Curso de criação de componentes em Delphi

Unidade 2. Um pouco de Teoria

Voltar ao índice

Por Luis Roche

O que são componentes? A biblioteca visual de componentes (VCL).

Os componentes são a pedra angular da programