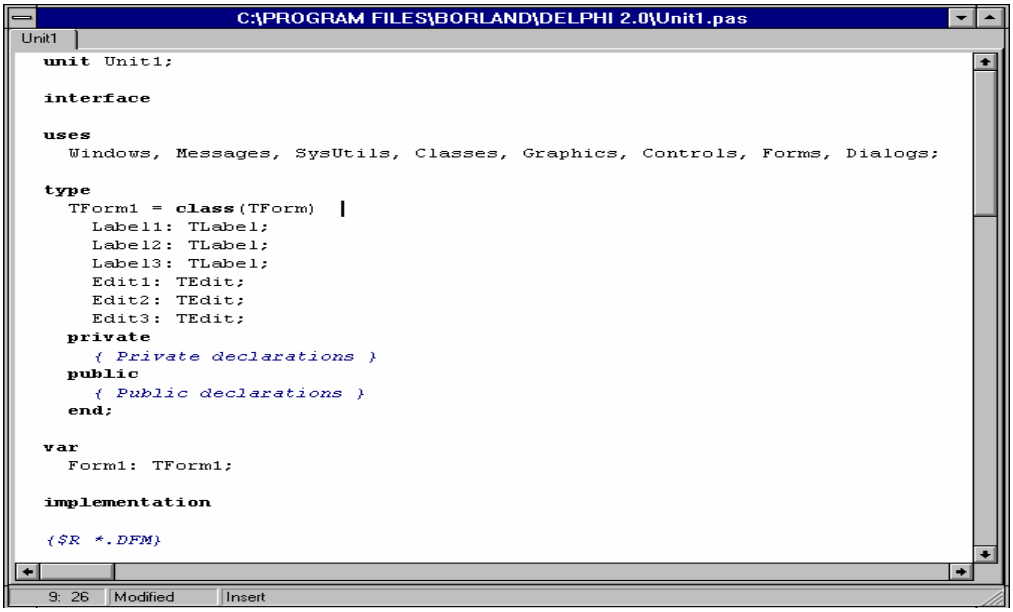
Execute os seguintes passos para adicionar um componente ao form:

Passo	Ação
1	De um clique sobre um componente na Component Palette.
2	De um clique sobre o form onde o componente deve aparecer.
3	Reajuste o componente arrastando seus manipuladores.

Você também pode adicionar um componente com um duplo-clique no componente na Component Palette. Um componente até o local desejado e reajusta-lo através dos manipuladores.

Código Fonte da Unit após Adicionar Componentes

Quando um componente é adicionado ao form, o código fonte do arquivo é modificado. Especificamente a definição type para o membro correspondente ao componente adicionado. O exemplo a seguir exibe a definição de type no arquivo unit que corresponde ao form mostrado anteriormente com três componentes Label e três componentes Edit:



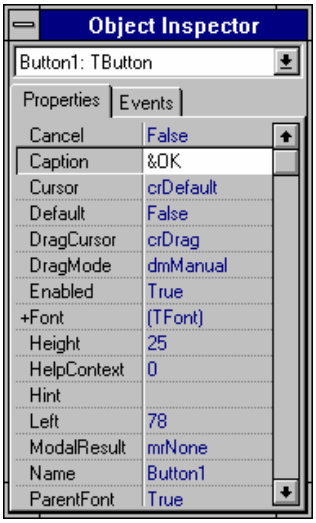
Estágio 4 - Definindo Propriedades dos Componentes

Como discutido anteriormente, cada componente tem um conjunto de atributos chamados de propriedades. Cada propriedade que possa ser alterada durante o design é exibido no Object Inspector. Você pode definir propriedade durante o design ou codificar para que uma propriedade do componente seja alterada durante a execução do programa.

Passos para Definir uma Propriedade de um Componente

Execute os passos a seguir para definir uma propriedade de um componente durante o design:

Passo	Ação
1	Dê um clique sobre o componente no form cuja propriedade você queira alterar.
2	Dê um clique sobre a aba da página Properties no Object Inspector. O exemplo a seguir mostra a página Properties para o componente Button.
3	Selecione a propriedade que você queira definir.
4	Altere o valor da propriedade na coluna Values.



Estágio 5 - Adicionando Event Handlers

Em programas dirigido a evento, responder a eventos do usuário ou do sistema é um ponto chave de sua aplicação. Alem das propriedades, os componentes possuem uma lista de eventos que podem ser detectados. Quando você adiciona event handlers a sua aplicação, você está dizendo ao componente que execute os comandos programados sempre que um evento em específico seja detectado. Por exemplo, um botão em seu form pode detectar quando um usuário clica sobre si, o que é conhecido como evento OnClick. O evento OnClick faz com que sua aplicação execute a ação especificada no event handler.

Passos para Adicionar um Event Handler

Execute os passos a seguir para adicionar um event handler:

Passo	Ação

1	Dê um clique sobre o componente no form que precise de um event handler.
2	Dê um clique sobre a aba da página Events no Object Inspector para exibir uma lista de eventos para o componente.
3	Dê um duplo-clique à direita da coluna de eventos para fazer com que o Delphi gere um event handler vazio e exiba o handler no Code Editor.
4	Digite o código a ser executado quando ocorrer o evento.

Estágio 6 - Compilando, Executando e Depurando a Aplicação

O compilador e o depurador (debugger) são partes do ambiente Delphi. O compilador inclui um habilitador Make automático para que quando sua aplicação for alterada, somente os arquivos alterados sejam recompilados. O debugger está ativo sempre que você executar aplicações dentro do ambiente do Delphi.

Passos para Compilar e Executar a Aplicação

Execute os passos a seguir para compilar e executar a aplicação:

Passo	Ação
1	Para compilar o projeto atual sem inicializar o arquivo executável resultante, no menu Compile, selecione Compile.
2	Para compilar quaisquer alterações e executar o arquivo do projeto corrente, selecione Run no menu Run.

Tutoria: Criando uma Aplicação

Introdução

O processo a seguir é um tutorial de exemplo. Uma maneira de entender o processo de desenvolvimento de aplicações no Delphi é construir uma aplicação de exemplo. Esta seção fornece um rumo na construção de uma aplicação simples utilizando diversos componentes padrão do Delphi. A aplicação permite que o texto seja digitado em edit box e adicionado em um list box com o clique de um botão.

Estágios do Tutorial

O processo deste tutorial envolve os seguintes estágios:

Estágio	Processo
1	Criar um projeto de exemplo
2	Adicionar componentes padrão
3	Definir propriedades dos componentes
4	Adicionar um event handler
5	Compilar e executar a aplicação de exemplo

Passos para Estágio 1

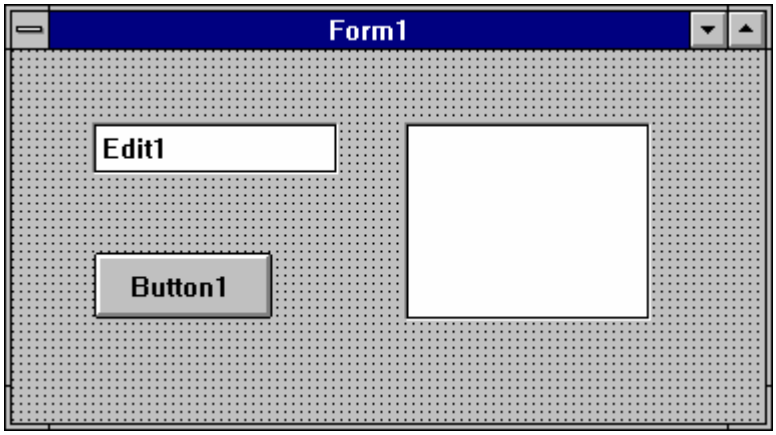
Execute os passos para abrir um novo projeto e chamá-lo PSAMPLE.DPR:

Passo	Ação
1	No menu File, selecione New Project para criar um projeto. O processo de abertura de um novo arquivo projeto adiciona automaticamente um novo form. Se o quadro de dialogo Browse Gallery aparecer, **Blank Form** é a opção default. Dê um clique em **OK**.
2	Se for solicitado a gravação das alterações do projeto atual selecione **No**.
3	No menu File, selecione **Save Project As**.
4	Quando o nome da unit for solicitado, digite *USAMPLE.PAS*. Este nome substitui o nome default UNIT1.PAS.
5	Quando o nome do projeto for solicitado, digite *PSAMPLE.DPR*. Este nome substitui o nome default PROJECT1.DPR.

Passos para o Estágio 2

Execute os passos a seguir para adicionar componentes da página Standard ao form no projeto PSAMPLE:

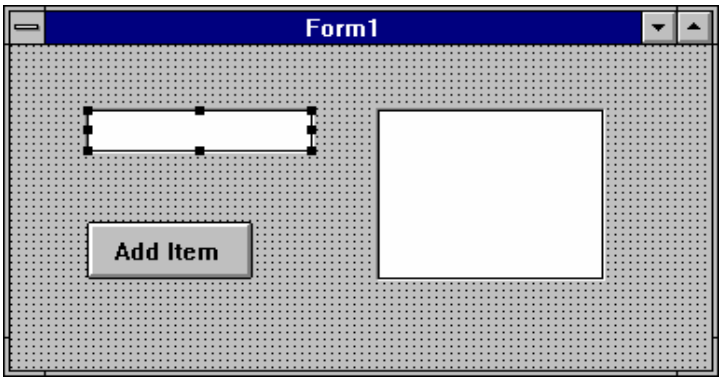
Passo	Ação
1	Dê um clique sobre a aba da página Standard da Component Palette para visualizar os componentes Standard.
2	Mova o ponteiro do mouse vagarosamente sobre cada componente para que o Help Hint seja exibido, e localize os componentes Button, Edit e ListBox.
3	Dê um duplo-clique sobre o componente Button para inseri-lo no form. Arraste o componente até a parte inferior do form.
4	Dê um duplo-clique sobre o componente Edit, e mova o componente até a parte superior do form.
5	Dê um duplo-clique sobre o componente ListBox para inseri-lo no form.
6	Arranje os componentes e reajuste o tamanho do form para que se pareça coma figura a seguir.



Passos para Estágio 3

Execute os passos a seguir para definir as propriedades dos componentes Edit e Button:

Passo	Ação
1	Dê um clique sobre o componente Edit. As propriedades do componente Edit são exibidas no Object Inspector. O nome default deste componente é Edit1, e seu tipo de objeto é TEdit.
2	No Object Inspector, dê um clique sobre a coluna Values da propriedade Text de Edit1 e a apague.
3	Defina a propriedade Caption para **Add Item**. Seu form deve se parecer com a figura a seguir:



Passos para o Estágio 4

Execute os passos a seguir para adicionar um event handler para o evento OnClick do componente **Add Item**:

Passo	Ação
1	Dê um clique sobre o botão **Add Item** de seu form para exibir as propriedades no Object Inspector.
2	Dê um clique sobre a aba Events do Object Inspector para exibir a página Events do botão.
3	Dê um duplo-clique sobre a coluna a direita do evento OnClick. O nome da procedure Button1Click aparecera na coluna. O Delphi gera um event handler vazio e o exibe no Code Editor. Digite o código a seguir dentro das declarações **begin....end;** da procedure.

```
if Edit1.Text<> '' then { Edit1 não esta vazio }
begin
   ListBox1.Items.Add(Edit1.text);
    Edit1.Text := '';
end;
```

Passos para o Estágio 5

Execute os passos a seguir para compilar e executar sua aplicação:

Passo	Ação
1	No menu Run, selecione Run. Esta opção compila e executa sua aplicação.
2	Digite algum valor, tal como seu nome, no componente Edit.
3	Dê um clique em Add Item para adicionar cada item a list box. Adicione nove ou dez itens a list box.
4	No menu File, selecione Save Project, e depois Close Project.

Utilizando o Project Manager

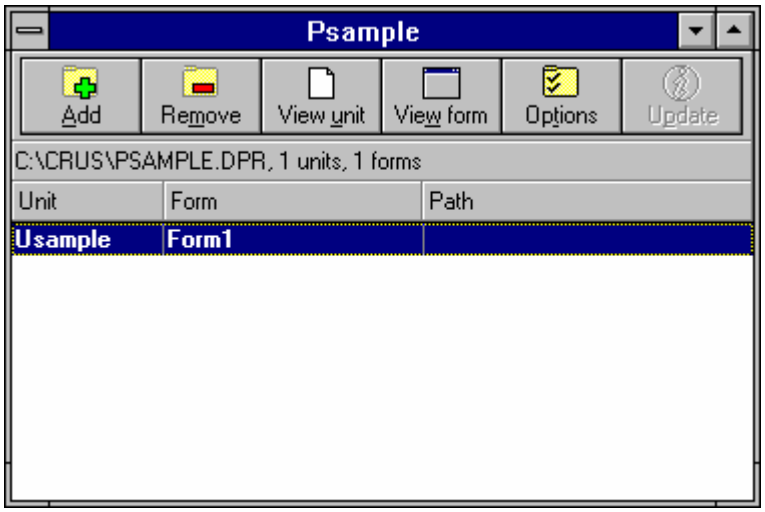
Introdução

O Delphi permite que você gerencie seus projetos através do Project Manager. O Project Manager lista os arquivos que compõe seu projeto e permite que você navegue pelos arquivos. Você também pode utilizar o Project Manager para:

- Adicionar units e forms ao projeto
- Remover units e forms de um projeto
- Especificar o form principal
- Especificar a localização dos arquivos de Help e ícone

Exibindo a Janela do Project Manager

Para exibir o Project Manager, no menu **View**, selecione **Project Manager**. A janela Project Manager aparecerá, como segue:



Descrição dos Botões do Project Manager

A tabela a seguir descreve os botões do Project Manager:

Botã0	Descrição
Add	Adiciona um arquivo de unit ao projeto atual
Remove	Remove um arquivo de unit do projeto atual
View unit	Exibe um arquivo de unit do projeto atual
View form	Exibe um form no projeto atual
Options	Exibe o quadro de diálogo Project Options para: • Alterar o form default, opções do compilador, aplicação, linker e diretório • Define símbolos condicionais
Update	Grava as alterações no projeto atual

Adicionando Units e Forms ao Projeto Utilizando o Project Manager

Um projeto default inicialmente contém um form e um arquivo de unit de código fonte. Entretanto, a maioria dos projetos contém múltiplos forms e units.
Você pode dar um clique com o botão direito do mouse e selecione **New Form** no SpeedMenu para adicionar units e forms ao seu projeto. Você também pode adicionar forms e units existentes ao seu projeto utilizando o botão **Add** na SpeedBar do Project Manager e selecionar o form ou unit a ser adicionado.

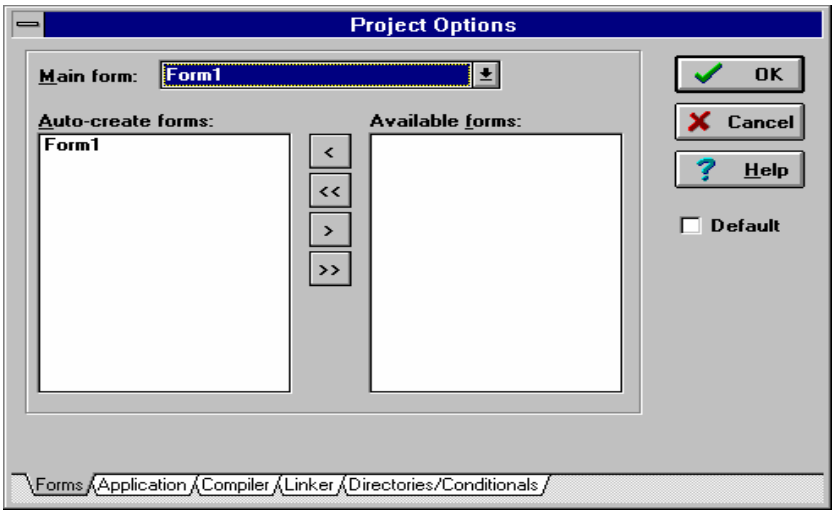
Removendo Units e Forms de um Projeto Utilizando Project Manager

Durante o desenvolvimento de um projeto, você pode achar necessário remover forms e units de seu projeto. Você pode fazê-lo a qualquer momento durante o desenvolvimento. Entretanto, como um form está sempre associado a uma unit, você não pode remove um sem remover o outro a menos que a unit não possua nenhum form associado a ele. Você pode remover units de um projeto utilizando o botão Remove no Project Manager.

Remover arquivos utilizando qualquer outro programa gerenciador de projeto ou digitando comandos no prompt do DOS não é recomendado. Estas ações não removem as entradas da cláusula Uses dos arquivos .DPR ou da janela do Project Manager e causará erros quando você compilar o programa.

Definindo Opções de Projeto

O quadro de dialogo Project Options permite definir diversas opções que afetam seus projetos. Para acessar o quadro de dialogo Project Options, de um clique sobre o botão Options no Project Manager. O quadro Project Options aparecera, como segue:



Descrição das Páginas do Project Options

A tabela a seguir descreve cada uma das páginas no quadro de dialogo Project Options e algumas das opções mais importantes em cada página:

Página	Descriçãao
Forms	Especifica o form principal de sua aplicação, os forms que devem ser criados automaticamente, e a ordem destes forms.
Application	Especifica um título, arquivo de help, e um ícone para sua aplicação.
Compiler	Permite definir opções para a forma como seu programa será compilado. Estas opções correspondem a definir diretivas a seu estado positivo (+) no código de seu programa.
Linker	Permite especificar a forma como seus arquivos de programa serão linkados.
Directories / Conditionals	Especifica a localização dos arquivos que o Delphi necessita para compilar, linkar e distribuir seu programa.

Utilizando Integrated Debugger

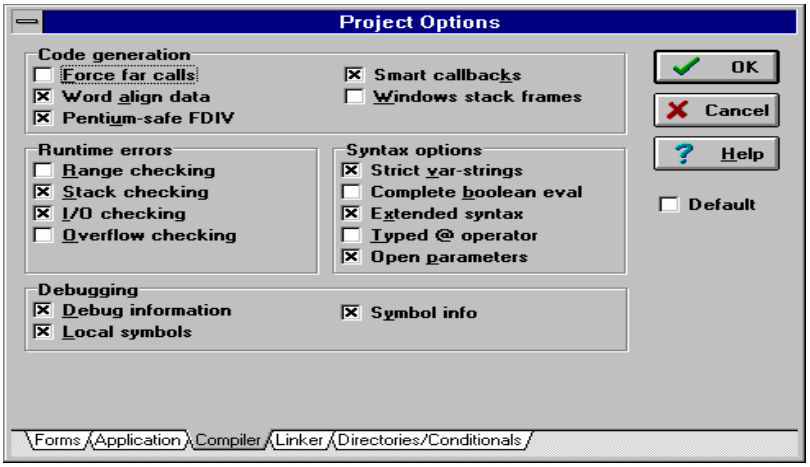
Introdução

O Delphi possui um depurador totalmente integrado que permite depurar uma aplicação sem deixar o ambiente visual de desenvolvimento. Ele oferece a capacidade de :

- Gerar informações a depuração de dentro de seu executável
- Habilitar e desabilitar a depuração integrada de dentro do IDE
- Definir breakpoints
- Visualizar o conteúdo das variáveis do programa
- Modificar valores de dados durante a execução do programa
- Visualizar o call stack

Gerando Informação de Depuração

O Delphi gera informação simbólica de depuração quando você compila seu programa com a ação de depuração habilitada. As opções de depuração encontram-se na página compiler do quadro de dialogo Project Options.
A página Compiler e opções de depuração para uma aplicação típica aparece como segue:



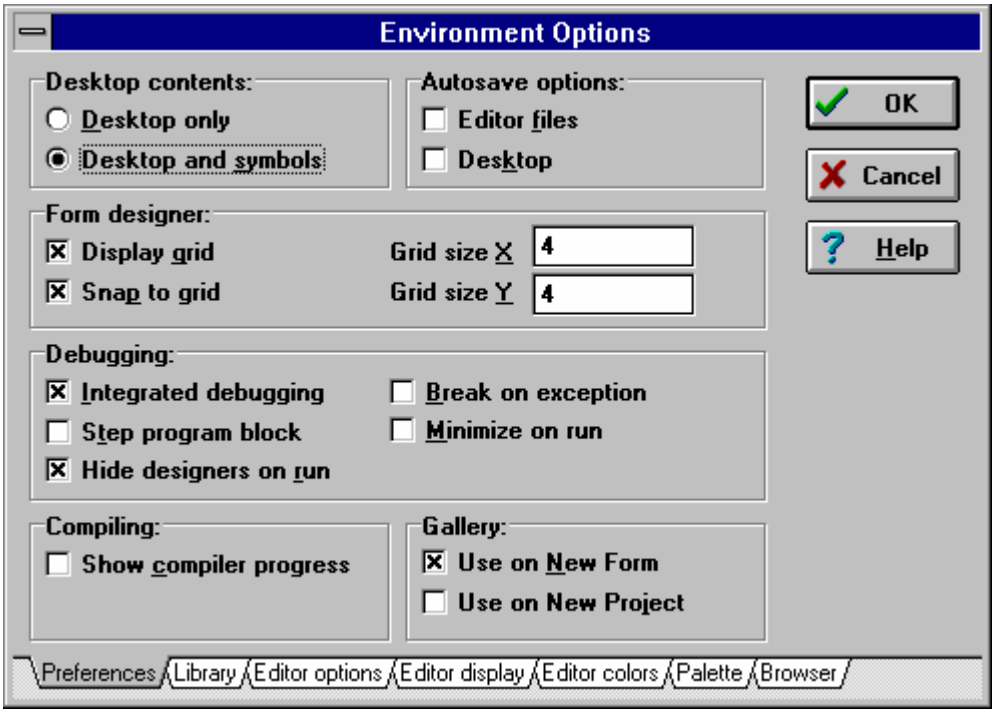
Descrição das Opções do Depurador na Página Compiler

A tabela a seguir descreve as opções de depuração na página Compiler do quadro de dialogo Project Options:

Opção	Descrição
Debug Information	Insere informação de depuração no arquivo UNIT.DCU. Informação de depuração consiste de uma tabela com linhas numeradas para cada procedure que mapeia endereços de código de objetos nos números.
Local Symbols	Gera local symbols information. Local symbols information consiste de : • Os identificadores na implementation part (e não na interface part) do módulo • Os identificadores dentro das procedures e funções do módulo. Local symbols information não inclui variáveis globais ou nomes declarados na interface section de uma unit.
Symbol Info	Gera symbol information. Symbol reference information consiste de tabelas que fornecem os números de linhas de todas as declarações e referências aos símbolos em módulo.

Habilitando o Intergrated Debugger

A depuração integrada está habilitada após a instalação do Delphi. É uma opção de ambiente da página Preferences que você pode definir. A página Preferences do quadro de dialogo Environment Options aparece como segue:



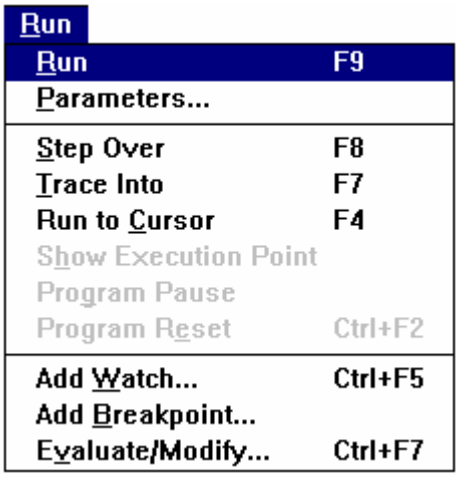
Descrição das Opções de Depuração na Página Preferences

A tabela descreve as opções de depuração na página Preferences do quadro de dialogo Environment Options:

Opção	Descrição
Intergrated Debugging	Quando habilitada, a depuração integrada está em efeito.
Step Program Block	Quando habilitada, o depurador passa pelos blocos de programa.
Hide Designer on run	Quando habilitada, a interface de design é oculta enquanto a aplicação estiver sendo executada.
Break on exception	Quando habilitada, a aplicação para quando ocorrer uma exception e a linha de código causadora é exibida no Code Editor.
Minimize on run	Quando habilitada, o Delphi é minimizado sempre que você executar uma aplicação dentro do IDE.

Controlando a Execução do Programa

Uma característica importante do depurador é que ele permite selecionar como o programa será executado, uma declaração por vez. Uma linha de código por vez, uma função inteira por vez, e assim por diante. O depurador permite controlar a execução de seu programa das opções no menu **Run**. O menu **Run**, como segue:



Descrição do Menu Run

A tabela a seguir fornece uma declaração das opções disponíveis no menu Run.

Opção	Descrição
Run	Permite compilar e executar suas aplicações.
Parameters	Permite especificar os parâmetros de inicialização para sua aplicação
Step Over	Permite executar seu programa uma linha por vez, pulando procedures, executando-as como uma única unit.
Trace Into	Permite executar seu programa uma linha por vez, rastreando a procedure e seguindo a execução de cada linha.
Run to Cursor	Permite executar o programa até a localização do cursor no Code Editor
Program Pause	Permite interromper temporariamente a execução de um programa
Program Reset	Permite terminar o programa sendo executado e remove- lo da memória
Add Watch	Abre o quadro de dialogo Watch Properties, permitindo criar e modificar observações
Add Breakpoint	Abre o quadro de dialogo Edit Breakpoint, permitindo criar e modificar pontos de interrupção
Evaluate/Modify	Abre o quadro de dialogo Evaluate/Modify,permitindo avaliar ou alterar o valor de uma expressão existente

Definindo Pontos de Interrupção

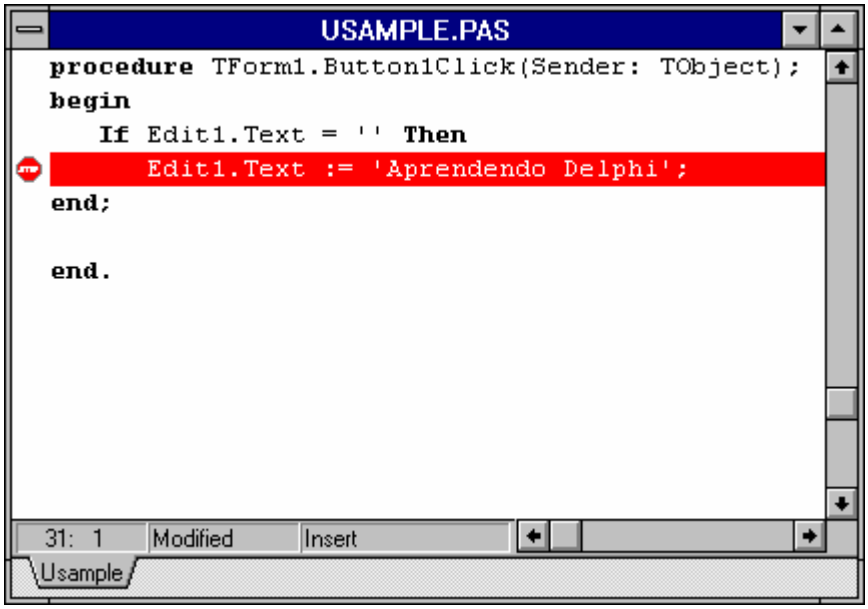
Você pode definir pontos de interrupção onde quiser que a execução seja interrompida. Pontos de interrupção são particularmente úteis quando utilizados em conjunto com Watches. Seu programa é executado na velocidade normal ate atingir o ponto de interrupção. Atingindo o ponto, o depurador exibe o Code Editor com a linha contendo o ponto de interrupção, permitindo modificar o código ou exibir o valor de variáveis utilizando a janela de observação.
Para definir um ponto de interrupção, você pode:

- Dar um duplo-clique a esquerda da linha do código fonte onde quiser definir um ponto de interrupção.
- Selecionar Toggle Breakpoint utilizando o Speedmenu Code Editor.

Pontos de Interrupção no Code Editor

Quando definir um ponto de interrupção, a linha de código correspondente ao ponto é realçada e um ícone de sinal de parada aparece na margem esquerda como mostrado na figura a seguir:

Você também pode utilizar a página Editor Colors do quadro de dialogo Environment Options para definir uma cor diferente, indicando pontos de execução, linhas de pontos de interrupção inválidos, e a propriedade habilitado/desabilitado do ponto de interrupção.



Visualizando o Conteúdo de Variáveis

A janela Watches monitora a alteração do valor das variáveis ou expressões durante a execução de seu programa. Conforme seu programa seja executado, quer esteja pulando sobre o código, o conteúdo da janela de observação é alterado conforme os valores das variáveis contidas na expressão de observação.

Adicionando Expressões de Observação

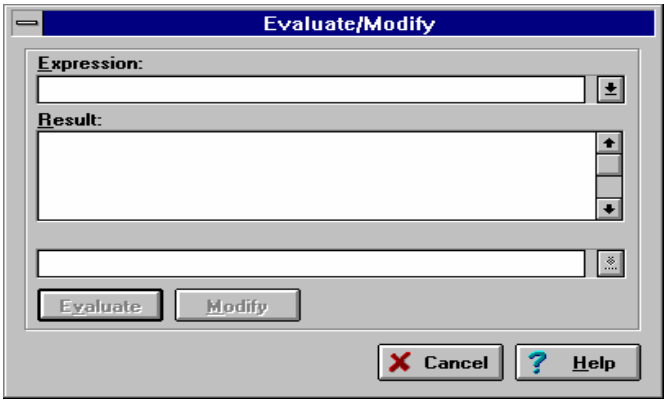
Para iniciar uma expressão de observação, selecione Add Watch no menu Run. O quadro de dialogo Watch Properties aparece, como segue:

Exibindo Expressões de Observação

A janela Watches exibe os valores atuais de expressões de observação e permite adicionar, editar, deletar, habilitar e desabilitar observações. Para exibir a janela Watch List, selecione Watches no menu View. A janela Watch List aparece.

Quadro de Diálogo Evaluate/Modify

No Delphi, você pode alterar os valores de variáveis e itens nas estruturas de dados durante a depuração. Para avaliar ou alterar o valor de uma expressão existente, utilize o quadro de dialogo Evaluate/Modify. Alterações feitas no quadro de dialogo Evaluate/Modify não afetam o código fonte ou o programa compilado. Para tornar as alterações permanentes, modifique seu código fonte e recompile seu programa. O quadro de dialogo Evaluate/Modify aparece como segue:



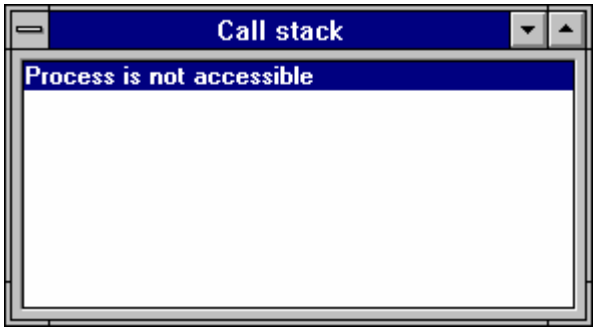
Visualizando o Call Stack

Quando estiver depurando, você pode achar útil a ordem das chamadas de funções ativas. A janela Call exibe o fluxo atual das chamadas de funções.

Janela Call Stack

A janela Call Stack exibe chamadas de funções ativas. Cada função aparece com uma lista de argumentos utilizadas na chamada. Você também pode utilizar a janela Call Stack para visualizar ou editar o código-

fonte associado com uma determinada chamada de função. Para visualizar a janela Call Stack, você pode selecionar View na barra de menu, e depois Call Stack. A janela Call Stack aparece como segue:



Lab: Modificando a Aplicação de Exemplo

Objetivos

Este Lab reforça sua habilidade em:

- Adicionar um componente ao form.
- Definir propriedades de um componente.
- Adicionar um event handler.
- Compilar um programa Delphi.

Cenário

Neste capítulo, você aprendeu a criar uma aplicação em um projeto chamado PSAMPLE. Em PSAMPLE, você adicionou os componentes ListBox, Button e Edit ao Form. O Evento OnClick do form adiciona itens digitados no componente Edit a list box. Neste lab, você vai melhorar esta aplicação, adicionando um botão Deletar e um Sair para esta aplicação.

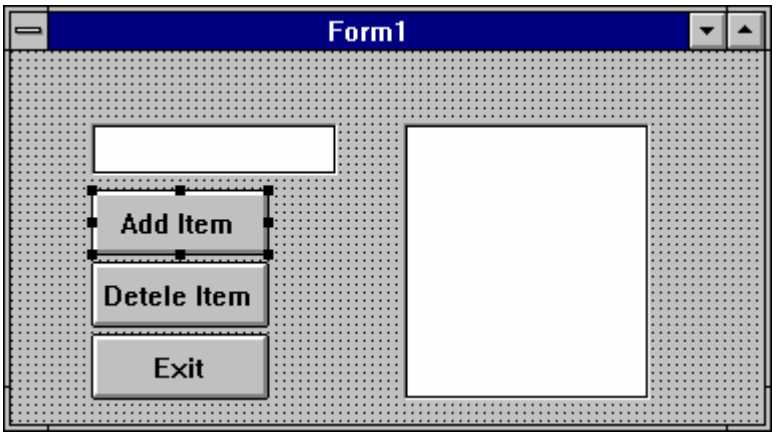
Processo

Utilize as seguintes diretivas para modificar a aplicação PSAMPLE:

Estágio	Processo
1	Abra a aplicação PSAMPLE criada neste capítulo e adicione dois botões ao form.
2	Defina as propriedades para estes botões para que um botão (Button2) tenha o Caption *Deleter Item* e o outro botão (Button3) tenha o Caption Exit e o primeiro com **Add Item**.
3	Adicione um event handler para o evento OnClick do botão com o Caption **Exit**. Utilize o método Close do TForm para sair da aplicação. A declaração para este evento segue abaixo: *Close;*
4	Adicione um event handler para o OnClick do botão com o Caption **Delete Item**. O event handler completo segue abaixo: *Procedure TForm.Button2Click(Sender: TObject);* *begin*

	with ListBox1 do *begin* *if ItemIndex<> -1 then* *Items.Delete(ItemIndex);* *end;* *end;*
5	Inclua também no event OnClick do botão com Caption Add Item o código abaixo: *ListBox1.Items.Add(Edit1.Text);*
6	Grave e execute a aplicação
7	Adicione quatro ou cinco itens à lista box
8	Delete itens da lista

Seu from deve estar similar a figura a seguir:



Processos Opcionais

O event handler para o evento OnClick do botão **Deletar Item** utiliza uma declaração de programação chamada de declaração **with**.
Utilize o sistema de help para determinar o que esta declaração faz. Revise o event handler Button2Click, habilitando o método para que você possa deletar a declaração with e as palavras-chave **begin...end** associadas.

Resumo do Capítulo

Pontos chave

Após completar este capítulo, você aprendeu:

- Que arquivos de projeto do Delphi controlam as aplicações construindo vários forms, executando as aplicações e exibindo o form principal da aplicação.
- Que adicionar um form cria dois arquivos:
  - Um arquivo de form com extensão .DFM contendo informações de resource do Windows e código Object Pascal do Delphi para a construção do form.
  - Um arquivo de unit com extensão .PAS contendo código Object Pascal.

- O Project Manager do Delphi lista os arquivos que compõem sua aplicação e permite navegar pelos arquivos.
- Que o Delphi possui um depurador totalmente integrado que permite depurar aplicações sem ter que deixar o ambiente de desenvolvimento visual.

Termos e definições

A tabela a seguir é uma referência rápida para os termos explicados neste capítulo.

Termo	Descrição
Breakpoint	Uma marcação em seu programa que causa uma pausa na execução durante o processo de depuração.
Event handler	Uma procedure que diz ao componente para que execute determinadas declarações do programa quando um evento em especial é detectado.
Project File	Um aquivo que controla uma aplicação Delphi construindo vários forms, executando a aplicação e exibindo o form principal da aplicação. O nome default é PROJECT1.DPR.
Watches	Expressões que permite monitorar o valor das variáveis ou expressões enquanto seu programa estiver sendo executado.

Capítulo 4

Estudo Detalhado de Componentes

Ao final do capítulo, você estará apto a:

- Explicar o que são componentes visuais e não-visuais
- Explicar o que são propriedades, eventos e métodos
- Descrever três maneiras de manipular ou personalizar componentes

Estudo Detalhado de Componentes

Overview

Componentes são blocos de construção de uma aplicação. Este capítulo cobre os principais aspectos de componentes: propriedades, eventos e métodos. O estudo dos componentes também inclui várias maneiras de arranjá-los, alinhá-lose definir propriedades de componentes.
Este capítulo também enfatiza a utilização do help on-line, pois é uma fonte extensiva sobre componentes.

Descrição de Um Componete

Introdução

Aplicações do Delphi são construídas utilizando-se componente. Um componente é um objeto que pode ser manipulado para construir e personalizar uma aplicação. Mesmo um form, embora não

seja encontrado na Component Palette, é um componente. Ele é um componente que pode conter outros componentes. Genericamente, o termo componente refere-se a itens encontrados na Component Palette. Este significado aplica-se no decorrer do curso.
Os componentes são categorizados em componente visuais e não-visuais.

Componentes Visuais

Componentes Visuais aparecem durante a execução da mesma forma como aparecem durante o design. Exemplos são botões e edit fields.

O Delphi utiliza o termo controle, como um significado para um componente visual que possa ser visualizado quando a aplicação estiver sendo executada. Os controles são divididos em dois grupos:
controles ajanelados e não ajanelados. Consulte o Apêndice A: Controles no Delphi, para mais detalhes.

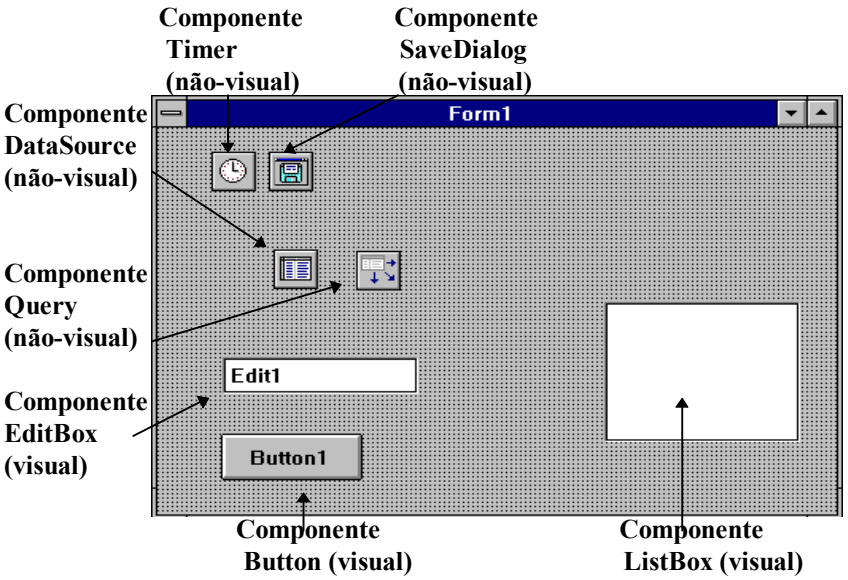
Componentes Não-visuais

Componentes não-visuais aparecem durante o design como pequenas figuras em um form. Entretanto, eles:

- Fazem com que um quadro de dialogo seja exibido quando chamado. Um exemplo é o componente SaveDialog.
- Não aparecem em momento nenhum durante a execução. Exemplos são os componentes Timer, DataSource, e Table.

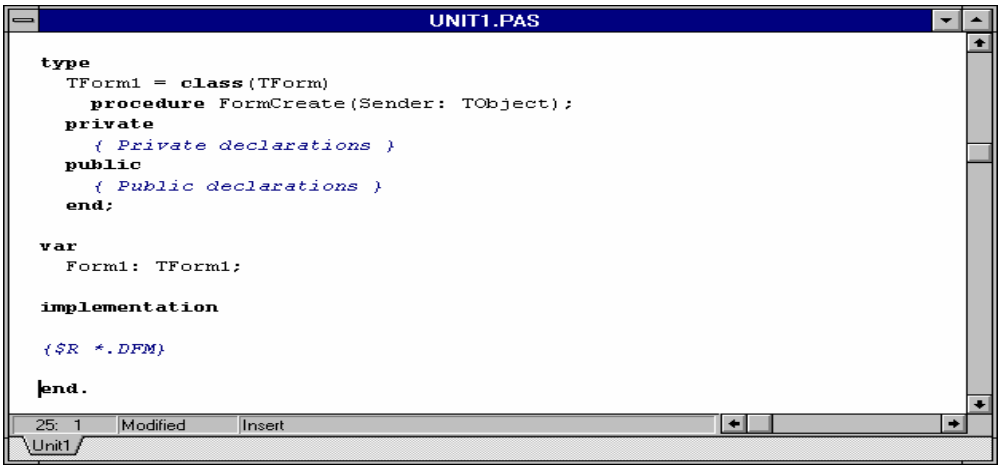
Exemplos de Componentes Não-Visuais

A figura a seguir mostra exemplos de componentes Visuais e não-visuais:



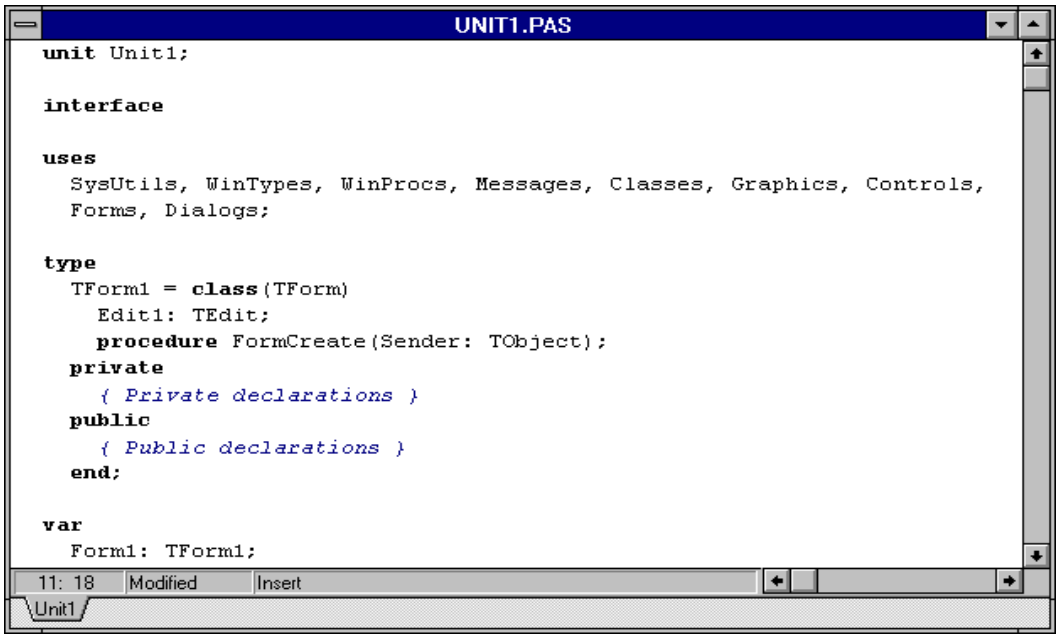
Código Gerado para um Form

Como mencionado, o Delphi permite criar uma aplicação com um mínimo de codificação. Quando você inicializa o Delphi, um novo form é automaticamente gerado e o Delphi cria um arquivo unit correspondente. O código correspondente ao form pode ser visualizado na janela do code editor. A janela code editor geralmente é posicionada atrás do form default, Form1. código é gerado quando você adiciona um componente é um evento, como você vera posteriormente neste capítulo. A janela do code editor aparece, como segue:



Código Gerado Adicionando um Componente

Quando você adiciona um componente ao form, o Delphi gera o código para isto, como mostrado no exemplo a seguir, com um componente Edit:



Aspectos de um Componente

Um componente Delphi possui os três seguintes aspectos:

- Propriedades
- Eventos

- Métodos

Cada Aspecto é explicado nas seções a seguir:

Propriedades

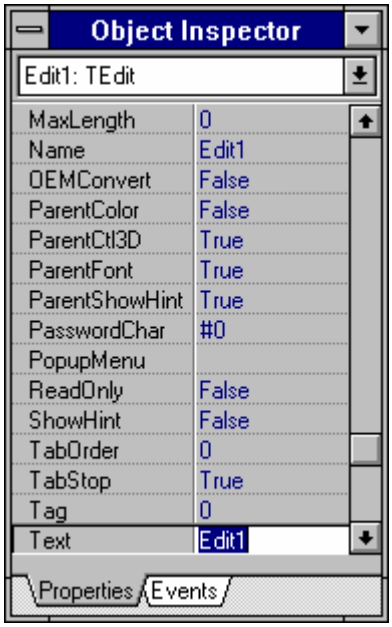
Propriedades são atributos ou campos de componente. Propriedades podem ser definidas durante a execução ou design. As propriedades controlam a forma como um componente se comporta e sua aparência em uma aplicação. Por exemplo, um botão é um componente que você pode adicionar a um form. Uma das muitas propriedades de um botão é a propriedade Caption. Definir a propriedade Caption altera o texto exibido no botão.

Influenciando o Comportamento de um Componente

O Object Inspector exibe as propriedades de um componente. O Object Inspector é a conexão entre a aparência visual do componente e o código que o Delphi gera para fazer com que a aplicação seja executada. O Object Inspector é utilizado para definir propriedades durante o design.

Exemplo:
Propriedade para o Component Edit

As propriedades são listadas na página Properties do Object Inspector. O exemplo a seguir mostra parte desta lista para o componente Edit adicionado ao form.



Exemplo:
Descrição das Propriedades para o Componente Edit

Muito embora as propriedades no Object Inspector variem para cada componente selecionado, uma vez que você entenda as propriedades de um componente, você pode aplicar este conhecimento para outros componentes. A tabela a seguir descreve a lista completa das propriedades do componente Edit. Esta lista é uma representação das propriedades comumente encontradas para os componentes do Delphi.

Propriedade	Descrição
AutoSelect	Determina se o texto é selecionado automaticamente quando o usuário mover-se ate o componente Edit ou Memo utilizando tab
AutoSize	Determina se um componente tem o tamanho reajustado automaticamente para acolher

	seu conteúdo
BorderStyle	Determina o tipo de borda exibido para um componente
CharCase	Determina se o estilo dos caracteres exibidos sera em maiúsculas, minúsculas ou misturadas
Color	Determina a seguir: • A cor de fundo de form • A cor de um controle ou figura
Ctl3D	Determina se o controle possui um visual tridimensional ou bidimensional
Cursor	Determina a imagem que o ponteiro do mouse assume (tal como seta ou I-beam) quando passar por área coberta pelo componente
DragCursor	Determina a imagem que o cursor assume (tal como seta ou I-beam) quando passar por sobre um componente que aceite arrasto
DragMode	Determina o componente drag-and-drop de um componente
Enabled	Determina se o componente responde a eventos do mouse, teclado ou timer
Font	Determina os atributos(cor, tamanho, estilo, ou nome) do seguinte: • Texto escrito sobre ou dentro de um componente ou objeto • Texto enviado a impressora
Height	Determina o tamanho vertical de um componente ou objeto
HelpContext	Determina um número único para cada tela a ser chamada no Help sensível ao contexto
HideSelection	Determina se o texto selecionado mantém-se selecionado quando o objeto perder o foco

Propriedade	Descrição
Hint	Determina a string de texto que aparecera quando o evento OnHint ocorrer (quando o cursor passar sobre um componente ou item de menu, seu significado sera exibido)
Left	Determina a localização horizontal em pixels do lado esquerdo de : • Um componente em relação ao form ou painel ou outros objetos container • Um form em relação a tela
MaxLength	Determina o número máximo de caracteres que um usuário pode digitar em um componente Edit ou Memo. Zero(0) significa sem limite.
Name	Determina um nome único para um componente ou objeto
OEMConvert	Determina se o texto é convertido para caracteres OEM
ParentColor	Determina onde o componente procurara pela informação sobre sua cor, como segue: • Se o valor form True, o componente utiliza a propriedade Color do componente pai. • Se o valor form False, o componente utiliza sua própria propriedade Color.
ParentCtl3D	Determina onde o componente procurara pela informação sobre seu visual tridimensional, como segue: • Se o valor form True, o componente utiliza a propriedade tridimensional do componente pai. • Se o valor form False, o componente utiliza sua própria propriedade tridimensional.
ParentFont	Determina onde um componente procurara para pela pela informação sobre seu fonte, como segue: • Se o valor form True, o componente utilizará a propriedade Font do componente pai. • Se o valor form False, o componente utilizará sua própria propriedade Font.
ParentShowHint	Determina onde um controle procurara se o Help hint deve ser exibido, como segue: • Se o valor form True, o controle utiliza a propriedade ShowHint do componente pai. • Se o valor form False, o controle utiliza sua própria propriedade ShowHint.
PassWordChar	Determina se um componente Edit ou Memo exibe caracteres especiais (ao invés do texto real) quando uma senha form digitada
PopupMenu	Identifica o nome o menu pop-up que aparecera quando um dos seguintes acontecer: • O usuário seleciona um componente e pressiona o botão direito do mouse. • O método PopUp de um menu pop-up é executado.
ReadOnly	Torna um componente read-only durante a execução, para que o usuário possa alterar o

	valor do campo ou do dataset
ShowHint	Determina se o Help está habilitado ou não, para a aplicação, como segue: • Se o valor form True, Help Hints está habilitado. • Se o valor form False, Help Hints está desabilitado.
TabOrder	Indica a posição do componente na ordem tab do container, a ordem na qual um componente recebe o foco quando a tecla tab é pressionada.
TabStop	Determina se um usuário pode pressionar tab até o componente
Tag	Cria um local disponível para armazenar valor integer como parte de um componente A propriedade Tag, embora não utilizada pelo Delphi, está disponível para necessidades especiais do usuário.
Text	Especifica a string de texto exibida em um componente ou outro objeto
Top	Determina o posicionamento vertical em pixels do canto superior esquerdo de: • Um componente em relação ao form, painel ou outro controle container • Um form em relação a tela
Visible	Determina se um componente aparece na tela
Width	Determina o tamanho horizontal do componente e outros objetos

Utilizando Editores de Propriedades para Definir ou Alterar uma Propriedade

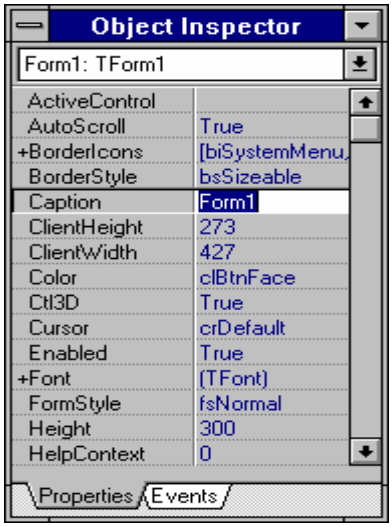
O Object Inspector oferece diversas maneiras de exibir propriedades e suas variações para que você possa defini-las ou alterá-las. Estes mecanismos são chamados de editores de propriedades.

Se você criar seu próprio componente, você pode utilizar os mesmos editores de propriedades para definir as propriedades de seu componente. Os editores são objetos do Delphi. Os editores são:

- Editor simples
- Editor drop-down list
- Editor com quadro de dialogo
- Editor com propriedades aninhadas

Editor Simples

O editor simples permite digitar um novo valor no lugar de um valor default para tipos numérico e strings alfanuméricas. No exemplo a seguir, o valor default da propriedade Caption é Form1. Você pode digitar um novo nome em seu lugar. O Delphi checa pela validade do valor para certificar-se de que uma string numérica não foi digitada no lugar de uma string alfanumérica ou vice-versa.



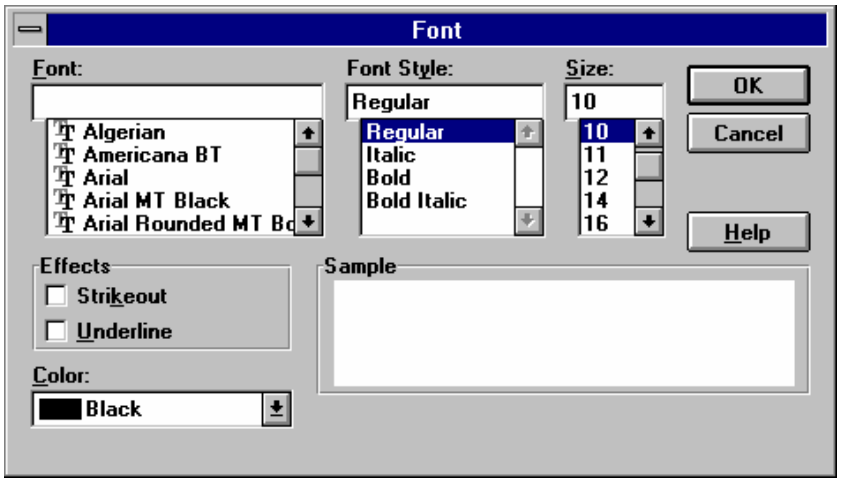
Editor Drop-Down List

Uma seta para baixo() após algum valor na coluna Values indica que um valor é limitado a uma lista de escolhas. Color e Cursor são exemplos de propriedades que possuem uma drop-down list.

Editor com Quadro de Diálogo

Reticências (...) indicam que existe um quadro de opções. O exemplo a seguir mostra um quadro de dialogo utilizado para definir diversos atributos para o tipo de objeto TFont. Font e Color são exemplos de propriedades que possuem um quadro de diálogo.

A propriedade Color não possui reticencias para indicar existência de um quadro de dialogo. Você deve dar um duplo- clique para exibi-lo.



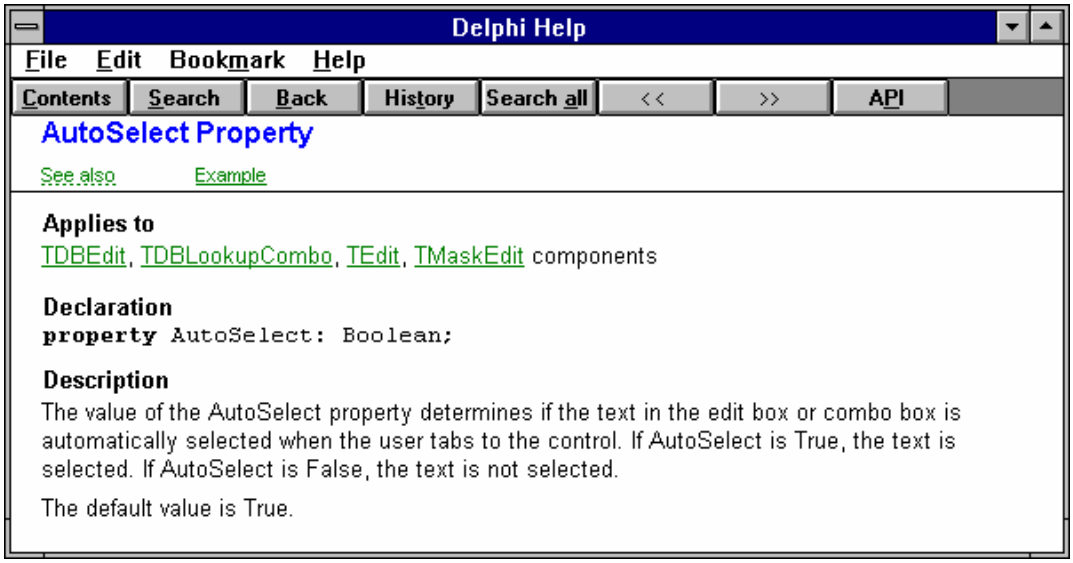
Editor com Propriedades Aninhadas

Algumas propriedades possuem menus aninhados (Submenus ocultos) de propriedades.Um sinal de mais (+) na frente do nome de tal propriedade indica que um ou mais menus aninhados de propriedades existem para aquela propriedade. Dando um clique sobre o sinal para exibir o submenu, um sinal de menos substitui o sinal de mais. Um sinal de menos (-) indica que não existe nenhum submenu adicional. Font é

um exemplo de propriedade que possui um menu aninhado. O sinal de mais no submenu +Style indica que a propriedade Style possui um menu aninhado.

Utilizando a Tecla F1 para Uma Descrição Completa de uma Propriedade

A importância de Help sensível ao contexto como característica do Delphi é aparente quando você precisa de uma descrição completa de uma propriedade. Pressionando a tecla F1 sobre a propriedade e seu(s) valor(es), como na propriedade AutoSelect, como segue:



Tópicos Relacionados Através da Tecla F1

Dando um clique sobre a opção See Also (em verde no texto) liga propriedade com tópicos relacionados. Uma lista de tópicos é exibida, como no exemplo a seguir, da propriedade AutoSelect:

Exemplos de Códigos Através da Tecla Fl

Dando um clique sobre a opção Example (em verde, no texto) liga a propriedade a um exemplo de código relacionado para propriedade AutoSelect.

O Help on-line inclui o comando Copy que permite copiar o texto do Help e cola-lo na aplicação. Isto pode ser especialmente útil se você encontrar exemplo de código que queira utilizar em sua aplicação. Para utilizar o comando Copy, exiba o tópico de Help que queira copiar, e no menu Edit, selecione Copy.

Passos para Definir Propriedades Durante o Design

Execute os passos para definir propriedades durante o design:

Passo	Ação
1	Dê um clique sobre uma página da Component Palette, e de um clique sobre o componente que você queira, como por exemplo o componente Edit.
2	De um clique sobre a área do form onde você queira inserir o componente. O componente aparece no form com o componente Edit. O nome do componente e o tipo do objeto Edit1: TEdit é colocado no Object Selector. A coluna Values exibe o nome Edit1 como o texto default da propriedade Text.
3	Utilize um editor de propriedade para alterar o valor da propriedade. As alterações feitas para a maioria das propriedades aparecem imediatamente no form.

Eventos

Introdução

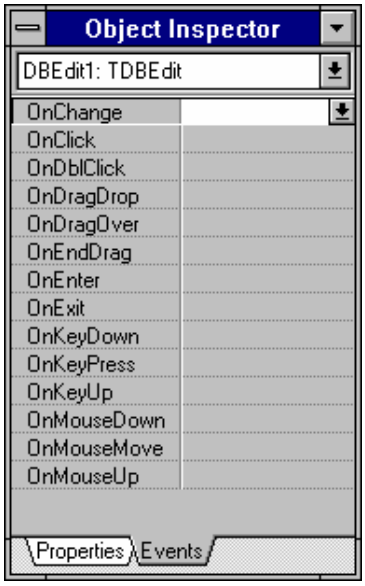
Evento é outra característica de um componente. Eventos são ações de usuários ou ocorrências do sistema que um componente possa reconhecer, tal como um clique de mouse.

Event Handler

Quando você seleciona um evento para um componente, o Delphi gera um heading de procedure e bloco de código. O código que você escreve especifica como um componente deve responder a um evento, e é chamado de event handler. Event handler são procedures especializadas. O Object Inspector permite especificar quais procedures estão associadas a determinados eventos.

Exemplo: Eventos para o Componente Edit

Os eventos estão listados na página Events do Object Inspector. O exemplo a seguir mostra uma lista de eventos para o componente Edit:



Exemplo : Descrição dos Eventos para o Componente Edit

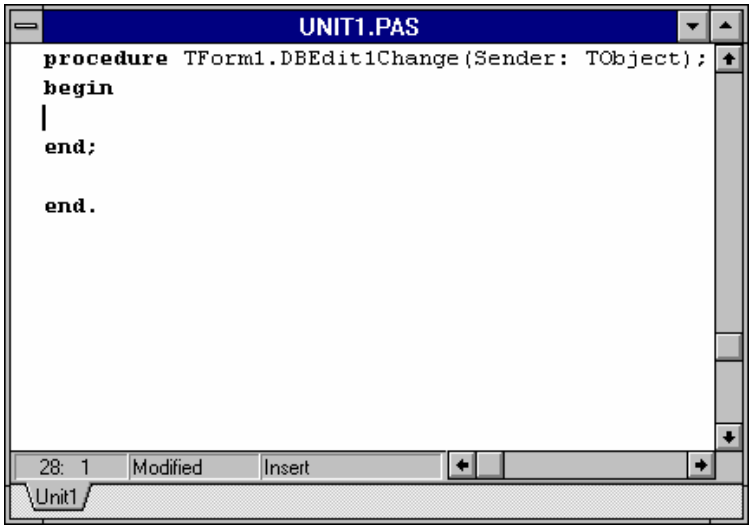
Muito embora os eventos no Object Inspector variem para cada tipo de componente, uma vez que você entenda os eventos para um componente, você pode aplicar este conhecimento para outros componentes.

A tabela a seguir lista os eventos e suas descrições para o componente Edit:

Evento	Descrição
OnChange	Ocorre quando um determinado valor de um objeto ou componente é alterado
OnClick	Ocorre quando o usuário da um clique sobre o componente
OnDblClick	Ocorre quando o usuário da um duplo-clique sobre um componente
OnDragDrop	Ocorre quando o usuário solta um objeto sendo arrastado
OnDragOver	Ocorre quando o usuário arrasta um objeto sobre um componente
OnEndDrag	Ocorre quando termina o arrasto de um objeto
OnEnter	Ocorre quando um componente torna-se ativo
OnExit	Ocorre quando o foco de input passa de um componente para outro
OnKeyDown	Ocorre quando o usuário pressiona qualquer tecla quando um componente possuir o foco
OnKeyPress	Ocorre quando o usuário pressiona uma única tecla de caractere
OnKeyUp	Ocorre quando o usuário solta uma tecla que estava pressionada
OnMouseDown	Ocorre quando o usuário pressiona o botão mouse enquanto o ponteiro do mouse estiver sobre um componente
OnMouseMove	Ocorre quando o usuário move o ponteiro do mouse quando o ponteiro do mouse estiver sobre o componente
OnMouseUp	Ocorre quando o usuário solta o botão do mouse quando o ponteiro do mouse estiver sobre um componente

Heading de de Procedure Gerado Através de um Duplo-Clique em um Evento

No Object Inspector, um duplo-clique na coluna Values próximo ao evento gera um heading de procedure para o código do evento e o associa ao evento, como mostrado no exemplo a seguir, no evento OnChange do componente Edit. O cursor é posicionado entre o par begin...end para que você possa digitar o código para o comportamento desejado.



Chamando Procedures com Parâmetros

Um event handler pode ter parâmetros. As instruções dentro da procedure podem utilizar os valores passados como parâmetros quando o programa é executado. Os valores são tratados como variáveis declaradas dentro da procedure para passagem de dados. Os parâmetros aparecem entre parêntesis após o nome da procedure, como mostrado na figura anterior.

Passos para Adicionar um Enevt Handler

Execute os passos a seguir para adicionar um event handler ao componente Edit:

Passo	Ação
1	Após modificar uma ou mais propriedades de um componente, de um clique sobre a aba da página Events para exibi-la. A página Events exibe todos os eventos reconhecidos pelo componente selecionado, como no exemplo anterior.
2	Selecione um evento e de um duplo-clique na coluna Values próximo ao evento. O Delphi gera um event handler (heading da procedure e o bloco de código) na janela do code editor. O cursor é posicionado dentro do par **begin...end**.
3	Dentro do par **begin...end**, digite as instruções que você quer que o Delphi execute quando o componente receber o evento. O Delphi adiciona uma instrução de procedure na porção interface do arquivo unit. Se você alterar o nome de event handler, o Delphi altera o nome onde quer que apareça dentro no código-fonte.

Métodos

Introdução

Alem de propriedades e eventos, um componente possui métodos. Como os componentes são objetos, eles herdam não somente propriedades e eventos, mas também métodos. Em geral, um método é uma procedure ou função associada a um componente. Nos termos de programação, um método é uma procedure ou função declarada como parte de um objeto.

Exemplo: Descrição dos Métodos de um Componente Edit

A tabela a seguir lista os métodos e suas descrições de um componente Edit:

Método	Descrição
BeginDrag	Uma procedure que inicia com o arrasto de um controle • Se o valor do parâmetro Immediate for True, o ponteiro do mouse altera para o valor da propriedade DragCursor e o arrasto inicia imediatamente. • Se o valor do parâmetro Immediate for False, o ponteiro do mouse não muda de valor e o arrasto inicia somente após o usuário mover o ponteiro do mouse cinco pixels.
BringToFront	Uma procedure que põe um componente ou form na frente de outros componentes ajanelados ou não, ou forms dentro de seu componente pai ou form.
ClassName	Uma função que retorna o nome da classe de um objeto
ClassParent	Uma função que retorna a classe que seja pai de um objeto
ClassType	Uma função que retorna o tipo de classe de um objeto
Create	Um construtor que inicializa um objeto ou componente de acordo com certas procedures padrão. Um construtor é declarado com a palavra reservada constructor.
Clear	Uma procedure que deleta itens ou texto de um controle
CleanSelection	Uma procedure que deleta o texto selecionado de um componente Edit ou Memo
CopyToClipboard	Uma procedure que copia uma seleção ao Clipboard
CutToClipboard	Uma procedure que deleta a seleção de um controle e a copia para o Clipboard
Destroy	Um destrutor que destrói um objeto, controle ou componente, e libera a memória que estava alocada a ele. Um destrutor é declarado com a palavra destructor
Dragging	Uma função que especifica se um objeto está sendo arrastado
EndDrag	Uma procedure que termina o arrasto de um objeto
Free	Uma procedure que destrói um objeto e libera a memória alocada para ele

GetSelTextBug	Uma função que copia para um buffer o texto selecionado de um componente Edit ou Memo, e retorna o número de caracteres copiados
GetTextLen	Uma função que retorna o tamanho do texto de um controle
Hide	Uma Procedure que opera somente durante a execução e torna um form ou controle invisível através da definição da propriedade Visible para False, do form ou controle
PasteFromClipboard	Uma procedure que copia texto do Clipboard para um controle
Refresh	Uma procedure que limpa qualquer imagem que esteja na tela e depois redesenha o controle inteiro
ScaleBy	Uma procedure que reajusta um controle a um percentual de seu tamanho anterior
ScrollBy	Uma procedure que rola o conteúdo de um form ou controle ajanelado
SelectAll	Uma procedure que seleciona o bloco de texto inteiro em um controle
SendToBack	Uma procedure que coloca um componente, ajanelado ou não, por tras de todos os outros componentes dentro de um componente pai ou form
SetBounds	Uma procedure que define as propriedades Left, Right, Height e Width de um componente aos valores passados com os parâmetros ALeft, ARight, AHeight e AWidth
SetFocus	Uma procedure que põe o foco de input no controle
SetSelTextBuf	Uma procedure que define texto selecionado em um componente Edit ou Memo ao texto na string null-terminated para o qual o buffer aponta
SetTextBuf	Define o texto em um componente para o texto do buffer
Show	Uma procedure que torna um form ou controle visual definindo sua propriedade Visible para True
Update	Uma procedure que redesenha um componente

Chamando Métodos

Você pode chamar métodos(funções ou procedures) utilizando a notação de ponto (.), conhecida como dot notation. Por exemplo para chamar um método show:

Edit.Show

Métodos, diferente de propriedades, nunca são ativados durante o design. Eles são procedures ou funções que operam no componente.

Utilizando a Tecla F1 para uma Descrição Completa dos Métodos

Pressionando **F1** sobre o nome de um determinado componente será mostrado uma lista dos métodos, bem como as propriedades e eventos associados àquele componente. Cada componente é listado pelo seu tipo de objeto. Você também pode selecionar **Topic Search** no menu **Help** para pesquisar qualquer componente.
Você pode querer pesquisar um compnente com o qual queira trabalhar para tornar-se familiar com todos os seus métodos disponíveis.

Manipulando e Personalizando Componentes

Introdução

Adicionar componente ao form é somente uma parte da tarefa em utiliza-los. Eles precisam ser manipulados e personalizados para melhor utilização e apresentação no form. As operações a seguir estão envolvidas :

- Definir propriedades comuns a múltiplos componentes
- Reajustar um componente
- Adicionar múltiplas cópias de um componente
- Agrupar componentes
- Alinhar componentes

Operações Adicionais

As operações sobre componentes a seguir não são cobertas no curso pois são idênticas no MS Windows:

- Deletar
- Mover
- Recortar e colar
- Copiar

Utilizando o Menu Edit

A maioria das opções para manipulação dos componentes estão no menu Edit.

Selecionando Um Componente e Vários Componentes

Quando um componente é selecionado, ele é realçado com pequenos quadrados negros ao redor de suas bordas. A seleção de componentes varia, se você quer selecionar um ou vários componentes. Você pode selecionar um componente das seguintes maneiras:

- Dando um clique sobre o componente no form
- Dando um clique sobre o nome do componente no Object Selector, na parte superior do Object Inspector
- Pressionando tab ate o componente no form

Você pode selecionar vários componentes das seguintes maneiras:

- Mantendo pressionada a tecla Shift conforme for dando cliques sobre os componentes
- Dando um clique em uma área em branco do form e arrastando o ponteiro do mouse pelo componente para inclui-los. Um retângulo aparecera conforme for arrastando o ponteiro do mouse.

Definindo Propriedades para Vários Componentes

No inicio do curso você aprendeu como definir uma propriedade para um componente. A maioria dos componentes possuem propriedades em comum, tais como as propriedades Heigth e Visible. Você pode definir propriedades em comum em um form sem ter que selecionar e alterar cada componente individualmente.

Passos para Definir Propriedades Compartilhadas

Siga os passos a seguir para definir propriedades em comum para vários componentes:

Passo	Ação
1	Selecione todos os componentes para os quais você queira definir propriedades em comum, dando um clique sobre o primeiro componente e, mantendo a tecla Shift pressionada, dando cliques em cada componente adicional. A página Properties do Object Inspector exibe somente as propriedades que todos os componentes possuem em comum.
2	Defina as propriedades que você queira compartilhar. Observe que todos os componentes adquirem a mesma definição da propriedade. Alterações visíveis no design são refletidas em cada componente.

Reajustando um Componente

Você pode reajustar componentes tanto quando os insere como após inseri-los. Quando você seleciona um componente, pequenos quadrados chamados manipuladores de reajuste aparecem em torno da borda do componente. Quando passar com o ponteiro do mouse sobre um manipulador, o ponteiro do mouse alterna para uma seta de duas pontas (). Quando o ponteiro do mouse estiver neste formato, você pode mover os manipuladores para tornar o componente maior ou menor. Se você reajustar o tamanho do componente conforme for adicionando, você não precisa utilizar os manipuladores.

Passos para Reajustar um Componente na Inserção

Execute os passos a seguir para reajustar um componente conforme for inserindo-o:

Passo	Ação
1	Selecione o componente na Component Palette.
2	Posicione o ponteiro do mouse no form onde queira que o componente apareça e arraste o mouse na direção desejada. Conforme for arrastado, um retângulo aparece para indicar o tamanho e posição do componente.
3	Quando o retângulo tiver o tamanho desejado, solte o botão do mouse. O componente aparece no mesmo tamanho que o retângulo.

Passos para Adicionar um Componente e Reajustá-lo

Execute os passos a seguir para adicionar um componente e reajustá-lo.

Para um reajuste em pixeis, você pode utilizar **Size** no menu **Edit**.

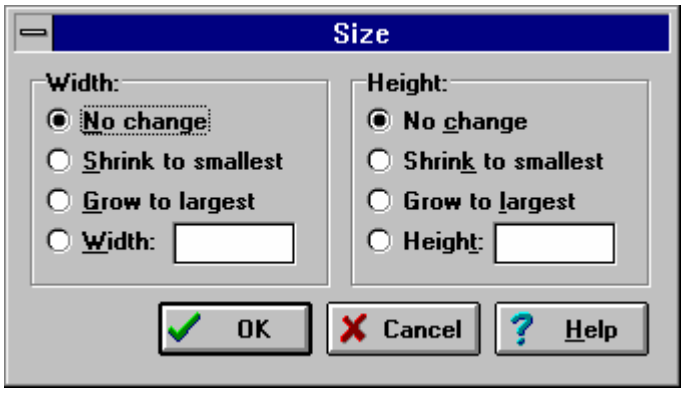
Passo	Ação
1	Selecione um componente na Component Palette.
2	Aponte para a área do form onde queira que o componente apareça e de um clique com o botão esquerdo do mouse. O componente aparece no form.
3	Passe com o ponteiro do mouse por sobre os manipuladores. Quando o ponteiro do mouse alternar para uma seta de duas pontas, arraste os manipuladores para reajustar o tamanho.

Passos para Reajustar Vários Componentes

Execute os passos a seguir para reajustar vários componentes ao mesmo tempo.

Passo	Ação
1	Selecione o primeiro componente.
2	Mantenha pressionada a tecla Shift e de um clique sobre o resto dos componentes que queira reajustar. Todos os componentes selecionados aparecem realçados.

3	No menu Edit, selecione Size. O quadro de dialogo aparece, como segue:
4	Selecione as opções de reajuste desejadas. Para um reajuste preciso, você pode digitar um número em pixels nos campos **Width** e **Height**.
5	Dê um clique em **Ok**.



Adicionando Várias Cópias de Um Componente

Você pode adicionar várias cópias do mesmo tipo de componente pressionando a tecla Shift quando selecionar o componente na Component Palette. Quando você pressiona a tecla Shift e da um clique sobre o componente para a primeira cópia, um retângulo aparece ao redor do componente, como mostrado no exemplo com o componente Edit:
Quando der um clique em uma área do form, a primeira cópia do componente aparecera. Cada clique no form, após isto, adiciona uma cópia ao form. Dando um clique sobre a ferramenta de seleção cursor (o primeiro botão da Component Palette) termina o modo de múltiplas cópias.

Alinhando Componentes

Você pode alinhar componentes através da barra de menus de três maneiras:

- No menu **Views**, selecione **Alignment Palette**
- No menu **Edit**, selecione **Align**
- No menu **Options**, selecione **Environment**

Environment no menu **Options** permite alterar opções default no ambiente do Delphi.

Você também pode alinhar componentes utilizando a grade do form. A grade é uma característica default e aparece como linhas de pontos do form.

Passos para Alinhar Componentes Utilizando a View/Alignment Palette

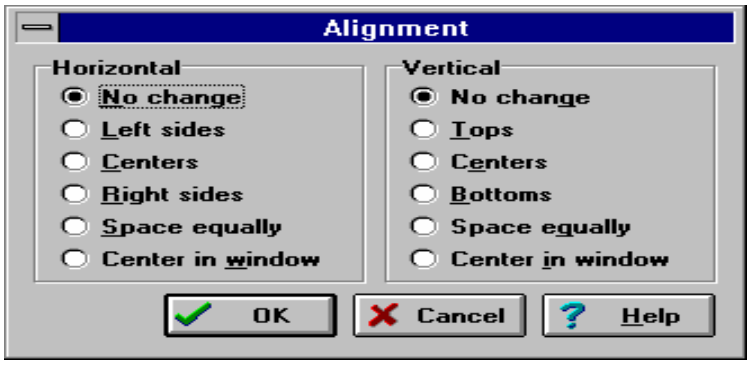
Execute os passos a seguir para alinhar componentes através do menu View selecionando Alignment Palette.

Passo	Ação
1	Selecione os componentes que queira alinhar.
2	No menu View, selecione Alignment Palette. A **Align Palette** aparece, como segue:
3	Passe com o ponteiro do mouse sobre cada botão para exibir seu significado.
4	Dê um clique em um botão de alinhamento.

Passos para Alinhar Componentes Utilizando Edit/Align

Execute os passos a seguir para alinhar componentes utilizando o menu Edit e selecionando Align.

Passo	Ação
1	Selecione os componentes que queira alinhar.
2	No menu Edit, selecione Align. O quadro de dialogo Alignment aparece como segue:
3	Dê um clique sobre uma opção de alinhamento e de um clique em OK.

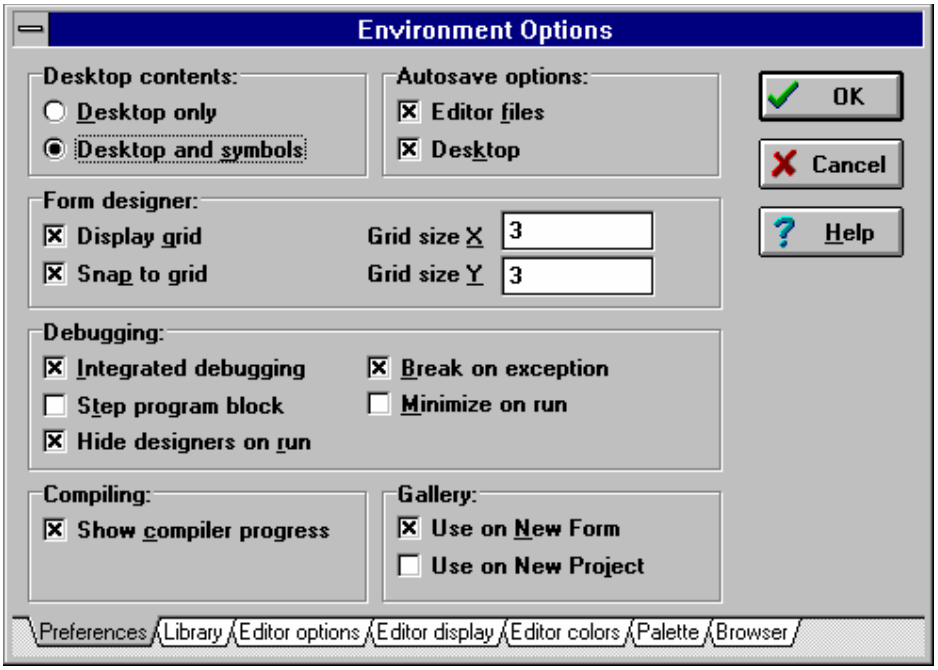


Passos para Alinhar Componentes Utilizando Options/Environment

Execute os passos a seguir para alinhar componente utilizando o menu Options e selecionando Environment.

Passo	Ação
1	No menu **Options**, selecione **Environment**. O quadro de dialogo Environment Options aparece, como segue, com a página Preferences na frente:
2	No GroupBox Form designer, certifique-se de que os seguintes estão habilitados: • **Display grid** Torna as linhas de pontos visíveis no form, para alinhamento dos componentes • **Snap to grid** Faz com que o canto superior esquerdo dos componentes sejam alinhados com a marca de grade mais próxima
3	Para alterar a granularidade (a distancia dos pontos da grade), digite novos valores no lugar dos exibidos. O valor default é 8 pixels para o espaçamento X (horizontal) e Y (vertical). Um número maior que 8 aumenta a distancia dos pontos, é um número menor que 8 aproxima os pontos.
4	De um clique em **OK**

5	Posicione os componentes no form alinhando-os com a grade.



Resumo do capítulo

Pontos Chave

Após completar este capítulo, você aprendeu que:

- Componentes visuais geralmente aparecem durante a execução da mesma forma que aparece durante o design.
- Componentes não visuais aparecem como pequenas figuras no form, mas não aparecem durante a execução a menos que exibam um quadro de dialogo.
- Um componente possui três aspectos: propriedades, eventos e métodos.
- Uma propriedade é um atributo de um componente. As propriedades controlam a forma como um componente é exibido e se comporta em uma aplicação.
- Um evento é uma ação do usuário ou uma ocorrência do sistema que sua aplicação pode reconhecer, tal como um clique de mouse.
- Um método é uma procedure ou função declarada como parte de um objeto.
- Manipulação e personalização de componentes envolvem várias operações, tais como definir suas propriedades ou reajustar os componentes.

Termos e Definições

A tabela a seguir é uma referência rápida aos termos mostrados neste capítulo:

Termo	Definição
Event handler	Uma procedure associada a um evento do componente que indica como manipular o evento
Evento	Uma ação do usuário ou ocorrência do sistema que um componente pode reconhecer
Função	Uma subrotina que retorna um valor
Grade	As linhas e colunas de pontos que servem como guias no alinhamento de componentes

	em um form
Instrução	A unit mais simples em um programa. Instruções são separadas por um ponto-e-virgula (;).
Método	Uma procedure ou função declarada dentro do objeto, componente ou controle.
Procedure	Um subprograma
Propriedades Compartilhadas	Propriedades comuns a todos os componentes selecionados
Rotina	Uma procedure ou função

Capitulo 5

Utilizando Forms

Ao final deste capítulo, você estará apto a :

- Adicionar um form do template ao projeto
- Gravar um form como um template
- Definir a ordem de tab
- Adicionar um menu ao form

Utilizando Forms

Overview

No Delphi forms são pontos focais da aplicação que você está desenvolvendo.Um form em branco é bem parecido com uma tela de pintura onde você pode adicionar componente, desenvolvendo a interface de usuário. Este capítulo cobre tópicos como a utilização de templates de forms, quadros de dialogo e menus.

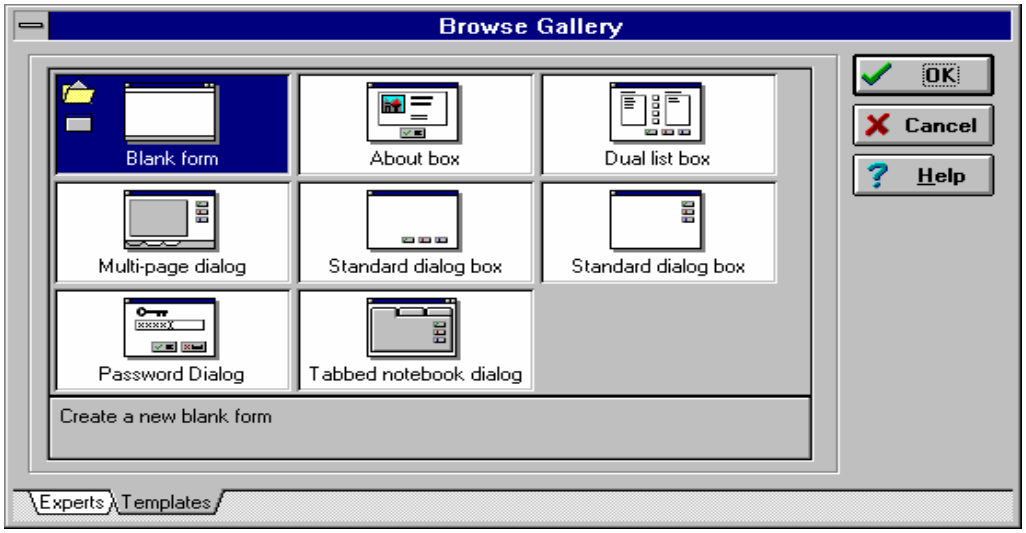
Utilizando um Template de Form

Introdução

Quando estiver construindo a interface de usuário de sua aplicação, você pode acelerar o desenvolvimento através da seleção de diversos templates de forms fornecidos com o Delphi. Um template é um form predefinido. Os templates incluídos com o Delphi fornecem um modelo para forms utilizados com maior freqüência, por exemplo, um quadro de dialogo padrão o um quadro Sobre que fornece informações sobre uma aplicação. Os templates de form disponíveis são exibidos na Forms Gallery.

Adicionando um Template de Form a um Projeto

Para utilizar os templates na Forms Gallery, habilite a opção Gallery no quadro de dialogo Environment Options. Para acessar este quadro, no menu **Options**, selecione **Environment** e depois a página Preferência. Quando a opção **Use on New Project** estiver habilitada, o quadro de dialogo Forms Gallery aparece quando você selecionar **New Form** no menu **File**. O quadro de dialogo Forms Gallery aparece, como segue:



Passos para Adicionar um Template a um Projeto

Para adicionar um template de form ao projeto, primeiro você deve ter um projeto aberto. Uma vez aberto o projeto, execute os passos a seguir para adicionar um template ao projeto:

Passo	Ação
1	No menu File, selecione New Form. O quadro de dialogo Browse Gallery aparece.
2	Na página Template, selecione o template apropriado.
3	De um clique em OK. Uma cópia do form e arquivos unit associados são adicionados ao seu project.

Tutorial: Adicionando um Quadro Sobre...

Introdução

Neste Tutoria, você utilizará a Forms Gallery para adicionar um template de form ao seu projeto. A Forms Gallery contém diversos templates de form para os tipos mais comuns. Um template utilizado na maioria das aplicações é o quadro Sobre... O quadro Sobre... é uma característica padrão que os desenvolvedores adicionam a maioria dos programas Windows.

Estágios do Tutorial

Este tutorial envolve os seguintes estágios:

Estágio	Processo
1	Adicione um form Sobre...utilizando o template About.
2	Modificar o design do form.
3	Exibir o form Sobre...

Os templates de form efetuam o primeiro estágio e parte do segundo por você.

Passos para o Estágio 1

Execute os passos a seguir para adicionar um quadro Sobre utilizando o template de form AboutBox.

Passo	Ação

1	Abra um novo projeto e grave o arquivo de unit como UABOUT.PAS é o arquivo de projeto como PABOUT.DPR.
2	No menu **File**, selecione **New Form**. O quadro de dialogo Browse Gallery aparece. Se a Browse Gallery não aparecer, você seleciona no menu **Options**, **Environment**. Localize a página Preference, e no grupo Gallery, habilite **Use on New Form**.
3	Selecione o template **About Box** na Gallery, e de clique em **OK**. Uma nova unit é um novo form são adicionados ao projeto.

Passos para o Estágio 2

Execute os passos a seguir para modificar o design do form AboutBox:

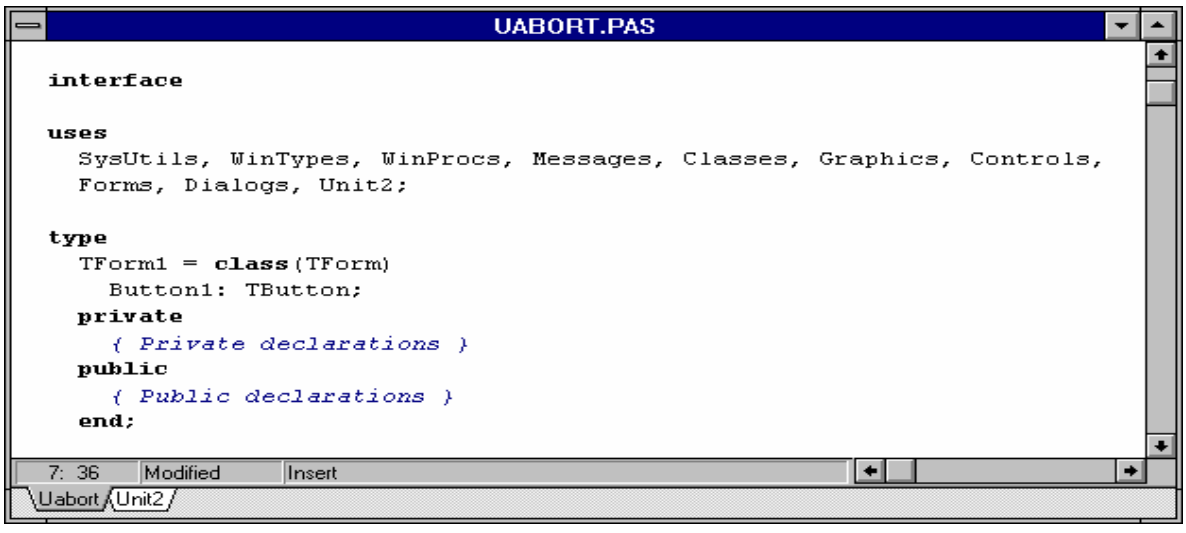
Passo	
1	Utilize os valores a seguir para a propriedade Caption dos componentes Label no Object Inspector.

Componente	Propriedade	Valor
ProductName	Caption	Exemplo de Quadro Sobre a Aplicação
Version	Caption	Versão 1.0
CopyRight	Caption	Copyright 1996
Comments	Caption	Tutorial de Quadro Sobre Aplicação

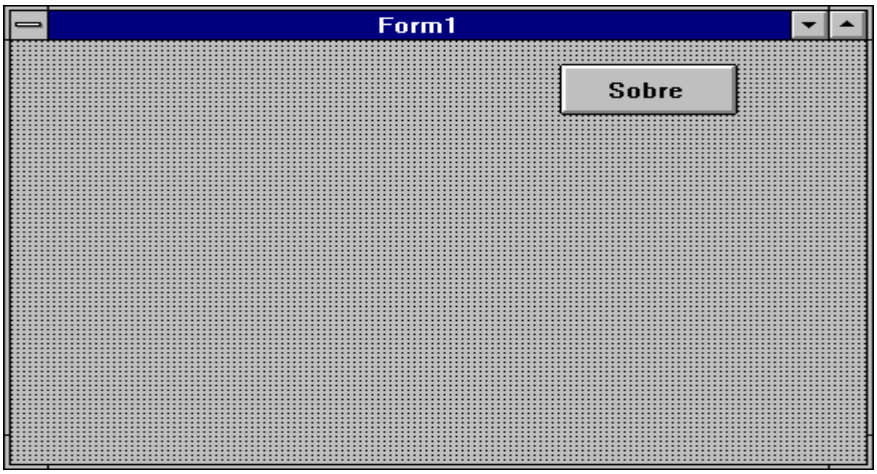
Passos para o Estágio 3

Para exibir o form, o primeiro passo é adicionar o identificador da unit do quadro Sobre... à cláusula uses da unit que irá exibir o quadro. Depois, utilize o método ShowModal ou Show em um event handler para exibir o form.

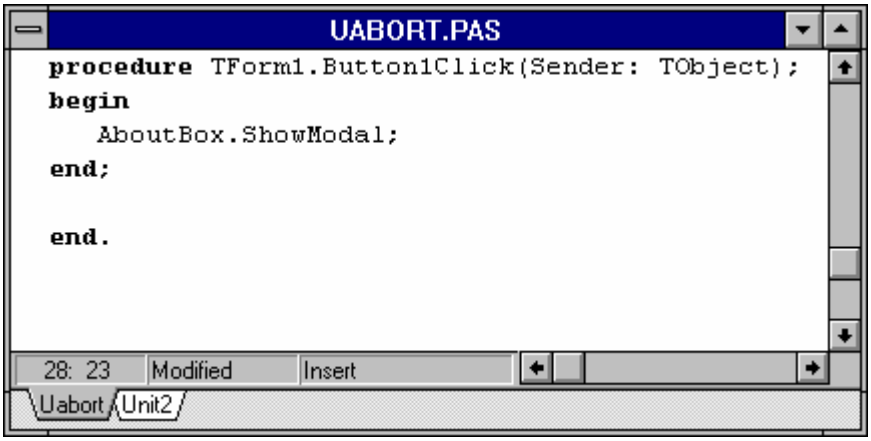
Passo	Ação
1	Adicione o identificador da Unit, Unit2, a cláusula uses da unit about, como mostrado na figura a seguir:



Passo	Ação
2	Adicione um botão ao form principal (Form1). Rotule este botão, como Sobre como na figura a seguir.



Passo	Ação
3	Crie um event handler para o evento OnClick do botão Sobre.
4	No event handler, digite a seguinte instrução: AboutBox.ShowModal; O event handler parecera com o seguinte:



Passo	Ação
3	Compile e grave sua aplicação
4	Execute e teste a aplicação

Gravando um Form como Template

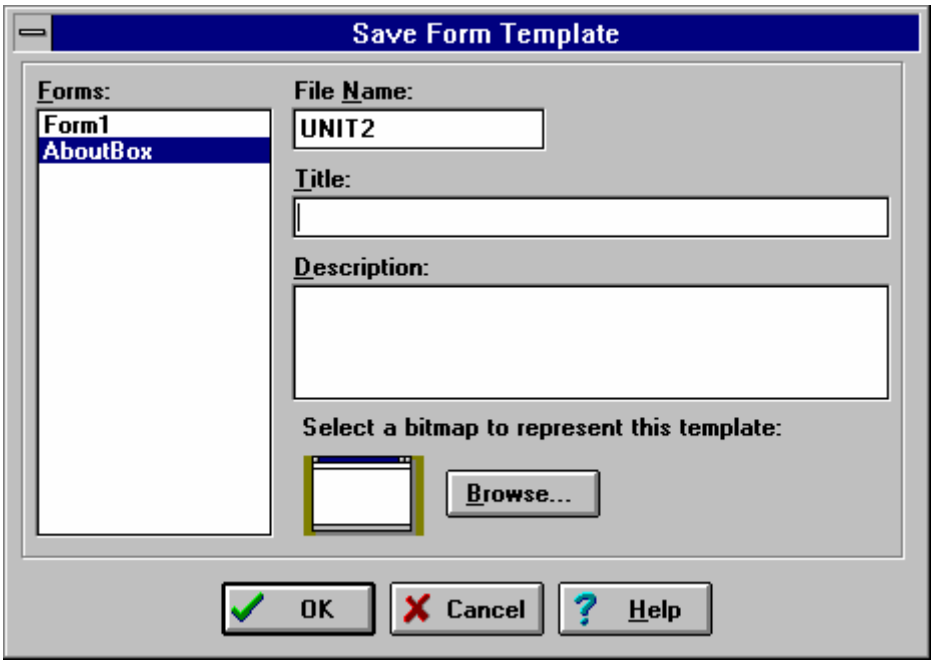
Introdução

Você pode gravar um form feito por você como um template. Estes templates estão disponíveis para serem utilizados em outros projetos. Gravar um form como um template insere o form no quadro de dialogo Browse Gallery. Para gravar um form como template, utilize o SpeedMenu do form e selecione Save As template. De um nome, descrição e icone ao novo template. Se você quiser especificar um icone para o template, você pode selecionar um bitmap existente ou criar um próprio.

Passos para Gravar um Form como um Template

Execute os passos a seguir para gravar um form como um template:

Passo	Ação
1	Dê um clique com o botão direito do mouse sobre o form para acessar o SpeedMenu.
2	Selecione Save **As Template**. O quadro de dialogo Save Form Template aparece, como segue:
3	No campo **Title**, digite um nome para o novo template.
4	No campo **Description**, digite uma descrição significativa do template.
5	Para associar um icone ao template, de um clique no botão **Browse**.
6	Localize e selecione o bitmap para associa-lo ao template, e de um clique no **OK**.
7	No quadro de dialogo Save As Form Template, de um clique em **OK** para gravar a informação e saia.



Criando Quadros de Dialogo

Introdução

O Delphi permite criar um número ilimitado de quadros de dialogo, do mais simples ao mais complexo. Este tópico cobre a criação de quadros de dialogo simples, tais como quadros de mensagem e de entrada de dados, bem como criar um quadro de dialogo com um form em branco.

Quadro de Mensagem

Um quadro de mensagem é uma janela que exibe text. Você pode utilizá-la para:

- Exibir informação
- Exibir uma mensagem de erro ou aviso
- Solicitar a confirmação de uma ação

Por exemplo, se o usuário tentar a aplicação antes de gravá-la, um quadro de mensagem deveria aparecer solicitando informando o usuário que os dados devem ser gravados antes de fechar. Para que a aplicação continue, o usuário deve remover o quadro de mensagem da tela manualmente, geralmente com o clique de um botão.
Para exibir um quadro de mensagem simples, utilize a função MessageDlg. O exemplo a seguir mostra a sintaxe da função:

Function MessageDlg (const Msg: String; AType: TMsgDlgType;
AButtons: TMsgDlgButtons; HelpCtx: LongInt): Word;

Explicação dos Parâmetros de MesageDlg

A tabela a seguir explica os parâmetros utilizados na sintaxe da função MessageDlg:

Parâmetro	Explicação
Msg	A mensagem que você quer exibir (uma constante string)
AType	Determina o tipo de quadro de mensagem que ira aparecer. Os valores possíveis são: • *mtWarning* Exibe um quadro de mensagem com um ponto de exclamação em amarelo(!) • *mtError* Exibe um quadro de mensagem com uma não vermelha • *mtInformation* Exibe um quadro de mensagem com um símbolo de informação em azul (i) • *mtConfirmation* Exibe um quadro de dialogo com um ponto de interrogação em verde (?)
AButtons	Determina quais botões aparecerão no quadro de mensagem. AButtons é do tipo TMsgDlgBtns que é um tipo set. Você pode incluir diversos botões dentro do set. O set pode incluir os seguintes valores: • mbYes Exibe um botão com um check mark em verde e o caption Yes • mbNo Exibe um botão com um circulo e uma barra por dentro, em vermelho e o caption No • mbOK Exibe um botão com um check mark em verde e o caption OK • mbCancel Exibe um botão com um X em vermelho e o caption Cancel • mbHelp Exibe um botão com um ponto de interrogação e o caption Help • mbYesNoCancel Exibe um conjunto que inclui os botões Yes,No, Cancel • mbOKCancel Exibe um conjunto que inclui os botões OK e Cancel
HelpCtx	Determina qual tela de Help está disponível para o quadro de mensagem. Um número de contexto para um Help sensível ao contexto.

Valores de Retorno de MessageDlg

Os valores de retorno para a função MessageDlg indicam o tipo de botões clicados pelo usuário. O valor de retorno pode ser um dos seguintes:

- mrNone
- mrOK
- mrCancel
- mrAbort

- mrRetry
- mrIgnore
- mrYes
- mrNo
- mrAll

Exemplos de Código Utilizando MessageDlg

O código de exemplo a seguir ilustra o uso da função MessageDlg:

```
Res := MessageDlg( ' Sair da Aplicação ??' , mtConfirmation, [ mbOk, mbCancel ], 0 );
If Res = mbOk Then
   Close;
```

Quadros Simples de Entrada de Dados

Um quadro simples de entrada de dados requer uma linha de informação do usuário. Um exemplo de um destes quadros é um que solicite uma senha. Para exibir um quadro de entrada de dados, utilize as funções InputBox ou InputQuery. A sintaxe para cada função aparece, como segue:

```
function InputBox(const ACaption, APrompt,ADfault: string): string;
function InputQuery(const Acaption, Aprompt: string; var Value: string): Boolean;
```

Explicação dos Parâmetros de InputBox e InputQuery

A tabela a seguir explica os parâmetros das funções InputBox e InputQuery:

Parâmetros	Explicação
ACaption	Determina o caption do quadro de diálogo
APrompt	Determima o texto de solicitação ao usuário
ADefault	Determina a string exibida quando o quadro aparecer
Value	Determina a string exibida quando o quadro aparecer, contém a string digitada pelo usuário quando **OK** for clicado. Utilizado por InputQuery.

Valores de Retorno de InputBox

A função InputBox exibe um quadro de entrada e aguarda que o usuário digite um valor. O valor de retorno de InputBox é uma string. Utilize a função InputBox quando não importa se o usuário escolher entre o botão OK ou Cancelar para deixar o quadro de dialogo. A função InputQuery exibe um quadro de entrada e aguarda que o usuário digite um valor. InputQuery retorna True se o usuário escolher OK e False se escolher **Cancelar**.

Forms MDI e SDI

Uma aplicação Multiple Document Interface (MDI) é uma aplicação com uma interface de janela principal, ou janela pai, que permite a abertura de diversos documentos, ou janelas filhas.
Uma aplicação Single Document Interface (SDI) pode processar mais de um documento dentro de uma janela pai.
Você pode implementar qualquer um de seus forms como MDI ou SDI. Embora você possa criar um form MDI ou SDI alterando a propriedade FormStyle de um form existente, é mais provável que você crie um form MDI ou SDI utilizando a Gallery.

Propriedades dos Quadros de Diálogo

Quando você cria um form no Delphi, ele possui as seguintes características:

- Botões de Maximizar e Minimizar
- Menu de Controle

- Borda de Reajuste

Estas características são úteis para quadros de dialogo não-modais, mas desnecessárias para quadros modais. Um quadro de dialogo não-modal é um quadro de onde o usuário pode remover o foco sem fecha-lo. Um quadro de dialogo modal requer que o usuário feche o quadro antes de continuar o trabalho na aplicação. Você pode definir a propriedade BorderStyle para bsDialog. Esta definição possui as seguintes características:

- Remover os botões de Maximizar e Minimizar
- Fornecer um menu Controle com as opções **Mover** e **Fechar**
- Tornar a borda do form não-reajustavel com aparência tridimensiaonal

Adicionando Botões de Comando

Se seu quadro de diálogo será utilizado em estado modal, você deve oferecer botões de comando dentro do quadro. Por exemplo, você deve oferecer um botão Sair que permite que o usuário saia do quadro de dialogo. Os botões de comando mais comuns incluem os seguintes:

- Um botão Cancelar ou Não que saia do quadro de dialogo sem gravar as alterações
- Um botão OK ou Sim que grave as alterações e depois saia do quadro de dialogo
- Um botão que tenha o foco por default para que o usuário possa pressionar a tecla Enter para aceitar o botão default

Você pode utilizar o componente BitBtn da página Additional para criar vários destes botões de comando. O componente BitBtn permite que você utilize os bitmaps padrões da Borland, tais como um check mark em verde () para um botão OK. Você pode selecionar o bitmap e caption padrão que aparecem no botão definindo a propriedade Kind.

Definindo a Ordem de Tab

Em uma aplicação sendo executada, a ordem de Tab é a seqüencia no qual o usuário pode mover-se entre os componentes pressionando a tecla Tab. Para permitir que os usuários utilizem a tecla Tab para mover-se pelos componentes, você deve definir a propriedade TabStop de cada componente para True. Por default, a ordem de Tab é definida de acordo com a ordem na qual você inseriu os componentes no form. Você pode alterar a ordem de Tab default alterando a propriedade TabOrder do componente no Object Inspector ou utilizar o quadro de dialogo Edit Tab Order.

Passos para Definir a Ordem de Tab

Execute os passos a seguir para definir a ordem de Tab utilizando o quadro de dialogo Tab Order:

Passo	Ação
1	Selecione o form contendo os componentes cuja ordem de tab você queira definir.
2	No menu **Edit**, selecione Tab Order. O quadro de dialogo Edit Tab Order aparece, listando os componentes na ordem atual, como no exemplo a seguir:
3	No campo **Controls listed in tab order**, selecione o controle (componente).
4	Utilize os botões com as setas para cima e para baixo para reordenar a ordem dos componentes. Você também pode arrastar os componentes.
5	Quando tiver terminado, de um clique em **OK**.

Definindo o Foco em Quadro de Diálogo Durante o Design

Em um quadro de dialogo, somente um componente por vez pode ter o foco. O foco pode ser definido durante o design ou durante a execução. Durante o design, você pode definir a propriedade ActiveControl do quadro de dialogo abrir. Se você não especificar a propriedade ActiveControl, o primeiro componente da ordem de Tab receberá o foco. Entretanto, isto não se aplica nas seguintes situações:

- O componente está desabilitado
- O componente não é visível durante a execução
- A propriedade TabStop do componente está definida para **False**.

Definindo o Foco em um Quadro de Diálogo Durante a Execução

Durante a execução, o usuário pode alterar o foco dos componentes automaticamente, utilizando a tecla Tab. Entretanto, você pode querer especificar que o foco seja alterado de campo para campo utilizando as teclas de seta. Você pode utilizar o método SetFocus para especificar quais componentes recebem o foco. Para alterar o componente ativo durante a execução, digite o seguinte código no event handler apropriado:

<componente>.SetFocus;

Por exemplo, a instrução a seguir especifica que Button2 é o componente ativo durante a execução:

Button2.SetFocus;

Criando Menus

Introdução

Em uma aplicação, os menus oferecem uma maneira dos usuários executarem comandos. O Delphi fornece dois tipos de componentes menu: MainMenu e PopupMenu. Durante o processo de especificação de um menu, você utiliza um recurso dos componentes TMainMenu e TPopupMenu chamado Menu Designer. O Menu Designer permite criar um menu ou adicionar um menu já pronto ao seu form. Você também pode utilizar o Menu Designer para deletar, editar ou reordenar itens do menu.
Quando utilizar o Menu Designer para criar um menu, você pode visualizar seu menu como ele aparecera na
execução sem ter que realmente executá-la.

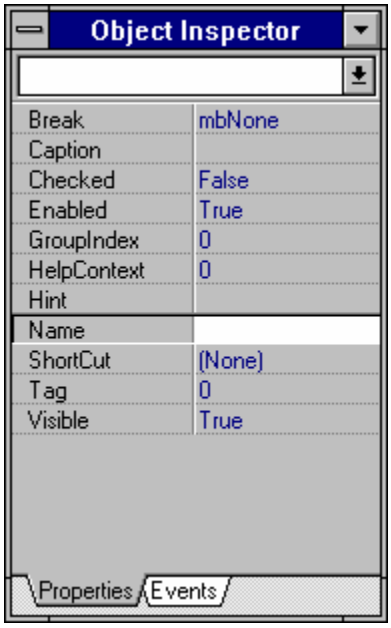
Adicionando um Menu

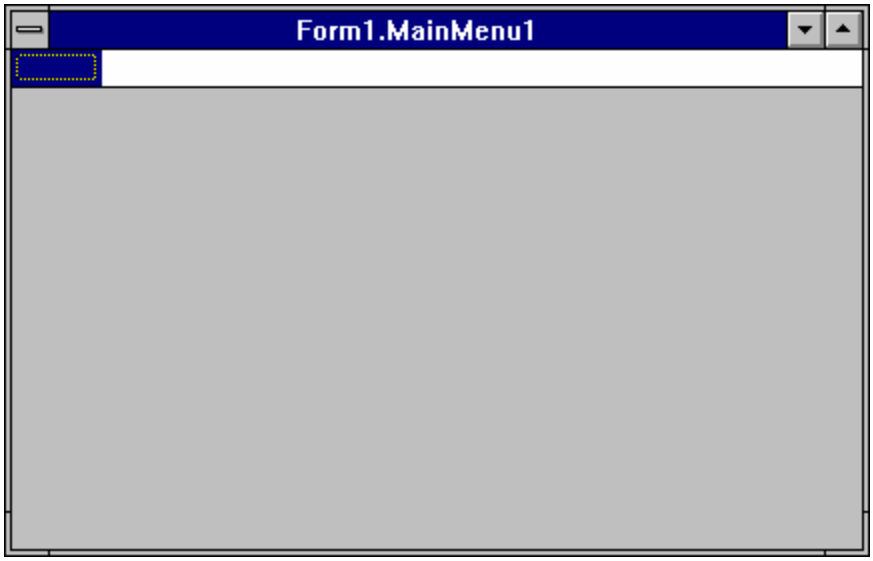
Para adicionar um menu a um form, utilize o Menu Designer. O Menu Designer permite definir um menu ou adicionar um menu personalizado ao seu form.

Passos para Adicionar um Menu

Execute os passos a seguir para adicionar um menu ao seu form durante o design:

Passo	Ação
1	Na página Standard da Component Palette, de um duplo-Clique no componente MainMenu ou PopupMenu. O componente é adicionado ao seu form.
2	Dê um duplo-clique no componente menu para abrir o Menu Designer. O Menu Designer e o Object Inspector aparecem, como segue. A propriedade Name no Object Inspector e selecionada.

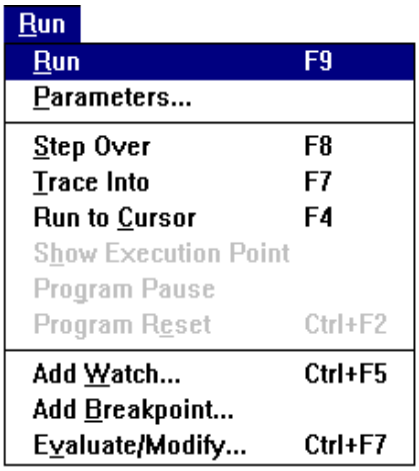




Passo	Ação
3	Digite o texto para o primeiro item da barra de menus na propriedade Caption do Object Inspector.
4	Pressione **Enter**. O primeiro item da barra de menu aparece na barra do form.
5	Para adicionar itens ao menu, digite o primeiro item na propriedade Caption.
6	Pressione **Enter** . O nome do item de menu aparece endentado abaixo do item da barra de menus.
7	Repita os passos 5 e 6 ate que você tenha adicionado todos os itens de menu que você queira para seu primeiro menu.
8	Para inserir o próximo item da barra de menus, de um clique na barra de menus e repita os passos 3 e 4.
9	Adicione qualquer item ao segundo menu repetindo os passos 5 e 6.

Adicionando Barras Separadoras

Adicionar uma barra separadora insere uma linha entre itens de menu ou grupos de itens. Você pode utilizar arras separadoras para agrupar os itens logicamente ou criar uma quebra visual na lista de itens. Para criar uma barra separadora, digite um hifen (-) como propriedade Caption para este item. Barras separadoras aparecem como segue:



Habilitando Acesso aos Menus pelo Teclado

Você pode permitir que os usuários acessem menus e comandos de menu através do teclado, como segue:

- Teclas de atalho
- Teclas aceleradora

A figura anterior também mostra um exemplo de um menu com teclas aceleradoras e teclas de atalho.

Adicionando Teclas Aceleradoras

Teclas aceleradoras permitem que o usuário acesse menus e comandos do menu utilizando a tecla **Alt** juntamente com a letra apropriada. Por exemplo, para acessar o menu File, o usuário pressionaria **Alt+F**. Você pode adicionar teclas aceleradoras ao seu menu precedendo a letra selecionada com o símbolo **&**, quando estiver digitando o caption do item. Por exemplo, para adicionar um menu Arquivo com a letra A como tecla aceleradora, digite *&Arquivo* como caption do item de menu.

Adicionando Teclas de Atalho

Teclas de atalho permitem que o usuário acesse comandos do menu utilizando o teclado para ignorar os menus. Por exemplo, você pode designar F1 como tecla para o comando Help. Você pode especificar atalhos digitando um valor na propriedade ShortCut ou selecionando na lista da propriedade ShortCut.

O Delphi não faz a checagem para assegurar que teclas aceleradoras ou atalhos não estão duplicados em sua aplicação. Isto é responsabilidade do desenvolvedor.

Propriedades Adicionais de Menu

Você pode definir as seguintes propriedades de menu para alterar a aparência dos menus em sua aplicação.

- Checked

  Definindo esta propriedade para **True**, um check aparece próximo ao item.
- Enabled

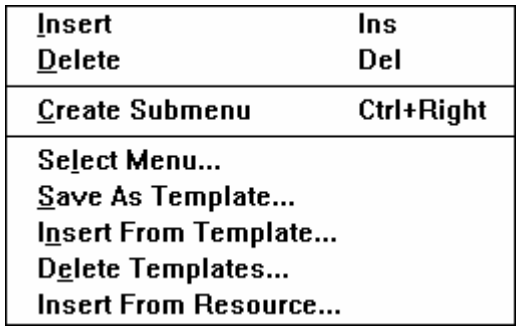
  Definindo esta propriedade para **False**, o item torna-se desabilitado, ou acinzentado, e o usuário não poderá acessa-lo ou a qualquer um de seus submenus.

Criando Menus Aninhados

Muitos menus contém menus aninhados, ou submenus, que fornecem ao usuário comandos adicionais para serem selecionados. Menus aninhados são indicados por uma seta próxima ao item na lista de itens. Você pode querer utilizar menus aninhados para economizar espaço vertical em sua aplicação. No Delphi, você pode criar tantos menus aninhados quantos forem necessários.

Utilizando o SpeeMenu do Menu Designer

O SpeedMenu do Menu Designer oferece acesso rápido aos comandos mais utilizados e opções de template de menu. Você pode exibir o SpeedMenu dando um clique com o botão direito do mouse na janela do Menu Designer ou pressionando **Alt+F10** enquanto o cursor estiver na janela do Menu Designer. O SpeedMenu aparece, como segue:



Descrição das Opções do SpeedMenu

A tabela a seguir descreve as opções no SpeedMenu do Menu Designer:

Opção	Descrição
Insert	Adiciona uma localização de item de menu antes do cursor
Delete	Remove o item de menu selecionado
Create Menu	Insere um item de menu a direita do item selecionado, adiciona uma ponta de seta para indicar um nível de alinhamento de menu
Select Menu	Permite selecionar um menu existente
Save As Template	Permite gravar um menu para utilização posterior
Insert From Template	Permite adicionar um template de menu previamente criado ao menu atual
Delete Template	Permite remover menus criados previamente
Insert From Resource	Permite importar um menu de um arquivo de resource (.RC) do Windows

Associando Eventos de Menu com Código

Introdução

Uma vez criado um menu, você precisa associar ao evento OnClick de cada item.

Manipulando Eventos de itens de Menu

O evento OnClick é o único evento para itens de menu. O código associado com o evento OnClick dos itens dos menus serão executados quando o usuário selecionar o item.
Você pode criar um event handler para qualquer item de menu dando um duplo-clique no item, na janela do Menu Designer e adicionando o código dentro da instrução begin...end.

Associando um Item de Menu a um Manipulador de Manu Existente

Se você quiser reutilizar código, você pode associar um item de menu com um event handler já existente. Para isto, vá para a página Events do Object Inspector e abra a lista dos event handlers para o evento OnClick. Você pode selecionar qualquer um dos event handlers que aparecem na lista, e o código para aquele event handler estar associado com o item de menu selecionado.

Lab: Adicionando um Template de Form a sua Aplicação

Objetivo:

Este lab reforça sua habilidade em:

- Utilizar um Template de Form para adicionar um form
- Digitar código para exibir um form
- atribuir teclas de atalho aos menus

Cenário

Neste lab você utilizará a aplicação desenvolvida no lab Adicionando Componentes Standard a uma Aplicação para adicionar um quadro de dialogo Sobre... utilizando um template de form.

Processo

Utilize o processo a seguir para aplicar o que você aprendeu:

Estágio	Processo
1	Abra o projeto chamado PLAB5.DPR. Grave o arquivo unit como ULAB8.PAS e o projeto como PLAB8.DPR. **Importante:** Certifique-se de utilizar o comando **Save File As**, bem como **Save Project As** no menu **File**.
2	Abra o Menu Designer para MainMenu1. Adicione um separador e a palavra Sobre no final do menu **File**.
3	Utilize um template de form para adicionar um quadro Sobre a aplicação. Certifique-se de gravar este form com o nome UNIT2.PAS.
4	Adicione uma tecla de atalho ao menu para que o usuário possa pressionar **F2** para exibir o quadro Sobre.
5	Adicione aceleradores para os itens **Abrir** e **Gravar**.
6	Adicione um event handler que exiba o from AboutBox.
7	Após compilar sua aplicação, grave-a. Execute e teste sua aplicação.

Resumo do capítulo

Pontos Chave

Após completar este capítulo,você aprendeu que:

- Templates fornecem uma "planta" para os forms mais utilizados, e eles compõem a Form Gallery.
- Você pode gravar uma cópia de um form por você como um template para ser utilizado posteriomente em outro projeto. Gravar um form como um template insere o form na lista de templates que você pode selecionar quando cria um novo form.
- Quando você cria um form, ele possui as seguintes características:

Botões de Maximizar e Minimizar
Menu Controle
Borda Reajustável

- Ordem de Tab é a seqüencia na qual o usuário pode se mover de componente a componente pressionando a tecla **Tab**.
- Menus fornecem uma maneira de agrupar comandos. Você pode utilizar o Menu Designer para criar ou adicionar um menu predeterminado ao form.

Termos e Definições

A tabela a seguir é uma referência rápida aos termos explicados neste capítulo:

Termo	Definição
MDI	Multiple Document Interface, uma aplicação feita de uma janela principal, ou janela pai, que permite abrir diversos documentos ou janelas
Ordem de Tab	A seqüencia na qual um usuário pode se mover de componente a componente pressionando a tecla Tab
Quadro de DiálogoModal	Um quadro de dialogo onde o usuário deve fechar antes de continuar a utilizar a aplicação
Quadro de Diálogo Não-Modal	Um quadro de dialogo que o usuário pode manter aberto enquanto continua a trabalhar na aplicação
SDI	Single Document Interface, uma aplicação que pode processar mais que um documento. Mas os documentos estão contidos de uma janela pai.

Capítulo 6

Adicionando Componentes Standard à uma Aplicação

Overview

Este capítulo cobre os componentes nas páginas Standard e Additional da Component Palette. Cada um destes componentes é descrito com propriedades, eventos e métodos significantes. Ao final do capítulo, você aplicará o que aprendeu para criar uma aplicação utilizando diversos componentes Standard.

Descrição dos Componentes Standard

Introdução

Como mencionado anteriormente, componentes Standard (Padrão) são componentes tipicamente utilizados no desenvolvimento de aplicações MS Windows. Eles estão nas páginas Standard e Additional da Component Palette (a página Additional contém componentes padrão adicionais).

Descrição dos Componentes Standard

A tabela a seguir descreve os componentes Standard na Component Palette:

Ícone	Visual	Propósito	Propriedades,Eventos,ou Métodos Significantes
MainMenu	Não	Permite criar menus para um form	P:Items P:AutoMerge M:Merge M:UnMerge
Popup Menu	Não	Permite criar menus pop-up tipo SpeedMenu	P:Items P:PopUpMenu P:AutoPopup E:OnPopup M:Popup
Label	Sim	Exibe texto, tais como títulos, que o usuário não pode acessar	P:Caption P:Alignment P:AutoSize

			P:Transparent P:FormControl P:WordWrap
Edit	Sim	Exibe uma área onde o usuário pode inserir ou alterar uma única linha de texto	P:Text P:Modified P:MaxLength M:SelectAll M:ClearSelection
Memo	Sim	Exibe uma área onde o usuário pode inserir ou alterar diversas linhas de texto	P:Text P:Modified P:MaxLength P:Lines M:Add M:Delete
Button	Sim	Um controle tipo botão. Os usuários dão um clique no botão para iniciar	P:Default P:Cancel P:ModalResult E:OnClick
CheckBox	Sim	Apresenta opções que o usuário pode habilitar ou desabilitar	P:Checked P:Caption P:AllowGrayed P:State E:OnClick
Radio Button	Sim	Apresenta opções exclusivas mutuamente	P:Checked P:Caption E:OnClick
ListBox	Sim	Exibe uma lista de escolhas	P:ItemIndex P:Columns P:MultiSelect P:Selected P:Items M:Add* M:Delete* M:Insert* * Items property (TStrings object)
Combo Box	Sim	Combina um edit box e uma list box para exibir uma lista de escolhas	P:Text P:ItemIndex P:Sorted P:Items M:add* M:Delete* M:Insert* * Items property (TStrings object)
ScrollBar	Sim	Move através de uma faixa de incrementos	P:Kind P:LargeChange P:SmallChange P:Min P:Max P:Position E:OnScroll M:SetParms
GroupBox	Sim	Agrupa componentes, geralmente	P:Caption

		utilizado para representar um grupo de opções relacionadas	P:Parent
Radio Group	Sim	Agrupa radio buttons para que trabalhem em conjunto como um grupo	P:Columns P:Items P:ItemIndex
Panel	Sim	Exibe um painel onde outros componentes podem ser inseridos	P:Align P:Alignment P:Caption

Descrição da Página de Componentes Additional

A tabela a seguir descreve os componentes na página Additional da Component Palette:

Ícone	Visual	Propósito	Propriedades, Eventos, ou Métodos Significantes
BitBtn	Sim	Fornece um botão que exibe um bitmap no botão	P:Kind P:Glyph P:Default P:Cancel P:ModalResult E:OnClick
Speed Button	Sim	Fornece um botão para representar um processo	P:Glyph P:NumGlyphs P:Layout P:Margin
TabSet	Sim	Cria abas de um Notebook para dar aparência de páginas	P:Tabs P:TabIndex P:FirstIndex P:Align
Notebook	Sim	Fornece uma pilha de diversas páginas (Utilizadas em combinação com o componente TabSet)	P:Pages P:PageIndex P:ActivePage M:OnClick
Tabbed Notebook	Não	Cria um quadro de diálogo com diversas páginas com abas para agrupar informações	P:ActivePage P:Pages P:PageIndex P:TabFont P:TabsPerRow M:GetIndexForPage M:SetTabFocus
MaskEdit	Sim	Exibe uma área onde o usuário pode inserir ou alterar texto utilizando somente caracteres válidos especificados pela propriedade EditMask	P:Text P:EditMask P:EditText P:MaxLength
Outline	Sim	Exibe informação em formas variadas de tópicos	P:CurItem P:Items M:Add* M:Delete* M:Insert* * Itens(Objeto TStrings)
StringGrid	Sim	Fornece uma maneira de exibir strings em colunas e linhas	P:Cells P:Objects

			P:Cols P:Rows
DrawGrid	Sim	Fornece uma maneira de exibir informação gráfica em linhas e colunas	P:DefaultDrawing P:Selection E:OnDrawCell M:CellRect M:MouseToCell
Image	Sim	Exibe um bitmap, ícone, ou metafile	P:Picture P:AutoSize P:Stretch
Shape	Sim	Exibe forrnas geométricas, tais como elipses, retângulo ou retângulo com cantos arredondados	P:Shape P:Align P:Color* P:Style* *(Object TBrush)
Bevel	Sim	Fornece um retângulo com linhas ou bordas em alto ou baixo relêvo	P:Shape P:Style P:Align
Header	Sim	Fornece um controle seccionado que exibe texto e permite que cada seção seja reajustada utilizando-se o mouse. Pode ser utilizado com o componente grid para exibir cabeçalhos de colunas	P:Componentes P:Sections
ScrollBox	Sim	Exibe uma área que pode ser rolada	P:HorzScrollBar P:VertScrollBar M:ScrollInView

Lab: Criando uma Aplicação Usando Componentes Standard

Objetivos

Este lab reforça sua habilidade em:

- Adicionar diversos componentes Standard ao form
- Adicionar componentes em um painel
- Digitar e exibir Help Hints
- Associar event handlers com components
- Utilizar um método de um componente

Cenário

Você desenvolverá um programa de bloco de notas. Este programa permitirá digitar texto, recortar e colar texto, gravar e abrir um arquivo texto. No processo de construção desta aplicação você utilizará os seguintes componentes das páginas Standard e Additional da Component Palette:

- Memo
- MainMenu
- Edit
- Label
- Panel
- SpeedButton

Processo

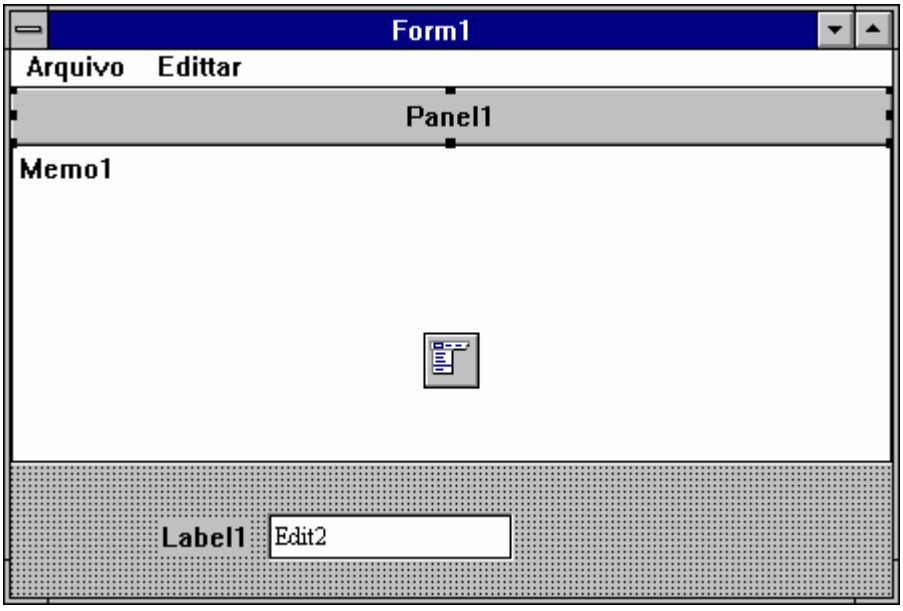
Utilize a seguir para aplicar o que você aprendeu:

Estágio	Processo
1	Abra um novo projeto. Grave o projeto nomeando o arquivo unit como ULAB5.PAS, e o arquivo de projeto como PLAB5.DPR.
2	Crie um menu principal com itens drop-down utilizando os seguintes modelos:

Arquivo
Novo
Abrir
Gravar
Sair

Editar
Recortar
Copiar
Colar

Estágio	Processo
3	Adicione os seguintes componentes: • Panel • Memo • Label • Edit Reajuste o tamanho dos componentes para que seu form esteja similar ao da figura a seguir:



Estágio	Processo
4	Defina as seguintes propriedades para os componentes do form:

Nome do Componente	Propriedade	Valor

Panel1	Caption Align	(Vazio) alTop
Label1	Caption	Nome do Arquivo:
Edit1	Text Name	(Vazio) EditBox
Memo1	Lines	(Vazio)
Form1	Caption	Bloco de Notas

Estágio	Processo
5	Adicionar event handlers para os itens **Recortar, Copiar e Colar** no menus que irão recortar, copiar, e colar textos no componente memo. Utilize os métodos CutToClipboard, CopyToClipboard, e PasterFromClipboard do componente TMemo. Utilize o Help para ver como estes métodos são utilizados e quais parâmetros (se houver) necessários.
6	Adicione um event handler OnClick para o item S**air** do menu que fehará a aplicação inteira.
7	Compile e grave sua aplicação e execute-a para testar
8	O componente Memo contém um objeto TString chamado Lines que contém o texto do objeto memo. Três métodos disponíveis para objetos TStrings são: -LoadFromFile -SaveToFile -Clear Procure por exemplos no Help e utilize-os para adicionar event handlers para os itens **Novo, Abrir, e Gravar** do menu **Arquivo**. Dica: Use o método Clear para o item **Novo** do menu. Utilize o valor digitado na propriedade Text do EditBox como nome de arquivo. Por exemplo, o event handler de SaveClick deve ser similar ao código a seguir:

```
Procedure TFrom1.Gravar1Click(Sender: Object);
begin
   if EditBox.Text  < > "then
      Memo1.Lines.SaveToFile(EditBox.Text);
end;
```

Estágio	Processo
9	Compile, grave e execute sua aplicação para testá-la.
10	Adicione três SpeedButtons no Panel de seu from. Nomeie-os como segue:

Nome do Componente	Propriedade	Valor
SpeedButton1	Nome	sbArquivoNovo
SpeedButton2	Nome	sbArquivoAbrir
SpeedButton3	Nome	sbArquivoGravar

Estágio	Processo
11	Utilize Object Inspector para associar cada um dos SpeedButtons com bitmap. Associe os bitmaps como segue:

Nome do Componente	Propriedade	Nome do Arquivo
sbArquivoNovo	Glyph	FILENEW.BMP
sbArquivoAbrir	Glyph	FILEOPEN.BMP

sbArquivoGravar	Glyph	FILESAVE.BMP

Estágio	Processo
12	Utilize a página Events do Object Inspector para associar cada um dos SpeedButtons como event handler apropriado. Associe as procedures como segue:

Nome do Componente	Evento	Event Handler
sbArquivoNovo	OnClick	Novo1Click
sbArquivoAbrir	OnClick	Abrir1Click
sbArquivoGravar	OnClick	Gravar1Click

Estágio	Processo
13	Utilize o Object Inspector para digitar um Hint para cada um dos SpeedButtons. Utilize a tabela a seguir:

Nome do Componente	Hint
sbArquivoNovo	Novo
sbArquivoAbrir	Abrir
sbArquivoGravar	Gravar

Estágio	Processo
14	Defina a propriedade ShowHints do componente Panel para True.
15	Compile, grave e execute sua aplicação para testá-la. Sua aplicação deve estar similar à figura a seguir. Grave seu projeto. Ele será utilizado novamente em lab posterior.

Processo Opcional

Siga os processos para prática adicional, se o tempo permitir:

Estágio	Processo
1	Adicione um componente PopupMenu contendo os seguintes itens de menu:

	Recortar Copiar Colar Gravar Sair
2	Atribua este menu pop-up à propriedade PopupMenu do componente TMemo (Memo1).
3	Associe os event handler apropriados aos itens do menu pop-up.
4	Compile, grave e execute sua aplicação para testá-la. Quando chamar o menu pop-up, sua aplicação deve estar similar à figura a seguir:

Resumo do Capítulo

Pontos Chave

Após completar este capítulo, você aprendeu que:

- Componentes Standard são componentes utilizados em aplicações típicas do MS Windows.
- Os componentes da página Additional são componentes Standard adicionais.

Termos e Definições

A tabela a seguir é uma referência rápida aos termos apresentados neste Capítulo

Termo	Definição
Help Hint	O significado de um botão que aparece quando você passa com o ponteiro do mouse sobre um botão

Capítulo 7

Arquitetura de Acesso a Dados do Delphi

Overview

Utilizando o Delphi, você pode criar aplicações de bancos de dados que trabalhem diretamente com bancos de dados desktop ou servidores de bancos de dados remotos, tais como Oracle, Sybase, e bancos de dados padrão ODBC. Este capítulo o introduz à arquitetura das aplicações Client/Server, bem como as ferramentas utilizadas para construí-las.

Características e Capacidades de Bancos de Dados

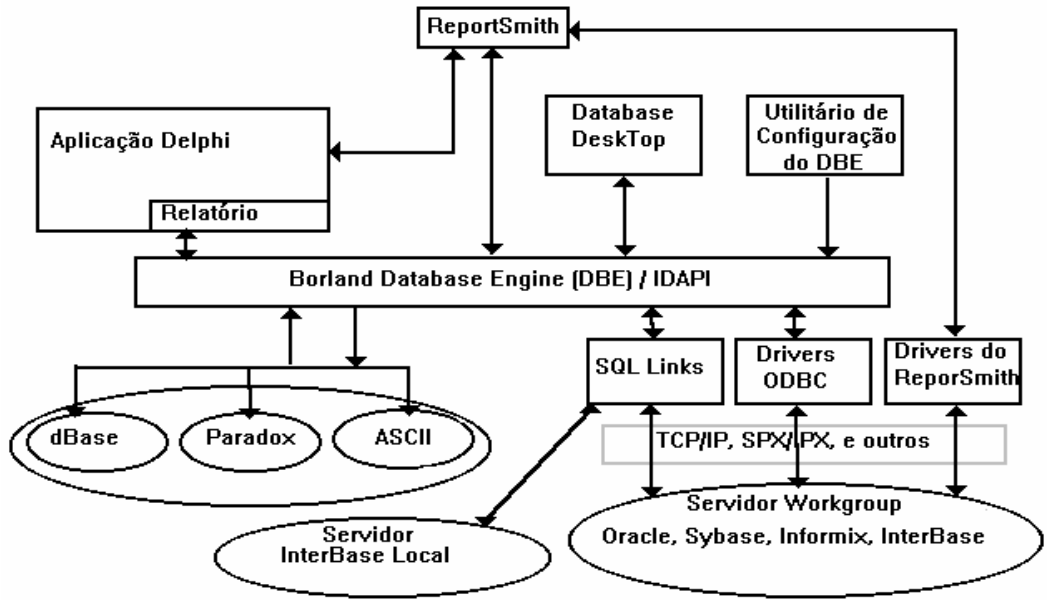
Introdução

Você pode construir uma aplicação de bancos de dados utilizando as ferramentas de desenvolvimento de banco de dados do Delphi, componentes Data Access e Data Control. Em uma aplicação de bancos de dados Delphi, os componentes se comunicam com o Borland Database Engine (BDE), que por sua vez, comunica-se com suas outras fontes:

- Dados locais, tais como dBASE, Paradox, e InterBase for Windows
- Dados remotos, tais como Oracle, Sybase, e InterBase

Relacionamento entre o Delphi e a Conectividade de Banco de Dados

O diagrama a seguir mostra o relacionamento entre as ferramentas do Delphi, aplicações de banco de dados do Delphi, o BDE, e os dados:



Os tópicos restantes deste capítulo fornecem maiores detalhes sobre o relacionamento mostrado neste diagrama.

Ferramentas de Bancos de Dados

Introdução

Você pode utilizar diversas ferramentas de bancos de dados na construção deste tipo de aplicação.

Ferramentas de Bancos de Dados

As ferramentas a seguir permitem construir aplicações de banco de dados que podem trabalhar com arquivos de dados locais ou em um servidor de rede local:

- Borland Database Engine (BDE)
- SQL Links
- Database Desktop
- Componentes Data Access
- Componentes Data Control

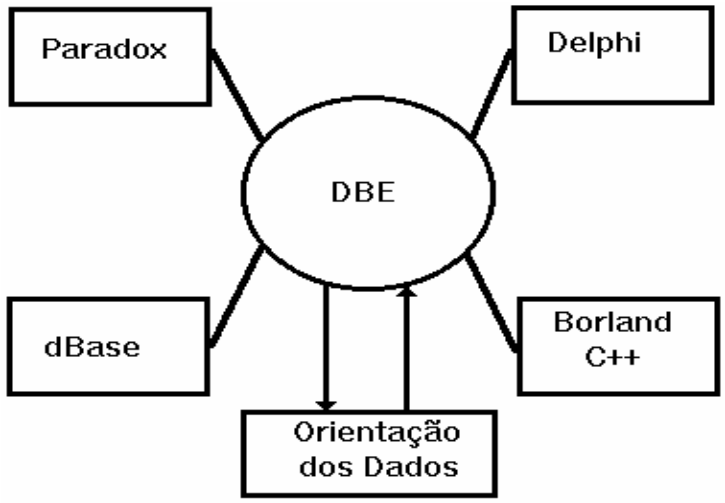
Borland Database Engine (BDE)

Introdução

O Borland Database Engine (BDE) é construído no Delphi para oferecer uma conexão entre controles Data Access e dados de várias fontes como Paradox, dBASE, Oracle, e InterBase. Bancos de dados, linguagens e ferramentas da Borland compartilham o mesmo Borland Database Engine. Este design em comum permite que os componentes sejam intercambiados para construir um ambiente de aplicação Client/Server que atenda às suas necessidades.

Relacionamento entre o BDE e Outros Produtos Borland

O Borland Database Engine é um componente integral do dBASE for Windows, Paradox for Windows, Delphi, e ReportSmith. Também está disponível como biblioteca API de engine de banco de dados para programadores C e C++.
O diagrama a seguir mostra o relacionamento entre o Borland Database Engine e os diversos produtos Borland:



O Borland Database Engine permite que ferramentas e aplicações Borland conectem transparentemente à dados residindo no dBASE, Paradox , Oracle, Sybase, Interbase, Informix, DB2, e qualquer banco de dados que utilizem o Open Database Connectivity (ODBC).

SQL Links

Introdução

O SQL Links dá às suas aplicações Delphi acesso a dados localizados em servidores Workgroup baseados em SQL, incluindo Sybase, você pode construir comandos SQL e passá-los através de um diver SQL Link ao servidor para execução.

Criando Aliases

Um Alias consiste de um nome e conjunto de parâmetros que descrevem recursos de rede e especificam a localização das tabelas do banco de dados. Um alias é necessário para acessar banco de dados SQL.
Embora você possa personalizar um alias após criá-lo, é mais fácil escolher parâmetros do diver que combinem com sua instalação servidora antes de criar os aliases para bancos de dados SQL. Desta forma, qualquer alias que você crie herdam estes parâmetros.

Configurando o Ambiente SQL

O diver SQL Link utiliza os parâmetros default quando instalado pela primeira vez. Como os parâmetros do SQL Link para seu diver serve como modelo para cada alias que você crie, você deve determinar se estes parâmetros estão corretos para sua instalação servidora antes de criar aliases para seu banco de dados SQL.
Para modificar seu ambiente SQL, utilize o BDE Configuration Utility (BDECFG.EXE) instalado com o Delphi. Os aliases são armazenados no arquivo de configuração IDAPI (IDAPI.CFG).

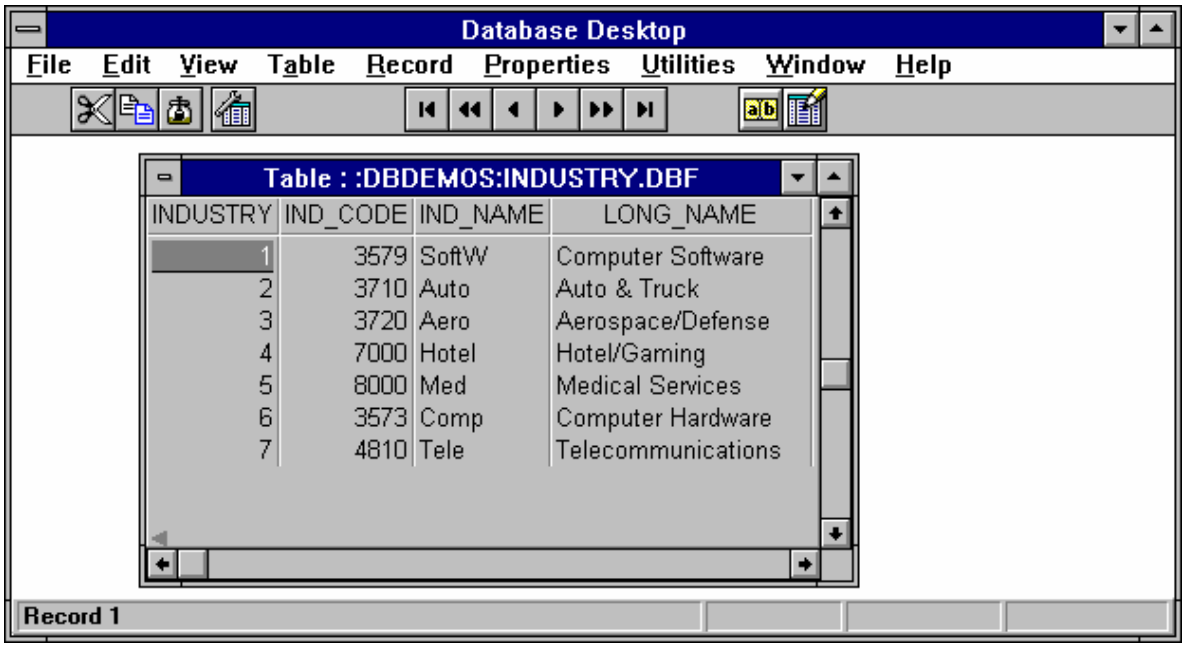
Database Desktop

Introdução

O Database Desktop (DBD) é uma ferramenta de definição e manutenção de banco de dados que lhe permitem consultar, criar, reestruturar, modificar e copiar tabelas de banco de dados. O Database Desktop também pode copiar dados e informações do dicionário de dados de um formato para outro.

Inicializando o Database Desktop

Você pode inicializar o Database Desktop no Gerenciador de Programas, dê um duplo-clique sobre o ícone do Database Desktop no grupo de programa Delphi. Quando selecionar uma tabela, a tabela aparece na janela do Database Desktop, como mostrado no exemplo a seguir:



Novos Recursos do Database Desktop

O Database Desktop contém as seguintes melhorias:

Recurso	Descrição
Menu Utilities	O Database Desktop agora inclui um menu Utilities com os seguintes comandos: • Add Adiciona registros de uma tabela em outra tabela • Copy Cria uma cópia de uma tabela, query QBE, ou comando SQL • Delete Deleta uma tabela, query QBE, ou comando SQL • Empty Remove todos os registros de uma tabela • Passwords Permite adicionar e remover senhas da lista de senhas utilizadas na sessão atual • Rename Altera o nome de uma tabela, query QBE, ou comando SQL • Sort Ordena os dados em uma registros que existam em outra tabela • Substract Remove de uma tabela de registro que existam em outra tabela
SQL	Você pode utilizar os seguintes comandos com tabelas SQL: • Add • Copy • Delete • Empty Acesso SQL está disponível para operações de queries e tabelas, incluindo criação e reestruturação. Você pode executar comandos SQL utilizando o SQL Editor.

Documentação do Database Desktop

O Delphi fornece a seguinte documentação do Database Desktop:

Documentação	Descrição
Online Help	Documentação completa do Database Desktop com os recursos usuais de help através da tecla F1, procura de tópicos e help sensível ao contexto. Além disso muitos tópicos possuem help orientado à tarefa
Delphi Database Application Developer's Guide	Documentação completa sobre o desenvolvimento de aplicações com um apêndice que resume os recursos do Database Desktop
Database Desktop User's Guide	Documentação completa sobre a utilização do Database Desktop, disponível no CD-ROM do Delphi

Descrição dos Componentes Data Access

Introdução

Os componentes Data Access estão na página Data Access da Component Palette. Estes componentes permitem construir aplicações de bancos de dados. Eles, juntamente com os componentes Data Control, são componentes data-aware. Componentes data-aware são componentes que tem ciência de dados e sua estrutura em um banco de dados.
Quando você constrói uma aplicação de banco de dados, você insere componentes Data Access em form e atribui propriedades que especificam o banco de dados, tabelas e registros a serem utilizados com este form. Muito embora componentes Data Access não sejam visíveis durante a execução, eles trabalham em segundo plano para regular o acesso aos dados.Os componentes Data Access aparecem na Component Palette.

Descrição dos Componentes Data Access

A tabela a seguir descreve os componentes Data Access na Component Palette:

Ícone	Visual	Propósito	Propriedades, Eventos, ou Métodos Significantes
TDataSource	Não	Atua como um canal entre um objeto Ttable ou Tquery e componentes de edição de dados como TDBGrid	P:AutoEdit P:State P:Enable P:Dataset P:OnDataChange E:OnUpdateData M:Edit
TTable	Não	Recupera dados de tabelas de bancos de dados através do BDE e os passa para um ou mais componentes data-aware	P:DatabaseName P:TableName P:ReadOnly P:Exclusive P:IndexName P:Tag P:MasterFields P:MasterSource M:GotoCurrent
TQuery	Não	Utiliza comandos SQL para recuperar dados de uma tabela através do BDE, ou utiliza comandos SQL	P:SQL P:DataSource

		para enviar dados de um componente a um banco de dados através do BDE	P:Local P:ParamCount P:Param P:Unidirectional P:ExecSQL
StoredProc	Não	Permite que aplicações executem stored procedures do servidor	P:DatabaseName M:Prepare M:ExecProc M:Open
TDatabase	Não	Define uma conexão contínua à um banco de dados	P:Active P:DatabaseName P:Handle P:PassWord M:Close M:Create M:Destroy M:Open
TBacthMove	Não	Copia a estrutura de uma tabela e seus dados . Pode ser utilizado para mover tabelas inteiras de um banco de dados a outro	P:Mode P:Source P:Destination P:ChangedTable Name P:KeyViolTable Name P:ProblemTable Name M:Execute
TReport	Sim	Permite a criação e impressão de relatórios de bancos de dados através do ReportSmith	P:ReportName P:ReportDir P:Preview P:Run P:AutoUnload P:UnitialValues M:RunMacro M:Connect M:SetVariable M:ReCalcReport

Descrição dos Componentes Data Control

Introdução

Os componentes Data Control dão às suas aplicações Delphi de banco de dados uma interface visual consistente, quer sua aplicação acesse um arquivo de banco de dados local ou um servidor remoto.
Baseado na instalação padrão, os componentes Data Control estão localizados na página Data Controls da Component Palette. Estes componentes fornecem um conjunto de componentes data-aware de interface de usuário, que você pode utilizar para criar aplicações baseadas em forms. Como mencionado anteriormente, os componentes Data Control e Data Access são data-aware.
Além da função de um componente Standard, componentes Data Control também podem exibir dados de um campo de uma tabela de um banco de dados ou enviar dados à uma tabela de um form. Os componentes Data Control aparecem na Component Palette.

Descrição dos Componentes Data Control

A tabela a seguir descreve os componentes Data Control na Component Palette:

Ícone	Visual	Propósito	Propriedades, Eventos, ou Métodos Significantes
TDBGrid	Sim	Permite visualizar e editar dados em formato tabular (tipo planilha). Faz uso extensivo das propriedades do Tfield para determinar a visibilidade de uma coluna, modo de exibição, e assim por diante.	P:Options P:ReadOnly P:Fields P:SelectedField P:FieldCount
TDBNavigator	Sim	Permite fazer o seguinte: -Mover o ponteiro de registro da tabela atual para frente ou trás -Iniciar o modo Insert ou Edit -Efetuar a inserção de registros novos ou modificados (Post)	P:Data Source P:VisibleButtons
TDBLable	Sim	Permite exibir um campo de um registro atualmente ativo. É a versão data-aware do componente TLabel	P:DataSource P:DataField P:Caption P:FocusControl P:Alignment P:AutoSize P:WordWrap P:Transparent
TDBEdit	Sim	Permite exibir ou editar um campo de um registro ativo. É a versão data-aware do componente TEdit	P:DataSource P:DataField P:EditMask P:Modified P:MaxLength P:ReadOnly P:AutoSelect P:SelText P:SelStart P:SelLentgh P:PassWordChar P:AutoSize M:SelectAll M:ClearSelection M:CutToClipboard M:CopytoClipboard M:PasterFrom Clipboard
TDBMemo	Sim	Permite exibir ou editar BLOBs de um registro ativo. É a versão data-aware do componente TMemo	P:DataSource P:DataField P:Text P:ReadOnly P:Modified P:MaxLength P:Lines P:SelText P:SelStart P:SelLength P:PassWordChar P:AutoSize P:ScrollBars

			P:WordWrap P:WantTabs P:M:Add M:Delete M:Insert M:CutToClipboard M:CopyToClipboard M:PasterFromClipboard M:SelectAll
TDBImage	Sim	Permite exibir, recortar, ou colar imagens bitmap BLOB para e de um registro ativo. É a versão data-aware do componente TImage	P:DataSource P:DataField P:Picture P:AutoDisplay
TDBListBox	Sim	Permite exibir valores de uma coluna de uma tabela. É a versão data-aware do componente TListBox	P:ReadOnly P:DataField P:Items P:ItemIndex M:Add M:Delete M:Insert
TDBCombo Box	Sim	Permite exibir ou editar valores de uma coluna de uma tabela. É a versão data-aware do componente TComboBox	P:ReadOnly P:DataSource P:DataField P:Items P:ItemIndex P:Sorted P:SelText P:SelStart P:SelLength M:Add M:Delete M:Insert M:SelectAll
TDBCheck Box	Sim	Permite exibir ou editar um campo de dado Booleano de um registro ativo. É a versão data-aware do componente TCheckBox	P:DataSource P:DataField P:ValueChecked P:ValueUnchecked P:ReadOnly
TDBRadioGroup	Sim	Permite exibir ou definir valores de colunas. É a versão data-aware do componente TGroupBox populado com radio buttons	P:DataSource P:DataField P:Items P:Value P:ReadOnly
TDBLookupList	Sim	Permite exibir dados de uma tabela de busca durante a execução. É a versão data-aware do componente TListBox	P:DataField P:DataSource P:LookupDisplay P:LookupField
TDBLookupCombo	Sim	Permite exibir dados de uma tabela de busca durante a exibição. É a versão data-aware do componente TComboBox	P:LookupSource P:LookupField P:LookupDisplay P:DataSource P:DataField

Resumo do Capítulo

Pontos Chave

Após completar este capítulo, você aprendeu que:

- No Delphi, você pode desenvolver uma aplicação de banco de dados que trabalhe diretamente com um banco de dados local (baseado em PC) e bancos de dados remotos (SQL).
- Os componentes Data Access e Data Control são componentes data-aware.
- Várias ferramentas permitem construir aplicações de bancos de dados. Estão incluídas:

Ferramenta	Descrição
Borland Database Engine (BDE)	Fornece uma conexão entre os controles Data Access e dados de um servidor de banco de dados remoto, no servidor de rede local ou local
Data Access Components	Componentes na página Data Access da Component Palette que permitem construir uma aplicação de banco de dados
Data Control Components	Componentes na página Data Controls da Component Palette que permitem construir uma aplicação de banco do dados
Database Desktop (DBD)	Uma ferramenta de definição e manutenção de dados que permite pesquisar, criar, reestruturar, modificar e copiar tabelas de banco de dados
SQL Links	Fornece acesso SQL aos dados localizados em servidores remotos às suas aplicações Delphi

Termos e Definições

A tabela a seguir é uma referência rápida aos termos explicados neste capítulo :

Termo	Definição
Alias	Um nome e um conjunto de parâmetros que descrevem um recurso de rede local e especifica a localização das tabelas de banco de dados .É necessário um alias para acessar um banco de dados SQL
BLOB	Binary Large Object
Data aware	Ter ciência dos dados e sua estrutura em um banco de dados
DBD	Database Desktop, uma ferramenta e definição e manutenção de dados que permite pesquisar, criar, reestruturar, modificar e copiar tabelas de bancos de dados
ODBC	Open Database Connectivity

Capítulo 8

Adicionando Componentes de Banco de Dados à uma Aplicação

Overview

Os componentes de banco de dados e suas capacidades descritas neste capítulo fornecem as ferramentas necessárias para a criação de aplicações que utilizem bancos de dados local, baseado em PC e remotos, baseados em servidores SQL.

Este capítulo aplica os componentes de banco de dados (descritos no capítulo anterior) no desenvolvimento de uma aplicação. Estes componentes estão nas páginas Data Access e Data Controls da Component Palette.

Conceito de um DataSet no Delphi

Introdução

Para manipular e consultar bancos de dados, você precisa entender o conceito de dataset. Um dataset no Delphi é um objeto que consiste de uma série de registros, cada um contendo qualquer quantidade de campos e um ponteiro para o registro atual. O dataset pode ter uma correspondência direta, um-para-um, com uma tabela física, ou, como um resultado de uma query, pode ser um subconjunto de uma tabela ou uma junção de diversas tabelas.

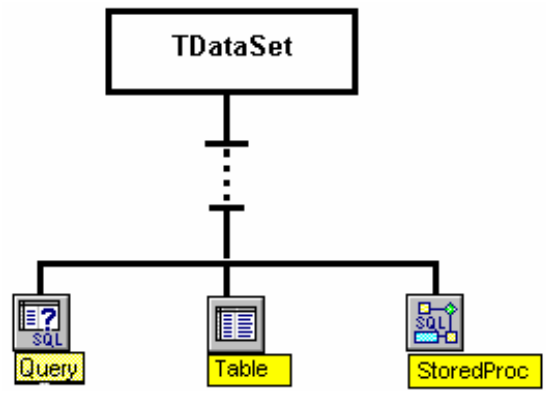
Tipo de Objeto TDataSet

Um dataset no Delphi é o tipo de objeto TDataSet e é uma classe abstrata. Uma classe abstrata é uma classe de onde você pode derivar outras classes, mas não pode criar uma variável desta classe. Por exemplo, ambos os componentes Query e Table classificam-se como componentes TDataSet porque cada um foi derivado do objeto TDataSet. Note que você não encontrará nenhum componente chamado TDataSet na Component Palette. O TDataSet contém as abstrações necessárias para manipular diretamente uma tabela. É a ferramenta utilizada para abrir uma tabela e navegar por suas colunas e linhas.
Os componentes neste capítulo são referenciados pelo seu tipo de objeto (identificado no Delphi pelo seu nome com o prefixo T). O termo componente DataSet é utilizado para referenciar um componente Table, Query, ou StoredProc. TTable, TQuery, e TStoredProc são descendentes de TDataSet, ou seja, eles herdam as propriedades de TDataSet.

O conceito de herança é discutido em detalhes no capítulo Object-Oriented Programming in Delphi de seu manual.

Diagrama dos componentes DataSet

O diagrama a seguir mostra o conceito dos componentes DataSet, que são componentes derivados do tipo de objeto TDataSet:



Utilizando o Componente e DataSource

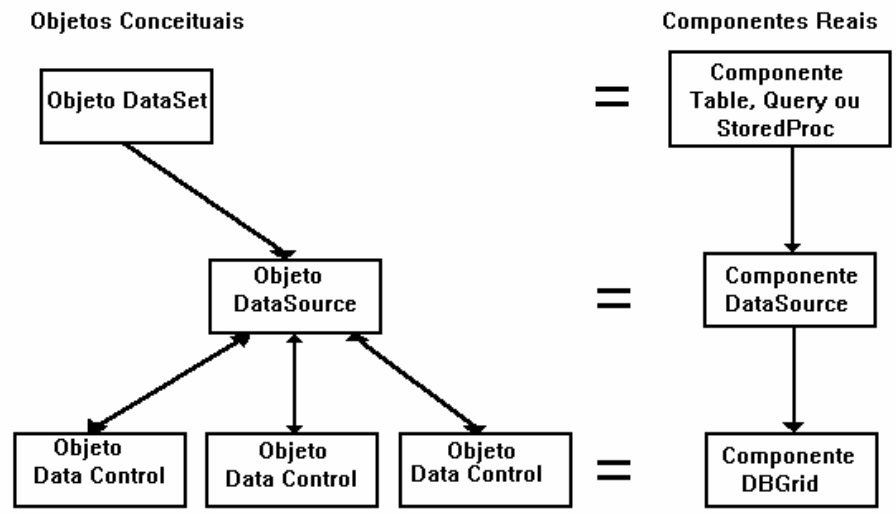
Introdução

O componente DataSource atua como intermediário entre o componente DataSet (TTable, TQuery, ou TStoredProc) e os componentes Data Control. Dentre os componentes Data Control incluem DBGrid, e DBText entre outros. Esta seção explica:

- Qual o papel do componente DataSource em uma aplicação de banco de dados
- Como utilizar propriedades e eventos do DataSource
- Como adicionar componentes DataSource à sua aplicação

Papel de um Componente DataSource

Um Componente DataSource gerencia o relacionamento entre uma tabela de banco de dados e a representação deste dado em seu form. É um intermediário entre os componentes DataSet e DataControl. Componentes DataSet gerenciam a comunicação com o Borland Database Engine (BDE), e o componente DataSource gerencia a comunicação com componentes data-aware Data Control.
Em uma típica aplicação de banco de dados, um componente DataSource é associado com um componente DataSet (Table ou Query) e um ou mais componentes Data Control (tais como DBGrid).
O diagrama a seguir mostra este relacionamento:



Utilizando Propriedades e Eventos DataSource

As propriedades e eventos chave do componente DataSource são:

- Propriedade DataSet
- Propriedade Enabled
- Propriedade AutoEdit
- Evento OnDataChange
- Evento OnUpdateData
- Evento OnStateChange

Cada uma destas propriedades e eventos são discutidas nas seções a seguir:

Utilizando a propriedade DataSet do Componente DataSource

A propriedade DataSet identifica o nome de um componente DataSet.
Você pode atribuir à propriedade DataSet através de programação ou utilizando o Object Inspector.
O valor atribuido à propriedade DataSet é o nome de um objeto TDataSet.
Por exemplo, a linha de código a seguir atribui um nome ou objeto TQuery à propriedade DataSet de um componente ou objeto DataSource.

```
DataSource1.DataSet  : =  Query1
```

Você pode inserir diversos componentes Query, Table, e StoredProc em um form a atribuir a propriedade DataSet baseado em uma condição no programa. Você também pode atribuir a propriedade DataSet a um nome ou objeto TQuery, TTable, ou TStoredProc encontrado em outro form utilizando o identificador da unit do form. Por exemplo, após incluir a Unit1 na cláusula uses, você pode digitar o seguinte:

```
DataSource1.DataSet  : =  Unit1.Form1.Table1;
```

Utilizando a Propriedade Enable do TDataSource

A propriedade Enable inicia ou termina a comunicação entre os componentes TDataSource e DataControl.
Os valores da propriedade Enable são:

- True

Os componentes Data Control conectados ao TDataSource "enxergam"as alterações do dataset.

- False

Você pode manipular o TDataSet através de programação sem que os componentes Data Control "enxerguem" as alterações.

Por exemplo, o código a seguir desabilita o TDataSource, procura por um valor coincidente de número do cliente, e depois habilita o TDataSource para que ou a linha do cliente ou a última linha seja exibida.

```
DataSource1.Enabled : = False;
Table.First;
   While not  Table1.EOF  do
begin
    if  Table1.FieldByName('NoCliente').AsString = LookupCust then
     Break;
     Table1.Next;
   end;
DataSource1.Enabled  : = True;
```

Utilizando a propriedade Enable permite que você desconecte temporariamente o componente visual Data Control do TDataSource.
No exemplo anterior, se a tabela contiver 2000 linhas e o TDataSource estiver habilitado, o usuário de sua aplicação veria 2000 linhas rolando na tela durante esta operação. Desabilitando TDataSource é uma maneira mais eficiente de se pesquisar em um grande número de linhas, pois o componente Data Control não atrasará a procura exibindo cada linha conforme esta for sendo alterada.

Utilizando a Propriedade AutoEdit de TDataSource

A propriedade AutoEdit controla como a edição é iniciada nos componentes Data Control.
Os valores das propriedades AutoEdit são:

- True

O modo de edição é iniciado sempre que o usuário comece a digitar dentro de um componente Data Control.

- False

O modo de edição é iniciado quando o método Edit é invocado, por exemplo, após o usuário dar um clique sobre o botão **Edit** do Navigator. Este parâmetro controla a edição.

Dentro do seu código, você pode utilizar os seguintes métodos para controlar as alterações nos dados da tabela quando AutoEdit estiver como **False**:

- Edit
- Post
- Cancel

O código a seguir é um exemplo utilizando o método Post:

```
DataSource1.DataSet.Edit;    {Start edit mode}
DataSource1.DataSet.Fields [3] .AsString : = "Hello";
DataSource1.DataSet.Post;
```

Utilizando o Evento OnDataChange de TDataSource

O evento OnDataChange ocorre sempre que:

- A propriedade State do TDataSource mudar do estado dsInactive
- Ocorrer uma alteração de campo, registro, tabela, query ou layout

Este evento é associado com alterações na exibição de dados, tais como rolar para um novo registro ou ativando TDataSource. Este evento é útil para monitorar alterações nos componentes Data Control.

Utilizando o Evento OnUpdateData de TDataSource

O evento OnUpdateData ocorre quando:

- O registro atual no TDataSet estiver para ser atualizado
- Uma alteração estiver para ser confirmada

Este evento é útil na monitoração de alterações nos dados de uma tabela. Por exemplo, você pode utilizar este evento para criar auditor de alterações nos dados em sua aplicação.

Utilizando o Evento OnStateChange de TDataSource

O evento OnStateChange ocorre sempre que a propriedade State de DataSource for alterada. Este evento é útil para monitoração de alterações na propriedade State. A propriedade State pode ter os seguintes valores:

- dsInactive
- dsBrowse
- dsEdit
- dsInsert
- dsSetKey
- dsCalcFields

Passos para Adicionar um Componente DataSource

Execute os passos a seguir para adicionar um componente DataSource ao form:

Passo	Ação
1	Utilizando a página Data Access da Component Palette, adicione o componente DataSource ao form.
2	Utilize o Object Inspector para definir a propriedade DataSet ou escreva o código apropriado.
3	Defina as propriedades AutoEdit e Enabled, ou ambas, apropriadamente.

4	Escreva event handlers para um ou todos os seguintes eventos: • OnDataChange event • OnUpdateData event • OnStateChange event

Utilizando o Objeto TDataSet

Introdução

Geralmente, quando você manipula um componente Table ou chama um método TTable, você está utilizando um método derivado do objeto TDataSet. TDataSet oferece um grande número de propriedade, métodos e eventos.

Você pode aprender mais sobre TDataSet utilizando o Help Online.
Esta seção discute um pouco sobre propriedades, métodos e eventos do objeto TDataSet.

Propriedades do Objeto TDataSet

A tabela a seguir descreve as propriedades mais importantes do objeto TDataSet:

Propriedade	Descrição
Active	Abre ou fecha um componente DataSet. Abrir um componente DataSet estabelece a conexão entre o componente e o banco de dados. Você pode definir a propriedade Active no Object Inspector durante o design ou diretamente em seu código, como segue: *if DeActivate = true then* *Table1.Active : = True* *Else* *Table1.Active : = False;* Os métodos Open e Close também definem a propriedade Active de TDataSet.
EOF (End of File) **BOF (Beginning of File)**	Propriedades somente de leitura (read-only) com os seguintes valores: • EOF é definido para True quando você tenta mover para além da última linha do dataset • BOF é definido para True quando o componente DataSet é aberto ou quando o ponteiro do DataSet para a linha atual estiver posicionado na primeira linha.
Fields	Um array do objeto TField. Você pode definir e ler dados dos campos da linha atual utilizando esta propriedade. Por exemplo: *Table1.Fields[0]. AsString : =* *'Hello';* *Table1.Fields[1] .AsString : =* *'World';*

Métodos do Objeto TDataSet

O objeto TDataSet fornece ao componente Table diversos métodos.
Alguns dos métodos mais importantes são mostrados na tabela a seguir:

Método	Descrição
Open **Close** **Refresh**	Operam nos datasets, como segue: • O método Open é equivalente a definir a propriedade Active para True. • O método Close define a propriedade Active para False

	• O método Refresh permite ler novamente o dataset do banco de dados . Se você precisa se assegurar que os dados são os mais atuais, contendo quaisquer alterações feitas por outros usuários, utilize o método refresh. Um componente Table, Query, ou StoredProc deve estar aberto com open quando Refresh for chamado. Um exemplo destes métodos é : *Table1.Open;* *Table1.Close;* *Table1.Refresh*;
First **Last** **Next** **Prior** **MoveBy**	Permite navegar ou alterar a linha atual do dataset. A seguir, um exemplo utilizando vários destes métodos: *Table1.First* *While not Table1.EOF do* *begin* *{Seu código aqui}* *Table1.Next;* *end;* O método MoveBy move um número especificado de linhas. Por exemplo: • *Table1.MoveBy(3)* move 3 linhas para cima. • *Table1.MoveBy(-2)* move 2 linhas para trás.
Insert **Edit** **Delete** **Append** **Post** **Cancel**	Permite modificar os dados em uma tabela de banco de dados, como segue: • O método Insert permite adicionar uma linha à tabela. Por exemplo: *Table2.Insert;* *Table2.Fields[0] .AsInteger : = 20;* *Table2.Fields[1] .AsString : = 'News';* *Table2.Fields[2] .AsString : = '5 Horas';* *Table2.Post;* • O método Post faz com que a operação Insert Update ou Delete ocorra. • O método Cancel faz com que uma operação Insert Delete, Edit ou Append não ocorrida seja cancelada.
FieldByName	Fornece uma maneira de acessar dados de uma coluna especificando no nome da coluna do banco de dados. Como no exemplo a seguir: *s : = Table2.FieldByName ('area') .AsString;*
SetKey **GotoKey**	Procura através dos datasets, como segue: • SetKey alterna a tabela para o modo de pesquisa (search). Enquanto neste modo, a propriedade Fields tem uma utilização especial. • GotoKey inicia a pesquisa por um valor coincidente com o valor encontrado em Fields[n]. Fields[n] contém o valor a ser pesquisado por valores em outras colunas definindo a coluna Fields correspondente. O exemplo a seguir mostra um exemplo da utilização de SetKey e GotoKey: *Table1.SetKey;* *Table1.Fields[0] .AsString* *Edit1.Text;* *Table1.GotoKey;*
SetRangeStart **SetRangeEnd** **ApplyRange**	Permite ser mais seletivo nos dados que sua aplicação seleciona ou utiliza na tabela. Geralmente uma tabela é grande e você quer selecionar somente uma série de valores da tabela. O método Range permite fazer tal seleção. Exemplo:

	Table1.SetRangeStart *Table1.Fields[0] .AsString : =* *Edit1.Text;* *Table1.SetRangeEnd* *Table.Fields[0] .AsString : =* *Edit2.Text;* *Table1.ApplyRange;* A primeira chamada à SetRangeStart o coloca no modo range e a propriedade Fields toma um significado especial. Utilize a propriedade Fields para especificar o valor de início para a série. A chamada para SetRangeEnd inicia um modo onde os valores digitados na propriedade Fields são utilizadas como o valor final da série. ApplyRange faz com que o comando seja processado. Um dataset é criado contendo os valores entre os valores de início e final.
FreeBookmark **GeTBookMark** **GotoBookmark**	Permite criar um marcador de linha em uma tabela ou query e depois retornar à esta linha posteriormente. Os métodos Bookmark utilizam um tipo de objeto chamado TBookMark. Por exemplo: *Var* *Marker : TBookMark;* *begin* *Marker : = Table2.GeTBookMark;* *Table2.GotoBookmark(Marker);* *Table2.FreeBookmark(Marker);* O método GeTBookMark aloca um marcador para linha da tabela. O método GotoBookmark altera a localização na tabela para a linha indicada pelo Bookmark alocado anteriormente. Utilize o método FreeBookmark para liberar o espaço alocado para o marcador.

Eventos do Objeto TDataSet

O objeto TDataSet permite responder a um grande número de eventos. A tabela a seguir descreve diversos deles:

Eventos	Descrição
OnOpen **OnClose** **OnNewRecord** **BeforeInsert** **AfterInsert** **BeforeEdit** **AfterEdit** **BeforePost** **AfterPost** **OnCancel**	Permite construir e controlar o comportamento de aplicações de bancos de dados. Exemplos de utilização seguem: • Evento BeforePost para validar os campos de um registro antes de inserí-lo ou atualizá-los • Evento AfterPost para gravar um registro de auditoria quando necessário • Evento OnDelete para gravar código que efetue a deleção em cascata quando apropriado

OnDelete	

Utilizando o Objeto TFields

Introdução

O objeto TField, como o objeto TDataSet, não é encontrado na Component Palette. É uma propriedade do objeto TDataSet (e Componente Table). Algumas propriedades no Object Inspector são objetos com seu próprio conjunto de propriedades. TFields é um deles.
Esta seção descreve a propriedade Fields, que é objeto TFields com seu próprio conjunto de propriedades.

Propriedades Fields do Objeto TDataSet

Uma das propriedades do Objeto TDataSet (portanto, os componentes Table, Query, e StoredProc) é a propriedade Fields.
Como discutido anteriormente neste capítulo, a propriedade Fields permite acessar os campos individuais do dataset. A propriedade Fields é um array dos objetos TFields. Este array ou lista é gerada dinamicamente durante a execução (e portanto, não aparece na lista de propriedades do Object Inspector). O array representa cada uma das colunas no componente Table.
Objetos estáticos TFields são visíveis no Object Inspector. Seções posteriores deste capítulo explicam como criar uma lista estática de objetos TField, mas primeiro, você deve entender algumas das propriedades destes objetos.

Propriedades do Objeto TField

Você deve entender em diversos exemplos o uso do método AsString. O objeto TField não faz nenhuma suposição sobre o tipo de dado que ele contém. Ele possui diversos métodos que permitem definir ou recuperar os valores de um campo. Alguns destes métodos são:

- AsBoolean
- AsFlot
- AsInteger
- AsString

O código a seguir mostra alguns exemplos para a utilização de cada um deles:

```
Fields[0] .AsString : = Ísto é uma string';
FieldByName ('Casado') .AsBoolean : = False;
SomaDespesas : = Fields[5] As.Float;
NoPedido ; = Fields[3] .AsInteger;
```

Outras Propriedades do Objeto TField

A tabela a seguir descreve diversas outras propriedades importantes do objeto TField:

Propriedade	Descrição
EditMask	Permite definir uma máscara de input para o campo
IsNull	Determina se um campo não possui valor. É uma propriedade read-only.
Size	Determina o tamanho de um campo. É uma propriedade read-only.
Text	Permite definir ou recuperar um valor de string de um campo
FieldName	Fornece o nome do campo do banco de dados. É uma propriedade read-only.

Utilizando o Componente Table

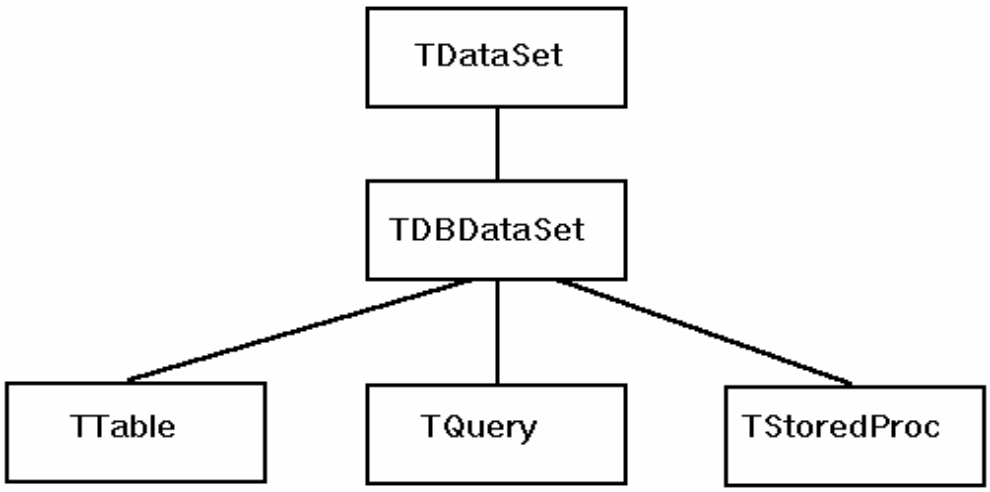
Introdução

O componente Table é um componente TDataSet que comunica com uma tabela de banco de dados através do BDE. A tabela do banco de dados pode ser tanto local ou em um servidor remoto. Esta seção discute:

- O papel do componente Table
- Propriedades, eventos e métodos do componente Table
- Passos para adicionar um componente Table à sua aplicação

O Papel do Componente Table

O diagrama a seguir mostra que TTable é derivado de TDBDataSet por herança e, por sua vez, TDBDataSet é derivado de TDataSet.
Muito da funcionalidade do componente Table é baseado neste relacionamento.
TTable herda de TDataSet a capacidade de manipular datasets. Ele fornece métodos, tais como Next, First, Last, Edit, SetRange e Insert.
TDbDataSet permite que TTable suporte trabalhar com senhas e outras tarefas associadas com a ligação de sua aplicação com um banco de dados. TTable adiciona métodos e atributos para manipulação de índices, e para armazenar o relacionamento entre esta tabela e outra em sua aplicação .



Propriedades do Componente Table

Como vimos, muito da funcionalidade do componente Table vêm do objeto TDataSet. O componente Table permite utilizar as propriedades, métodos e eventos TDataSet, mas possui diversas propriedades próprias relativas a tabelas de banco de dados. Por exemplo, o componente Table permite que você:

- Especifique índices a serem utilizados
- Crie um cursor "Linkado"

Cursores "Linkados" coordenam dois ou mais componentes DataSet para criar forms master-detail.
A tabela a seguir descreve as propriedades mais importantes do componente Table:

Propriedade	Descrição
DatabaseName	Especifica o seguinte: • O diretório local de um banco de dados que contenha a tabela a ser visualizada • O alias de um banco de dados remoto
TableName	Especifica o nome do banco de dados a ser visualizado
Exclusive	Controla o acesso de usuário ao banco de dados. Os valores são: • True Assegura que nenhum outro usuário acesse ou modifique a tabela enquanto você a mantiver aberta • False Permite que outros usuários acessem ou modifiquem a tabela enquanto você a mantiver aberta. Este é o default.
IndexName	Identifica um índice secundário para Table. Você não pode alterar IndexName enquanto a tabela estiver ativa
MasterFields	Especifica o nome dos campos ligados ou campos na propriedade MasterFields para criar um cursor "linkado"a uma tabela secundária Para especificar diversos campos, você deve separar os nomes de campos com uma barra vertical (|) .
MasterSource	Especifica o TDataSource de onde TTable obterá os dados para a tabela master
ReadOnly	Põe a tabela em modo somente-leitura. Os valores são: • True Previne o sistema de gravar alterações ao banco de dados onde a tabela resida • False Permite que o sistema grave alterações ao banco de dados onde a tabela resida

Você não pode alterar a propriedade ReadOnly enquanto a tabela estiver ativa.

Passos para Adicionar um Componente Table

Execute os passos a seguir para adicionar um componente table ao form:

Passo	Ação
1	Utilizando a página Data Access da Component Palette, adicione um componente Table ao form.
2	No Object Inspector , localize a propriedade DatabaseName e digite o diretório onde o banco de dados reside, ou digite um nome de alias do banco de dados.
3	Localize a propriedade TableName e digite o nome da tabela ou selecione uma tabela da lista drop-down.
4	Adicione um componente DataSource e defina o valor da propriedade de DataSet igual ao do componente Table.
5	Adicione componentes Data Control e conecte-os ao componente DataSource para exibir dados da tabela do banco de dados.

Tutorial: Criando uma Aplicação Utilizando Métodos TDataSet do Componente Table

Introdução

Este processo é um tutorial de exemplo. Você construirá uma aplicação de exemplo utilizando uma tabela chamada COUNTRY.
Esta tabela exibe informação sobre países do mundo inteiro. Ao invés de utilizar o DBNavigator, você utilizará botões padrão e métodos DataSet para fornecer a funcionalidade do DBNavigator.
Este tutorial mostra como utilizar:

- Os métodos First, Next, Prior e Last
- Os métodos BOF e EOF
- Os métodos Edit, Insert e Cancel

Estágios do Tutorial

O processo deste tutorial envolve os seguintes estágios:

Estágio	Processos
1	Adicionar e definir propriedades para os componentes TDataSet
2	Adicionar e definir propriedades para componentes DBGrid e Button
3	Criar event handlers OnClick para componentes Button
4	Executar e testar a aplicação

Passos para o Estágio 1

Execute os passos a seguir para adicionar e definir propriedades para os componentes TDataSet:

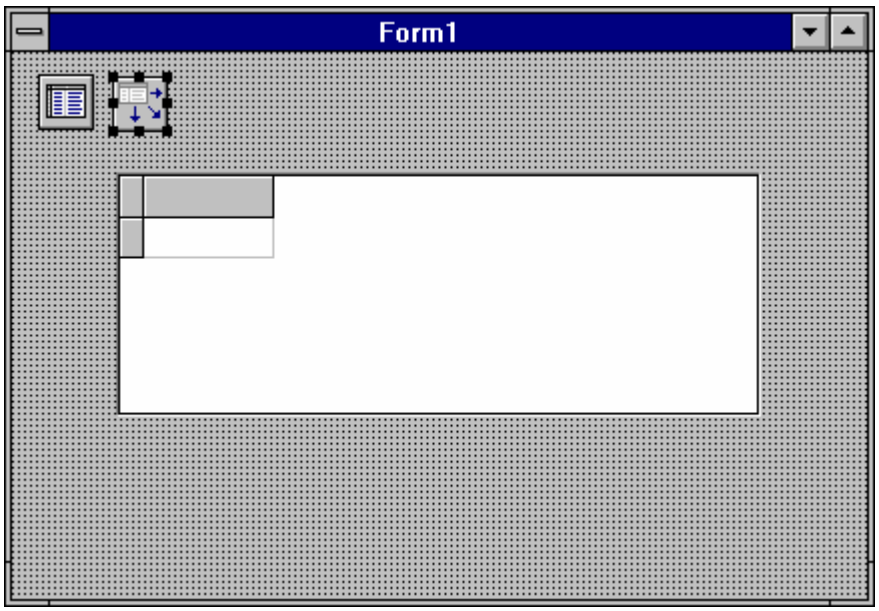
Passo	Ação
1	Abra um novo projeto e grave-o. Quando solicitado, grave a unit como UDSEVENT.PAS e o projeto como PDSEVENT.DPR.
2	Utilizando a página Data Access da Component Palette, adicione o seguinte ao seu form: • Um componente Table • Um componente DataSource
3	Defina as propriedades para os componentes Table e DataSource como segue:

Nome do Componente	Propriedade	Valor
Table1	DatabaseName TableName Active	DBDEMOS COUNTRY.DB True
DataSource1	DataSet AutoEdit	Table1 False

Passos para o Estágio 2

Execute os passos a seguir para adicionar e definir propriedades para os componentes DBGrid e Button:

Passo	Ação
1	Utilizando a página Data Control da Component Palette, adicione um componente DBGrid. Arranje os componentes para que seu form esteja similar à figura a seguir:



Passo	Ação
2	Utilize a tabela a seguir para definir propriedades do DBGrid para que utilizem o componente DataSource do form:

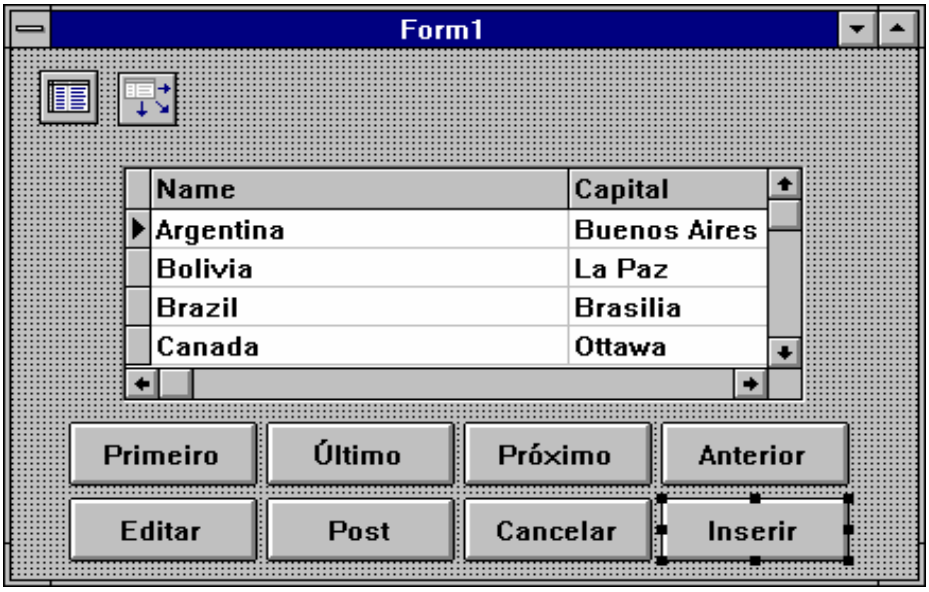
Nome do Componente	Propriedade	Valor
DBGrid1	DataSource	DataSource1

Passo	Ação
3	Adicione oito botões ao form, e arranje-os para que seu form esteja similar à próxima figura:

Passo	Ação
4	Utilize o Object Inspector para definir as seguintes propriedades aos componentes Button:

Nome do Componente	Propriedade	Valor
Button1	Caption	Primeiro
Button2	Caption	Último
Button3	Caption	Próximo
Button4	Caption	Anterior
Button5	Caption	Editar
Button6	Caption	Post
Button7	Caption	Cancelarr

Button8	Caption	Inserir



Passos para o Estágio 3

Execute os passos a seguir para criar event handlers OnClick para os componentes Button:

Passo	Ação
1	Digite o código a seguir para os event handlers OnClick nos botões apropriados.

Nome do Componente	Evento	Código
Button1	OnClick	Table1.First;
Button2	OnClick	Table1.Last;
Button3	OnClick	if not Table1.EOF then Table1.Next;
Button4	OnClick	if not Table1.BOF then Table1.Prev;
Button5	OnClick	Table1.Edit;
Button6	OnClick	if Table1.State in [dsEdit, dsInsert] then Table1.Post;
Button7	OnClick	Table1.Cancel;
Button8	OnClick	Table1.Insert;

Passos para o Estágio 4

Execute os passos a seguir para executar e testar a aplicação:

Passo	Ação
1	Compile e grave a aplicação. Execute e teste cada botão para verificar se o método funciona.
2	Quando tiver completado o teste, grave e feche o projeto.

Utilizando o Fields Editor

Introdução

O Fields Editor permite criar uma lista de campo de banco de dados.
Quando um componente DataSet como os componentes Table ou Query é ativado pela primeira vez, uma lista de campo é gerada dinamicamente para dataset baseado nas colunas da tabela ou código SQL. O Fields Editor permite especificar e posteriormente modificar uma lista estática de componentes Field.

Possibilidades do Fields Editor

O Fields Editor permite que você:

- Crie um modelo estático das colunas de uma tabela
- Especificar a ordem das colunas no DataSet
- Especificar o tipo das colunas
- Adicionar a lista estática de campos
- Remover campos da lista
- Modificar propriedades Display de TFields estáticos
- Definir campos calculados
- Definir novos componentes Field baseados nas colunas existentes na tabela

Criando um Modelo Estático de uma Tabela de Banco de Dados

Utilize o Fields Editor quando quiser criar um modelo estático das tabelas do banco de dados. Um modelo estático não é alterado quando modificações são feitas na tabela física no banco de dados.
Quando você adiciona colunas utilizando o Fields Editor, objetos TFields são criados para cada campo adicionado ao DataSet. Após adicionar campos utilizando o Fields Editor você pode visualizar estes campos no Object Inspector. Cada objeto TField possui um conjunto de propriedades, eventos e métodos que você pode utilizar em sua aplicação.

Passos para iniciar o Fields Editor

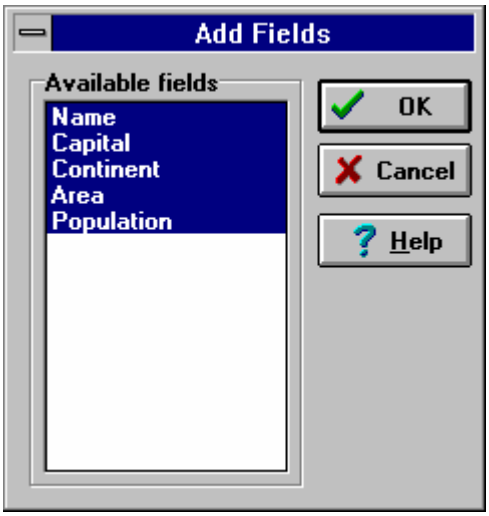
Execute os passos a seguir par iniciar o Fields Editor:

Passo	Ação
1	Adicione um componente Table ou Query ao seu form.
2	Defina a propriedade DatabaseName do componente Table ou Query.
3	Execute um dos seguintes: -Defina a propriedade TableName do componente Table -Defina a propriedade SQL do componente Query.
4	Selecione o componente DataSet no form, e pressione o botão direito do mouse para exibir o SpeedMenu.
5	No SpeedMenu, selecione Fields editor. A primeira janela do Fields Editor aparecerá, como segue:

Passos para Criar uma Lista Estática de Campos

Execute os passos a seguir para criar uma lista estática de campos após abrir o Fields Editor:

Passo	Ação
1	No quadro de diálogo Fields Editor, dê um clique em Add. Cada uma das colunas na tabela ou query aparecem selecionadas no quadro de diálogo Add Fields, como segue:

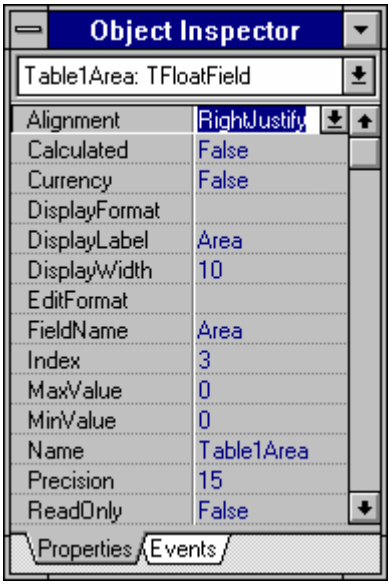


Passo	Ação
2	Selecione os campos que você queira adicionar e dê uma clique em OK. O quadro de diálogo a seguir é exibido:

Passo	Ação
3	Dê um clique em Add para adicionar campos estáticos, adicionar à lista de campos no dataset.
4	Dê um clique em Define para criar um novo campo baseado em um campo existente, ou criar um campo calculado.
5	Dê um clique em Remove para deletar um campo estático da lista de campos no dataset.

Propriedades do Componente Field

Após utilizar o Fields Editor, quaisquer campos adicionados ao dataset são refletidos no Object Inspector. A figura a seguir mostra o tipo de objeto TStringField (componente Field) e suas propriedades associadas.



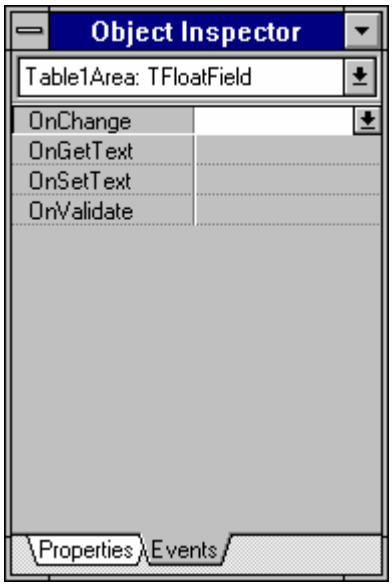
Descendentes de TField

Quando você utiliza o Fields Editor, o Delphi cria objetos estáticos que correspondem aos campos visíveis no Object Inspector. Estes objetos são descendentes do tipo de objeto TField. A tabela a seguir descreve os objetos descendentes TField:

Descendente	Descrição
TStringField	Dado texto de tamanho fixo, até 255 caracteres
TIntegerField	Números inteiros de -2,147,483,648 a 2,147,483,647
TSmallField	Números inteiros de -32678 a 32767
TWordField	Números inteiros de 0 a 65535
TFloaTField	Números reais com grandezas absolutas de 1,2x10e-324 a 1.7x10e308 com precisão de 15 a 16 dígitos
TCurrencyField	Valores monetários. Números reais com grandezas absolutas de 1.2x10e-324 a 1.7x10e308 com precisão de 15 a 16 dígitos
TBCDField	Números reais com número fixo de casas decimais. Com precisão de 18 dígitos. O intervalo dos valores depende do número de casas decimais.
TBooleanField	Valor True (verdadeiro) ou False (falso)
TDateTimeField	Valor de data e hora
TDateField	Valor de data
TTimeField	Valor de tempo
TBlobField	Campo de dado arbitrário, sem limite de tamanho
TBytesField	Campo de dado arbitrário, sem limite de tamanho
TVarBytesField	Campo de dado arbitrário de até 65535 caracteres, com tamanho real armazenado nos primeiros dois bytes
TMemoField	Texto de tamanho arbitrário
TGraphicField	Gráfico de tamanho arbitrário, tal como bitmap

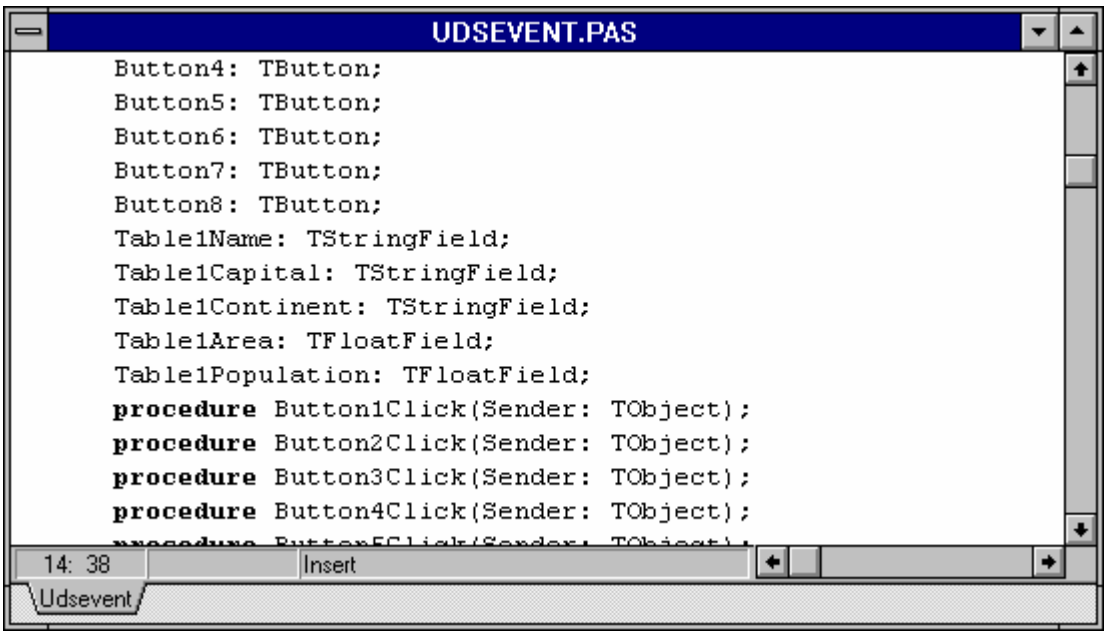
Eventos do Objeto Tipo TStringField

A figura a seguir mostra eventos que um objeto tipo TStringField pode reconhecer:



Campos Estáticos

Quando você utiliza o Fields Editor, são adicionados campos estáticos ao tipo de objeto TForm1 do form .
Na figura a seguir, o Fields Editor adicionou os seguintes campos ao TForm 1:
Table1Name, Table1Size, Table1WIGHT, Table1AREA, e Table1BMP:



Utilizando o Componente Database Grid

Introdução

Como você já pode ter observado, o componente DBGrid (databasegrid) fornece uma maneira conveniente de se exibir diversas linhas de dados de um componente Table ou Query. Sua aplicação pode utilizar o DBGrid para inserir, deletar, editar ou exibir dados de um banco de dados. Combinado com o DBNavigator, o DBGrid permite protipar e exibir rapidamente dados do banco de dados.
Até agora, você tem visto exemplos utilizando o componente DBGrid.

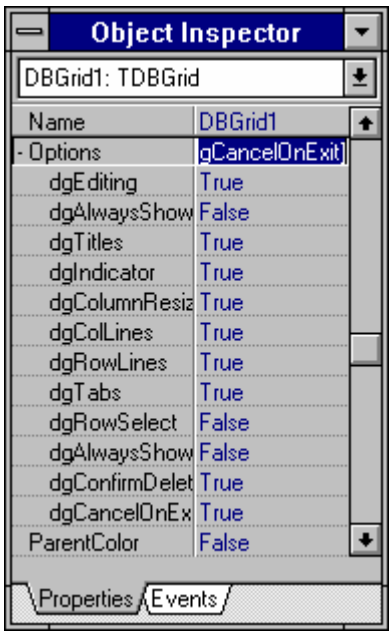
Este tópico oferece sugestões para:

- Definir a propriedade Options do Componente DBGrid
- Definir caracteristicas de exibição de campo para o componente DBGrid

Uma lista completa das propriedades, métodos e eventos do DBGrid encontram-se no Help Online.

Propriedades de Options do DBGrid

A figura a seguir mostra um conjunto de propriedades que compõem a propriedade Options do componente DBGrid. O conjunto aparece quando você clica o sinal (+) na frente da propriedade Options.



Descrição da Propriedade Options

Você pode alterar a aparência e comportamento de uma grade alterando valores da propriedade Options, utilizando o Object ou escrevendo código. A tabela a seguir descreve as definições de Options do componente DBGrid:

Propriedade Options	Descrição Quando Definido para True...
dgEditing	O usuário pode editar dados na grade. Quando a propriedade ReadOnly do DataSet for True e dgEditing também for True, os usuários podem utilizar a tecla Insert para inserir uma linha em branco, ou pressionar a tecla de seta para baixo quando posicionado no final da grade para adicionar uma linha em branco, embora não possam digitar texto na nova linha.
dgTitles	Os títulos das colunas são visíveis.
dgIndicator	Um pequeno ponteiro fica visível para indicar a coluna atual.
dgColumnResize	As colunas podem ser reajustadas.
dgColLines	As linhas entre as colunas ficam visíveis
dgRowLines	As linhas entre as linhas ficam visíveis.
dgTabs	Os usuários podem pressionar a tecla Tab e Shift+ Tab para se moverem entre as colunas da grade.

Propriedade Options como Tipo Set

A propriedade Options do componente DBGrid é um tipo set. Você pode modificar a propriedade Options utilizando operadores set. As linhas a seguir são comandos Object Pascal válidos:

DBGrid1.Options : = DBGrid1.Options + [dgTitles];
DBGrid1.Options : = DBGrid1.Options - [dgTitles, dgRowLines];

Definindo Características de Exibição de Campo para o Componente DBGrid

Geralmente no desenvolvimento de uma aplicação, você precisa controlar o comportamento de campos no componente DBGrid. O comportamento default do DBGrid é determinar dinamicamente o tamanho do campo e permitir ao usuário o uso do mouse para reajustar o tamanho do campo. A chave para obter o controle das características de exibição do DBGrid ou outro componente data-aware é criar uma lista

estática de componentes Field. Uma vez criados componentes para cada um dos campos no dataset, você pode definir o seguinte:

- Tamanho de exibição
- Formato de exibição
- Máscara de Edição
- Rótulos de exibição

Passos para Definir Tamanho de Exibição

Execute os passos a seguir para criar campos de exibição de tamanho fixo no componente DBGrid:

Passo	Ação
1	Abra o Fields Editor para o componente TDataSet que será exibido no componente DBGrid.
2	Adicione cada um dos campos de banco de dados que você queira no dataset. Este passo cria componentes TFields estáticos para os campos a serem exibidos no DBGrid.
3	Localize o componente DBGrid no Object Inspector, e defina a opção dgColumnReSize para False. Como alternativa, você pode escrever comandos Object Pascal em sua aplicação para alterar esta propriedade.
4	Altere o tamanho da coluna exibida de uma destas maneiras: -Utilize o mouse para arrastar e reajustar o tamanho das colunas no DBGrid -Defina a propriedade DisplayWidth para cada um dos componentes Field que o Fields Editor adicionou.

Definindo a Propriedade DisplayLabel

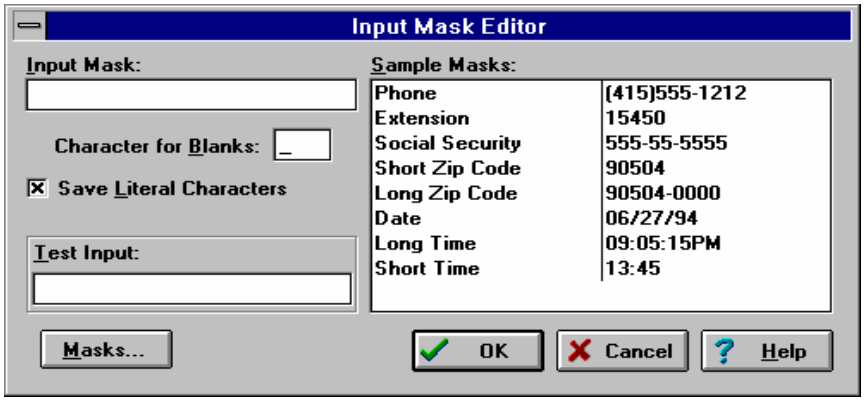
Uma vez que você utilizou o Fields Editor para gerar um conjunto de objetos TField para o dataset, você pode utilizar o Object Inspector para definir a propriedade DisplayLabel do componente Field. O componente DisplayLabel do componente Field.

Definindo Propriedade DisplayMask

Campos, Float, Integer e Date possuem uma propriedade DisplayMask. Você pode utilizar esta propriedade para formatar a exibição em um DBGrid ou outro componente Data Control. Por exemplo, o formato de exibição mm-dd-yy pode ser utilizado para exibir um campo data.

Definindo a Propriedade EditMask

Os componentes Field possuem uma propriedade EditMask que você pode definir sempre que digitar dados em um DBGrid ou outro componente Data Control. Para definir uma propriedade EditMask, localize o componente Field no Object Inspector, e clique a propriedade EditMask. O quadro de diálogo Input Mask Editor é exibido, como segue:



Para testar sua máscara de edição, digite um valor no campo **Test Input**.

Utilizando o Componente Query

Introdução

Até agora, este capítulo cobriu as seguintes informações:

- As capacidades do objeto DataSet e suas propriedades, métodos e eventos
- O Fields Editor, que permite definir objetos correspondentes aos campos no dataset

O componente Query, como Table, é derivado do objeto TDataSet.Desta forma, tudo sobre TDataSet aplica-se ao componente Query.
Esta seção explica o seguinte:

- O componente Query
- Propriedades, métodos e eventos importantes do componente
- Uma aplicação de exemplo utilizando o componente Query

Componente Query

O componente Query permite utilizar comandos SQL para executar o seguinte:

- Especificar ou criar datasets que possam ser exibidos
- Inserir linhas
- Editar a atualizar colunas
- Deletar linhas

O componente Query gerencia a comunicação com o BDE e serve como interface entre o BDE e os componentes DataSource (TDataSource) em seus forms

O Componente Query Relacionado com o Componente DataSource

Como com o componente Table, um componente DataSource é anexado ao componente Query para gerenciar a comunicação entre o componente Data Control e o componente Query. Entre os componentes Data Control incluem DBGrid, DBEdit, e DBLooup.
Uma aplicação típica possui um componente DataSource para cada componente Query.

Propriedade do Componente Query

A tabela a seguir descreve diversas propriedades importantes do componente Query:

Propriedade	Descrição
Active	Abre ou fecha uma query. Os valores são: • True Abre uma query, o que faz com que o comando SQL seja executado, como no exemplo: *{Abre a query}* *Query1.Active : = True;* • False Fecha uma query, como segue: *{Fecha a query}* *Query1.Active : = False;*
DatabaseName	Identifica o alias do banco de dados ou o drive e diretório de um banco de dados local. A propriedade DatabaseName pode ser definida somente quando a query não estiver ativa, como no exemplo a seguir: *{Fecha o DBDataSet}* *Query1.Active : = False;* *Query1.DatabaseName : = 'Demos'*; *Query1.Active : = True;*
Fields	Suportam os campos no componente Query. É uma propriedade somente durante execução

	e é utilizada para examinar ou modificar um determinado campo, como no exemplo a seguir: *Query1.Fields[3] .AsString : = ÓK';*
DataSource	Fornece valores para queries parametrizadas. Uma query parametrizada é uma onde um ou mais valores na condição de seleção não é conhecida
Params	Guardam os parâmetros para uma query parametrizada. Uma query parametrizada envolve um ou mais valores na condição de seleção que não são conhecidas até a execução, como no exemplo a seguir: *Select * from Orders* *Where CustNo = : SomeNo* Esta é uma propriedade de somente-leitura, durante a execução. Consulte o Help Online para maiores informações sobre queries parametrizadas.
SQL	Guarda o texto do comando de query SQL
EOF (End of File) **BOF (Beginning of File)**	Propriedade somente-leitura com valores a seguir: • EOF é True quando você tenta mover para além da última linha do dataset. • BOF é True quando o componente DataSet é aberto, ou quando o ponteiro do TDataSet da linha atual estiver na primeira linha.

Métodos do Componente Query

A tabela a seguir descreve alguns métodos do componente Query:

Método	Descrição
ExecSQL	Executa comando SQL atribuido à propriedade SQL se o comando não retornar dados. Quando estiver inserindo, atualizando ou deletando dados, você deve utilizar este método. Se estiver executando um comando de seleção, utilize o método Open. A seguir um exemplo da utilização do método ExecSQL: *Query1.Close;* *Query1.Clear;* *Query1.SQL.Add ('Delete emp where empno = 1010');* *Query1.ExecSQL;*
Open	Abre o componente Query. É equivalente a definir a propriedade Active para True. A seguir um exemplo utilizando o método Open: *Query1.Open;*
Close	Fecha o componente Query fazendo com que quaisquer atualizações pendentes sejam efetuadas no banco de dados. Chamar Close é equivalente a definir a propriedade Active para False. A seguir mostramos um exemplo utilizando o método Close: *Query1.Close;*
Prepare	Traduz a propriedade SQL para criar a propriedade Text para Submeter ao servidor. O método Prepare também envia a requisição ao servidor para propósito de otimização, embora nenhum valor parametrizado esteja incluido. A requisição inteira com parâmetros não é submetida até que o método Open ou ExecSQL sejam chamados. Se você não chamar Prepare explicitamente, o Delphi chama Prepare implicitamente quando utilizar o comando em ExecSQL. A seguir um exemplo utilizando o método Prepare: *Query1.Close;* *Query1.SQL : =* *'Delete emp where empno = : empno';* *Query1.Prepare;*

Métodos TQuery Herdados de TDataSet

O objeto TDataSet fornece ao componente Query uma grande variedade de métodos. Alguns dos mais importantes são mostrados na tabela a seguir:

Método	Descrição
First **Last** **Next** **Prior** **MoveBy**	Permite navegar ou alterar a linha atual do dataset. A seguir um exemplo utilizando diversos destes métodos: *Query1.First* *While not Query1.EOF do* *begin* *{Seu código aqui}* *Table1.Next;* *End;* O método MoveBy move um número determinado de linhas. Por exemplo: • Query1.Moveby(3) move 3 linhas para cima. • Query1.Moveby (-2)move 2 linhas para trás
Insert **Edit** **Delete** **Append** **Post** **Cancel**	Permite modificar o conjunto resultante de uma query. O método Insert permite adicionar linhas à tabela, como no exemplo a seguir: *Query2.Insert;* *Query2.Fields [0] .AsInteger : = 20;* *Query2.Fields [1] .AsString : = 'News';* *Query2.Fields [2] .AsString : = '5 horas';* *Query2.Post;* • O método Post faz com que as operações Insert, Update, ou Delete ocorram. • O método Cancel faz com que um Insert, Delete, Edit ou Append não completado seja cancelado.
SetKey **GotoKey**	Pesquisa através dos datasets, como segue: • O método SetKey alterna o dataset para o modo de procura. Neste modo, a propriedade Fields tem uso especial. • O método GotoKey inicia uma procura por um valor que coincida com o valor encontrado em Fields[n]. Fields[n] contém o valor que você está procurando e que ocorre na primeira coluna da tabela ou dataset. Você pode procurar por valores em outras colunas definindo a coluna correspondente. O exemplo a seguir mostra a utilização dos métodos SetKey e GotoKey: *Query1.SetKey;* *Query1.Fields [0] .AsString :* = Edit1.Text; *Query1.GotoKey;*
FreeBookmark **GeTBookMark** **GotoBookmark**	Permite criar um marcador em uma linha na tabela ou query e retornar posteriormente para esta linha, como segue: • O método FreeBookmark libera espaço alocado para o marcador • O método GeTBookMark aloca um marcador para a linha atual da tabela. • O método GotoBookmark altera a localização na tabela para a linha indicada por um marcador alocado préviamente . Os métodos Bookmark utilizam o objeto tipo TBookMark, como no exemplo a seguir: *Var* *Marker : TBookMark;* *begin* *Marker : = Query2.GeTBookMark;* *Query2.GotoBookmark (Marker);* *Query2.FreeBookmark (Marker);*

Eventos do Componente Query Derivados de TDataSet

O componente Query responde aos eventos herdados do objeto TDataSet. A tabela a seguir descreve estes eventos:

Eventos	Descrição
OnOpen **OnClose** **OnNewRecord** **BeforeInsert** **AfterInsert** **BeforeInsert** **AfterEdit** **BeforePost** **AfterPost** **OnCancel** **OnDelete**	Permite construir e controlar o comportamento da aplicação de banco de dados. Por exemplo: • O event handler BeforePost valida os campos de um registro antes de inserir ou atualizar os dados • O evento AfterPost é útil para gravar um registro de auditoria quando necessário. • O evento OnDelete é útil para escrever código que efetue a deleção em cascata quando apropriado.

Tutorial: Criando uma aplicação Utilizando o Componente Query

Introdução

Esta seção fornece um aprendizado na utilização do componente Query. O componente Query possui muitas características avançadas. Entretanto, este exemplo focaliza as características básicas deste componente.
Este tutorial demonstra como utilizar um componente Query para criar um dataset, e métodos TDataSet para executar operações no dataset.

Estágios do Tutorial

Este tutorial envolve os seguintes estágios:

Estágio	Processo
1	Construir uma aplicação e exibir os dados utilizando um comando SQL
2	Modificar uma aplicação para consultar um banco de dados, baseado no campo CustID

Passos para o Estágio 1

Execute os passos a seguir para construir uma aplicação de clientes e exibir dados do cliente utilizando um comando SQL:

Passo	Ação
1	Abra um novo projeto e grave-o. Quando solicitado, grave a unit como EX7SQL.PAS e o projeto como EX7P.DPR.
2	Utilizando a página Data Access page da Component Palette, adicione os seguintes ao seu form: • Um componente Query • Um componente DataSource
3	Defina as propriedades dos componentes Query e DataSource, como segue:

Nome do Componente	Propriedade	Valor
Query1	DatabaseName RequestLive	DBDEMOS True
DataSource1	DataSet	Query1

Passo	Ação
4	Utilizando a página Data Controls da Component Palette, adicione os seguintes ao seu form: • Um componente DBGrid Um componente DBNavigator
5	Utilize a tabela a seguir para definir propriedades dos componentes DBNavigator e DBGrid para que utilizem o componente DataSource do form:

Nome do Componente	Propriedades	Valor
DBGrid1	DataSource	DataSource1
DBNavigator	DataSource	DataSource1

6	Crie um event handler OnActivate para o form utilizando o Object Inspector. Digite o código abaixo no handler OnActivate: *Query1.SQL.Add ('Select * from customer');* *Query1.Open;* Este código adiciona uma instrução SQL à propriedade SQL do componente Query. Você pode definir esta instrução SQL como uma propriedade utilizando o Object Inspector. Após adicionar a instrução SQL, seu event handler deve ser: *Query1.SQL.Add('Select * From Customer');* *Query1.Open;*
7	Quando sua aplicação for compilada satisfatoriamente. grave-a. Depois, execute sua aplicação e teste-a utilizando o Navigator para atualizar e inserir linhas no dataset.
8	Quando tiver terminado, feche a aplicação.

Passos para o Estágio 2

Execute os passos a seguir para modificar uma aplicação para consultar uma base de dados, baseado em uma identificação do cliente:

Passo	Ação
1	Utilizando a página Standard do Component Palette, adicione os seguintes ao form: • Dois componentes botão • Um componente Edit • Um componente Label
2	Utilize a informação da tabela a seguir para definir propriedades destes componentes:

Nome do Componente	Propriedade	Valor
Button1	Caption	Query
Button2	Caption	Exec Query
Edit1	Text Visible	(Empty) False
Label	Caption Visible	IdCliente False

Passo	Ação
3	Adicione o seguinte event para o evento OnClick do Button1: *Edit1.Visible : = True;* *Label1.Visible : = True;*
4	Adicione o seguinte event handler para o evento OnClick do Button2

Query1.Close;
Query1.SQL.Clear;
Query.SQL.Add
*('Select * from customer where '+'CustNo = '+ Edit1.Text);*
Query.Open;
Edit1.Visible : = False;
Label1.Visible : = False;

Passo	Ação
5	Compile e grave sua aplicação. Execute e teste-a através do seguinte: • Selecione um número de cliente da lista de clientes e dê um clique em Query. • Digite o número e dê um clique em Exec Query para exibir um único cliente no grid.

Utilizando o Visual Query Builder

Introdução

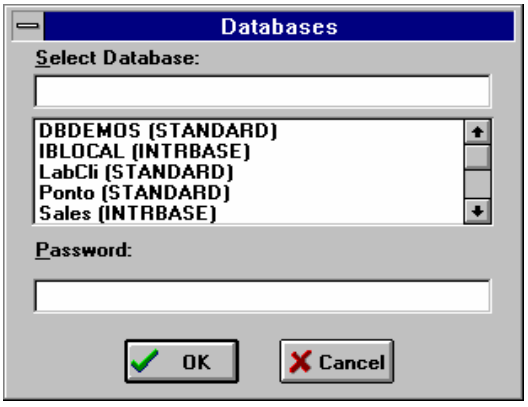
O Visual Builder (VQB) é uma ferramenta para construção de queries baseadas em SQL. Com esta ferramenta, você pode construir queries complexas com pouco ou nenhum conhecimento de SQL. O VQB permite construir estas queries complexas iniciando com um query simples, permitindo executar a query e fornecer ferramentas para refiná-la . Você constrói queries incrementalmente adicionando expressões, tabelas, campos e relacionamentos até obter os resultados desejados.

Ativando o Visual Query Builder

Para ativar o Visual Query Builder , você deve utilizar um componente Query. Uma vez inserido um componente Query em seu form, você pode ativar o Visual Query Builder selecionando o componente Query e dando um clique com o botão direito do mouse. O SpeedMenu do Componente aparecerá. Então, selecione **Query Builder**.

Selecionando um Alias de banco de dados

O item de menu **Query Builder** exibe o quadro de diálogo Databases, exibido na figura a seguir. Este quadro de diálogo permite selecionar um banco de dados e logar-se nele. O logon pode ser em um servidor de banco de dados local ou remoto durante o design.



Janela do Visual Query Builder

Após logar-se ao banco de dados, a janela do Visual Query Builder é apresentada. O quadro de diálogo Add sobrepõe-se.

Toolbar do Visual Query Builder

A Toolbar do Visual Query Builder aparece para selecionar as operações a serem executadas. A Toolbar aparece, como segue:

Descrição da Toolbar do Visual Query Builder

A tabela a seguir descreve os botões da Toolbar do Visual Query Builder respectivamente:

Botão	Descrição
New	Inicia uma nova query
Open	Abre um arquivo de query
Save As	Grava uma query em um arquivo
Options	Exibe o quadro de diálogo Options, que permite definir diversas opções de query. Por exemplo, você pode definir uma opção para remover linhas duplicadas.
Table	Exibe o quadro de diálogo Add Table, que permite adicionar tabelas à instrução SQL
Expression	Exibe o quadro de diálogo Expression, que permite criar expressões SQL, por exemplo, upper (Nome) ou Sum (Custo_Item)
SQL	Exibe a janela SQL Statement, que contém, a instrução SQL atual.
Run	Executa a instrução SQL atual e exibe os resultados
OK	Define a propriedade SQL do componente Query para a instrução SQL atual no Visual Query Builder
Cancel	Sai do Visual Query Builder sem definir a propriedade SQL do componente Query
Help	Exibe o Help online do Visual Query Builder

Quadro de Diálogo Add Table

Quando você abre o Visual Query Builder, o quadro de diálogo Add Table aparece. Este quadro permite adicionar tabelas à query. Você adiciona tabela à query quando inicia o processo de construção da query ou quando quer modificar uma query existente.
O quadro de diálogo Add Table lista os nomes de todas as tabelas no banco de dados atual. Se você quiser incluir tabelas de sistema, dê um clique no check box **Include System Tables.**

Passos para Adicionar Tabelas à Query

Execute os passos a seguir para adicionar uma ou mais tabelas ao espaço de trabalho de Visual Query Builder a serem incluídas na query:

Passo	Ação
1	Insira um componente Query ao form, e dê um clique com o botão direito do mouse sobre o componente para exibir o SpeedMenu.
2	Selecione Query Builder para exibir a janela do Visual Query Builder.
3	Se o quadro de diálogo Add Table não aparecer na frente da janela do Visual Query Builder, dê um clique sobre o botão Table na Toolbar para exibi-lo.
4	Selecione o nome da tabela da lista de tabelas exibida no quadro de diálogo Add Table, e dê um clique em Add. A tabela aparece no espaço de trabalho da janela do Visual Query Builder.
5	Repita o passo 4 até que todas as tabelas sejam adicionadas à query, e dê um clique em Close.

Adicionando Colunas à Query

Para adicionar uma coluna de uma das tabelas à query, você pode efetuar um dos seguintes:

- Selecionar o nome da coluna e arrastar a coluna e soltá-la na grade da query, na parte inferior da janela.
- Dê um duplo-clique no nome da coluna para inserí-la na grade da query.

Especificando uma Condição de Join

Geralmente você precisará combinar informações de diversas tabelas. Por exemplo, você pode querer consultar o banco de dados para encontrar todos os pedidos , o nome dos clientes, a data de pedido e o fornecedor de um dos itens. Esta query envolverá colunas das tabelas CUSTOMER, VENDORS, ITEMS e ORDERS. Para construir uma query deste tipo e complexidade, você precisará especificar como as diversas tabelas deverão ser combinadas. Com o Visual Query Builder você especifica as colunas a serem combinadas arrastando o nome de uma coluna e soltando-a sobre o nome da coluna onde a combinação será feita. Quando completar esta operação, uma linha será desenhada no espaço de trabalho da query, ligando as colunas das duas tabelas.

Revisando e Editando o Critério de Combinação

Você pode revisar e editar o critério de combinação dando um duplo-clique sobre a linha indicando a combinação no espaço de trabalho da query. O quadro de diálogo Join aparecerá.

Especificando Critério de Seleção de uma Query

Para especificar um critério de seleção de uma query, utilize a linha Criteria da grade da query.

Se a linha Criteria não estiver visível, utilize a barra de rolagem.
A linha Criteria permite qualquer expressão válida de query dentro da cláusula WHERE de uma instrução SQL. Expressões válidas incluem os seguintes operadores:

Operador	Significado
=	Igual
>	Maior que
<	Menor que
!=	Não igual
like	Comparação de string de caracteres com separação de padrão
between	Não menor que o valor inicial e não maior que o valor final
in	Encontrado em uma lista

Expressões digitadas na linha Criteria são condições AND. Por exemplo, se você tiver uma tabela com um nome de coluna e quiser encontrar todas as entradas na coluna nome que iniciem com C, você digitaria nome like ‘C. /. ‘na linha Criteria. O efeito seria equivalente a adicionar AND nome LIKE ‘C . /.’à cláusula WHERE da instrução SQL. Condições OR São digitadas como expressões na linha OR (abaixo da linha Criteria) da grade da query.

Ordenando Resultados da Query

Você pode ordenar resultados da query na ordem ascendente ou descendente em uma determinada coluna. Para especificar a ordem de ordenação, posicione o ponteiro do mouse sobre a coluna, na linha Sort da

grade da query. Dê um clique com o botão direito do mouse e selecione. Ascending ou Descending no menu pop-up. Você pode especificar até oito colunas para ordenação.

Agrupando Resultados de Query

Você pode agrupar resultados de query utilizando a linha Option da grade da query. Para especificar uma opção para uma coluna, posicione o ponteiro do mouse sobre a coluna na linha Option. Dê um clique com o botão direito do mouse para exibir o SpeedMenu Option. As opções são funções de agrupamento: Show, Average, Count, Maximum, Minimum, Sum, Group (1), e Group (2).

Especificando Condições de Agrupamento

Você pode especificar condições de agrupamento utilizando a linha Group Condition da grade da query. Condições de agrupamento especificam operações para agregação de colunas. Por exemplo, Você pode querer linhas de dados onde a coluna Average Cost, seja maior que zero.. A linha Group Condition é equivalente a adicionar uma expressão com a cláusula HAVING em uma instrução SELECT do SQL agrupado (cláusula GROUP BY).

Definindo Expressões de Query

O Visual Query Builder permite definir expressões como parte de uma query. Expressões podem ser cálculos sobre valores de dados numéricos ou expressões de strings, tais como concatenação ou uma substring.
Para definir uma expressão, dê um clique no botão Expression na Toolbar. O quadro de diálogo Expression aparecerá

O quadro de diálogo Expression permite desenvolver expressões SQL. Você pode especificar operadores aritméticos, tais como:

- + (sinal “mais” ,de adição)
- - (sinal”menos” , de subtração)
- * (asterisco, para multiplicação)
- / (barra, para divisão)

Você pode incluir nomes de colunas e agregar expressões, tais como avg, count, min,max,e sum. Ou você pode editar expressões manualmente ou construir uma, utilizando o quadro de edição Expression.

Definindo Opções para Queries

O quadro de diálogo Options é utilizado para especificar opções para instruções SQL. Você pode especificar que registros duplicados sejam removidos . Esta opção tem o efeito de utilizar a instrução SQL DISTINCT.
Para especificar uma opção, dê um clique no botão Options na Toolbar. O quadro de diálogo Options aparece.

Exibindo Resultados de Query

Você pode executar a query que o Visual Query Builder gera. O resultado da query é exibido na janela Result Window. Esta janela permite verificar se as colunas da query, critério de seleção, agrupam ento e ordenação foram especificados corretamente pela query.
Para executar a query, dê um clique no botão Run da Toolbar. Uma Result Window aparecerá.

Exibindo a Intrução SQL para Query

Você pode checar instruções SQL para uma query visualizando a janela SQL Statement. A janela exibe a instrução SQL SELECT associada com a query atual. Conforme adicionar ou alterar colunas de query, critério ou ordenação, a SQL Window é atualizada automaticamente. Visualizar as intruções SQL fornece um feedback imediato sobre a contrução da query e o auxilia no aprendizado da sintaxe SQL.
Para checar as instruções, dê um clique no botão SQL na TollBar. A janela SQL statement aparecerá.

Definindo e Utilizando Campos Calculados

Introdução

Um campo calculado é um campo que exibe valores gerados pelo programa. Por exemplo, você pode ter uma tabela de banco de dados contendo informações sobre objetos retangulares. Cada linha possui uma coluna de altura e largura, mas você precisa exibir a área dos objetos retangulares. A solução é calcular a área através das alturas e larguras.
O Delphi permite utilizar campos calculados em suas aplicações.
Implementar um campo calculado envolve o seguinte processo:

Estágio	Processo
1	Definir o campo calculado
2	Escrever código para o campo calculado

Passos para o Estágio 1

Campos calculados são definidos com a assistência do Fields Editor.
O Fields Editor trabalha com os componentes Table e Query do TDataSet. Quando você define um campo calculado para um dataset, o Fields Editor adiciona um novo objeto campo ao dataset.
Execute os passos abaixo para definir um campo calculado

Passo	Ação
1	Selecione o componente Query ou Table (derivado TDataSet), e inicialize o Fields Editor. Para TTable, este passo assume que as propriedades DatabaseName e TableName estão definidas. Para TQuery, este passo assume que as propriedades DatabaseName e SQL estão definidas.
2	Adicione campos para exibição ou cálculo no quadro de diálogo Fields Editor.
3	Dê um clique no botão Define no Fields Editor. O quadro de diálogo Define aparecerá.
4	Digite um nome para o campo calculado na caixa de texto Field Name. Conforme digitar o nome, ele é exibido concatenado ao nome do componente no campo Component name. Você pode alterar este nome se desejar.
5	Selecione uma entrada da caixa de lista Field type.
6	Habilite Calculated.
7	Dê um clique em OK para completar a definição do campo calculado.

Passos para o Estágio 2

Uma vez definido o campo, você fornece valores para o campo escrevendo código para ele.
Execute os passos abaixo para implementar um campo calculado:

Passo	Ação
1	Selecione o componente DataSet (TQuery ou TTable) no Object Selector do Object Inspector.
2	Exiba a página Events do Object Inspector
3	Dê um duplo-clique sobre a coluna do event handler do evento OnCalcFields para criar e exibir a procedure OnCalcFields no Code Editor.
4	Digite o código que define o valor e propriedades do campo calculado.
5	Compile e grave sua aplicação. Execute e teste-a para certificar-se que o campo está sendo calculado corretamente.

Tutorial: Criando uma Aplicação Utilizando um Campo

Introdução:

A melhor maneira de se aprender a trabalhar com campos calculados é utilizando um exemplo. Nesta seção você exibirá informação sobre países do mundo. As tabelas deste exemplo, Country, contém dois campos chamados Area e Population, entre outros.

Passos para Exibir a População por Kilometro Quadrado

Assumindo que o campo Area esteja em Kilometros quadrados execute os passos a seguir para exibir a população por kilometro quadrado em um form:

Passo	Ação
1	Abra um novo projeto e grave-o. Quando solicitado, grave a unit como EX7BMAIN.PAS e o projeto como EXAMP7B.DPR.
2	Utilizando a página Data Access da Component Palette, adicione um componente Table e DataSource ao seu form.
3	Defina as propriedades para os componentes Table e DataSource como segue:

Nome do Componente	Propriedade	Valor
Table1	DatabaseName TableName Active	DBDEMOS COUNTRY.DB True
DataSource1	DataSet	Table1

Passo	Ação
4	Utilizando a página Data Controls da Component Palette, adicione um componente DBGrid e DBNavigator ao seu form.
5	Utilize a tabela a seguir para definir as propriedades do DBNavigator e DBGrid para que utilizem o componente DataSource.

Nome do Componente	Propriedade	Valor
DBGrid1	DataSource	DataSource1
DBNavigator	DataSource	DataSource1

Passo	Ação
6	Inicie o Fields Editor para o Componente Table1.
7	Adicione todos os campos da tabela ao dataset.
8	Defina um campo chamado PopArea e selecione FloaTField na list box Field type.
9	Certifique-se que o check box Calculated está selecionado, e clique em OK.
10	Saia do Fields Editor.
11	Utilize o Object Inspector para criar um event handler para o evento OnCalcFields do componente Table. Digite código no event handler para que fique similar ao seguinte:

Passo	Ação
12	Utilize o Object Inspector para definir as seguintes propriedades do componente Field:

Nome do Componente	Propriedade	Valor
Table1PopArea	DisplayFormat	. #

Passo	Ação
13	Execute e teste a aplicação
14	Quando tiver terminado o teste, grave e feche seu projeto.

Utilizando os Componentes Database Lookup

Introdução

Um database lookup, ou tabela de busca é o processo de encontrar texto descritivo de um valor codificado. Uma situação comum onde tabelas de busca seriam utilizadas, seria quando você está editando o input de usuário e quiser exibir uma informação mais significativa, e não valores codificados.
O Delphi fornece dois componentes para buscar valores em uma tabela de banco de dados, ou editar o input contra um valor em uma tabela. Eles são os seguintes:

- Componente DBLookupList
- Componente DBLookupCombo

O termo lookup em cada nome de componente refere-se às tabelas de busca. O termo lookup tables refere-se às tabelas que contém informação descritiva sobre um valor codificado. Esta seção discute a utilização destes componentes. Eles se encontram na página Data Controls da Component Palette.

Componente DBLookupList

O componente DBLookupList é um componente ListBox data-aware projetado para buscar valores em uma tabela baseado no valor de uma segunda tabela. O DBLookupList contém um conjunto finito de dados, o usuário deve selecionar uma das opções da lista. Um DBLookupList permite exibir um conjunto de opções baseado no valor em outra tabela. O componente DBLookupList difere do componente DBListBox porque permite coordenar o valor selecionado do DBLookupList com a linha corrente de outra tabela do banco de dados.

Componentes DBLookupCombo

O componente DBLookupCombo é um componente ComboBox datawarw similar ao BDLookupList, exceto que um usuário pode selecionar um valor na lista ou digitar um novo valor. Um ComboBox de onde o DBLookupCombo é derivado combina as capacidades de um ListBox com as capacidades de um componente Edit.

Adicionando um Componente Database Lookup a um Form

Quando você adiciona um componente DBLookupList ou DBLookupCombo ao seu form, assumimos que:

- Você possui uma aplicação de banco de dados
- O form na aplicação possui pelo menos um DataSource e um componente derivado de TDataSet sendo utilizado para exibir informações do banco de dados

Adicionar um componente database lookup envolve o seguinte processo:

Estágio	Processo
1	Adicionar o componente database lookup e ligá-lo a um componente DataSource existente e propriedade DataFeild
2	Adicionar um novo componente Query ou Table (TDataSet) e DataSource, e utilizar este DataSource para buscar valores codificados no primeiro DataSource

Passos para o Estágio 1

Execute os passos a seguir para adicionar um componente DBLookupList ou DBLookupCombo a um data source existente:

Passo	Ação
1	Adicione um componente database lookup em seu form.
2	Defina a propriedade DataSource a um componente DataSource que exista no form e que contenha o valor que você esteja procurando.
3	Defina a propriedade DataField ao campo que necessite de busca.

Passos para o Estágio 2

Execute os passos a seguir para anexar o componente DBLookupList ou DBLookupCombo à tabela de busca:

Passo	Ação
1	Adicione um componente TDataSet utilizando um componente Table ou Query que corresponda à tabela de busca.
2	Defina a propriedade DatabaseName do novo componente TDataSet.
3	Execute um dos seguintes: • Defina a propriedade TableName do novo componente Table. • Entre com um query para a propriedade SQL do componente query.
4	Adicione um novo componente DataSource, e defina a propriedade DataSet ao novo componente TDataSet.
5	Defina a propriedade LookupSource de componente DBLookup ao novo componente DataSource.
6	Defina a propriedade LookupField ao valor chave da tabela de busca.
7	Defina a propriedade LookupDisplay ao campo que você queira exibir no componente DBLookupList.

Tutorial: Criando uma Aplicação Utilizando um Componente Database Lookup

Introdução

A melhor maneira de se aprender a utilizar um componente DBLookup é através de um exemplo. Nesta seção, você utilizará um componente DBLookupList para exibir um nome de empresa baseado no campo CustlD encontrado na tabela Orders.

Passos do Tutorial

O processo do tutorial envolve os seguintes estágios:

Estágio	Processo
1	Criar um form que exiba uma lista estática dos campos de uma tabela de banco de dados utilizando um componente Table, DataSource, DBGrid e DBNavigator
2	Adicionar e conectar um componente TDBLookupList à aplicação
3	Adicionar código para coordenar ações do componente TDBLookupList quando o sistema insere ou atualiza uma linha

Passos para o Estágio 1

Execute os passos a seguir para criar um form que exiba campos selecionados da tabela Orders:

Passo	Ação
1	Abra um novo projeto e grave-o Quando solicitado, grave a unit como EX7CMAIN.PAS e o projeto como EXAMP7.DPR.

2	Utilizando a página Data Access da Component Palette, adicione um componente Table e DataSource ao seu form.
3	Utilizando a página Data Controls, adicione os seguintes ao seu form: • Um componente DBGrid • Um componente DBNavigator
4	Defina as seguintes propriedades para cada componente, como mostrado na tabela:

Nome do Componente	Propriedade	Valor
Table1	Database Name TableName Active	DBEMOS ORDERS.DB True
DataSource1	DataSet AutoEdit	Table1 False
DBGrid1	DataSource	DataSource1
DBNavigator	DataSource	DataSource1

Passo	Ação
5	Utilize o Fields Editor para adicionar os seguintes campos ao dataset Table1: • OrderNo • CustNo • SaleDate • ItemsTotal • AmountPaid
6	Arranje os campos no Fields Editor para que OrderNo seja o primeiro campo e CustNo seja o segundo.
7	Compile e grave a aplicação.Execute e teste-a.

Passos para o Estágio 2

Execute os passos a seguir para adicionar um componente DBLookupList à aplicação:

Passo	Ação
1	Utilizando a página Data Controls da Component Palette, adicione um componente DBLookupList ao seu form:
2	Utilize o Object Inspector para definir as seguintes propriedade do componente DBLookupList:

Nome do Componente	Propriedade	Valor
DBLookupList1	DataSource DataField	DataSource1 CustNo

Passo	Ação
3	Adicione um novo componente Table e DataSource ao seu form. Defina as propriedades para cada componente, como mostrado na tabela a seguir:

Nome do Componente	Propriedade	Valor
Table2	DatabaseName TableName Active	DBDEMOS CUSTOMER.DB True
DataSource2	DataSet	Table2

Passo	Ação
4	Utilize o Fields Editor para adicionar os seguintes campos ao dataset Table2: • CustNo • Company
5	Conecte o componente DBLookupList ao segundo DataSource utilizando a tabela a seguir:

Nome do Componente	Propriedade	Valor
DBLookupList1	LookupSource LookupField LookupDisplay	DataSource2 CustNo Company

Passo	Ação
6	Compile e grave a aplicação. Execute e teste-a. O componente DBLookupList destaca a empresa que corresponde ao valor de CustNo na linha do DBGrid.

Passos para o Estágio 3

Este código de event handler permitirá definir o valor de CustNo durante o modo de edição ou inserção através de um duplo-clique no componente DBLookupList.
Execute os passos a seguir para adicionar um event handler para os eventos de DBLookupList, OnDblClick, ou OnClick:

Passo	Ação
1	Adicione as instruções a seguir no evento OnDblClick do componente DBLookupList: *if Table.State in [dsEdit, dsInsert] then* *Table1CustNo.Value := Table2CustNo.Value;*
2	Compile e grave sua aplicação. Execute e teste-a alternando a aplicação entre o modo de edição e inserção e utilizando o DBLookupList para definir o valor de CustNo na tabela Orders.

Lab: Utilizando Componentes de Banco de Dados

Objetivos

- Este lab reforça a sua habilidade em:
- Adicionar um componente Table
- Adicionar um componente Database Grid
- Adicionar um componente Database Query
- Adicionar componentes Data Access
- Utilizar o Visual Query Builder
- Manipular e coordenar as ações dos componentes Data Access e Data Control

Cenário

Você construirá o início de um sistema de acompanhamento de vendas. Este sistema utilizará uma única grade de banco de dados para executar duas tarefas:

- Exibir informações de clientes e faturas
- Ligar clientes a faturas

No processo de construção desta aplicação, você utilizará e coordenará as ações dos componentes Data Access, Data Control e Standard.

Processo

Utilize o seguinte processo para aplicar o que você aprendeu:

Estágio	Processo
1	Abra um novo projeto e grave-o. Quando solicitado, grave a unit como LAB7MAIN.PAS e o projeto como LAB7.DPR. Altere o Caption do form para que exiba Aplicação de Acompanhamento de Vendas.
2	Adicione os seguintes componentes ao form: • DataSource • Table • DBGrid • Button Defina as propriedades a seguir aos componentes, como mostrado na tabela:

Nome do Componente	Propriedade	Valor
Table1	DatabaseName TableName Active	DBDEMOS CUSTOMER.DB False
DataSource1	DataSet	Table1
DBGrid1	DataSource	DataSource1
Button1	Caption	Abrir Tabela &Customer

Estágio	Processo
3	Adicione event handler OnClick para Button1 para que funcione como um botão de Liga?Desliga da Tabela Customer. Compile e grave sua aplicação. Execute e teste-a.

```
Procedure TForm1.Button1Click(Sender: Tobject);
begin
    if Table1.Active = True then
    begin
       Table1.Close;
       Button1.Caption :+ Ábrir Tabela  &Customer ';
     end
     else
        begin
            Table1.Close;
             Button1.Caption : = 'Fechar Tabela &Customer';
         end
end;
```

Estágio	Processo
4	Adicione os seguintes componentes ao form: • Query • Data Source • Button Defina as seguintes propriedades aos componentes:

Nome do Componente	Propriedade	Valor
Query1	Database Name	DBDEMOS

	SQL Active	select*from orders False
DataSource2	DataSet	Query1
Button2	Caption	Abrir Tabela &Order

Estágio	Processo
5	Adicione e event handler OnClick a seguir para Button2. Compile e grave sua aplicação. Execute e teste-a. Você consegue abrir a tabela Order?

```
procedure TForm1.Button2Click (Sender : Tobject);
begin
    if Query1.Active = True then
    begin
        Query1.Active := False;
         Button2.Caption :+ Ábrir Tabela &Order';
    end
    else
    begin
        Query1.Active :+ True;
        Button 2.Caption :+ 'Fechar Tabela &Order';
    end
end;
```

Estágio	Processo
6	Adicione dois componentes RadioButton ao form. Defina as seguintes propriedades aos componentes:

Nome do Componente	Propriedade	Valor
RadioButton1	Caption Checked	Ver Clientes True
RadioButton2	Caption	Ver Pedidos

Passo	Ácão
7	Adicione o event handler OnClick a seguir para RadioButton1 e RadioButton2 respectivamente: *DBGrid1.DataSource :+ DataSource1;* *DBGrid1.DataSource :+ DataSource2;*
8	Compile e teste sua aplicação como segue: • Dê um clique nos botões para abrir ambas as tabelas. • Utilize os botões de rádio para alternar entre os conjuntos de dados. • Tudo está funcionando?
9	Adicione os seguintes componentes complementares: • Group • Button • Dois componentes Edit • Dois componentes Label Defina as propriedades a seguir para cada um dos componentes e utilize o código a seguir para event handler OnClick de Button3:

Nome do Componente	Propriedade	Valor
GroupBox1	Caption	(Vazio)
Button3	Caption	&Definir Clientes
Edit1	Text	(Vazio)
Edit2	Text	(Vazio)

Label1	Caption	Início:
Label2	Caption	Final:

```
procedure TForm1.Button3Click(Sender : Tobject);
begin
    Table1.SetRangeStart;
    Table1.Fields[0] .AsString :+ Edit1.Text;
    Table1.SetRangeEnd;
    Table1.Fields[0] .AsString :+ Edit2.Text;
    Table1.ApplyRange;
end;
```

Estágio	Processo
10	Compile e grave sua aplicação. Execute e teste-a, como segue: • Dê um clique em Abrir Tabela Orders e depois em Ver Pedidos. • Digite valores nos campos Início e Final (por exemplo, 1005 e 1009), e dê um clique em Definir Clientes. • Qual o maior número de pedido exibido?----- • Qual o menor?----
11	Adicione os seguintes componentes complementares ao seu form: • Query • DataSource • RadioButton Defina as propriedades a seguir para cada componente, como mostrado na tabela.

Nome do Componente	Propriedade	Valor
Query2	DatabaseName	DBDEMOS
DataSource3	DataSet	Query2
RadioButton3	Caption	Ver Pedidos de Clientes

Estágio	Processo
12	Execute as tarefas a seguir para dar à sua aplicação a habilidade de utilizar o componente DBGrid para exibir todos os seus pedidos agrupados por cliente. • Inicialize o Query Builder utilizando o SpeedMenu do Componente Query2. Selecione DBDEMOS na lista. • Adicione as tabelas Customer e Orders. • Crie uma ligação do campo CustNo na tabela Customer com o campo CustNo na tabela Orders. • Adicione os seguintes campos ao resultado da query: Customer.CustNo Customer.Company Orders.OrderNo Orders.AmoundPaid
13	Defina o critério de ordenação. Ordene o resultado da query na ordem ascendente por Customer Number. Grave seu trabalho e saia do Visual Query Builder.
14	Utilizando o Object Inspector, localize o componente Query2. Visualize a propriedade SQL e defina a propriedade Active para True.
15	Adicione o seguinte event handler OnClick para RadioButton3: procedure TForm1.RadioButton3Click(Sender : Tobject); begin DBGrid1.DataSource :+ DataSource3; end;

16	Compile e grave sua aplicação. Execute e teste-a, como segue: • Dê um clique em Ver Pedidos de Clientes. • Dê um clique em Ver Pedidos de Clientes. Sua query foi executada corretamente?

Resumo do Capítulo

Pontos Chave

Após completar este capítulo, você aprendeu que:

- Para manipular e consultar bancos de dados no Delphi, você deve entender o conceito de um dataset. Um dataset no Delphi é o objeto tipo TDataSet e é uma classe abstrata. Os componentes Query, Table, e StoredProc são chamados de componentes TDataSet porque são derivados do objeto TDataSet.
- O componente DataSource atua como um intermediário entre o componente DataSet (TTable ou TQuery) e o componente Data Control.
- O componente Table é um componente DataSet que se comunica com uma tabela de banco de dados através do BDE.
- O Fields Editor permite criar uma lista de campos de um dataset.
- O componente DBGrid fornece uma maneira conveniente de exibir diversas linhas de dados de um componente Table ou Query. Sua aplicação pode utilizar o DBGrid para inserir, deletar, editar ou exibir dados de um banco de dados.
- O componente Query permite utilizar SQL, gerenciar a comunicação com o BDE, e serve como interface entre o BDE e os componentes DataSource em seus forms.
- O visual Query Builder é uma ferramenta visual para construir queries baseadas em SQL, e permite construir queries complexas com pouco ou nenhum conhecimento de SQL.
- O Fields Editor trabalha com os componentes Table e Query do TDataSet para definir campos calculados. O Fields Editor adiciona um novo objeto campo ao dataset.
- O Delphi fornece os componentes DBLookupList e DBLookupCombo para valores de busca em uma tabela.

Termos e Definições.

A tabela a seguir é uma referência rápida aos termos explicados neste capítulo.

Termo	Definição
Campo calculado	Um campo que exibe valores gerados pelo programa
Classe abstrata	Uma classe de onde você pode derivar outras classes. Entretanto, você não pode criar uma variável desta classe.
Componente DataSet	Um componente Table, Query, ou StoredProc, que é derivado de um objeto the TDataSet
Database lookup	O processo de encontrar o texto descritivo para um valor codificado
Dataset	Um objeto no Delphi que consiste de uma série de registros, cada um contendo qualquer número de campos e um ponteiro para o registro atual
Query parametrizada	Uma query onde um ou mais valores na condição de seleção são desconhecidos até o momento da execução
Tabelas de busca	Tabelas que contém informação descritiva sobre um valor codificado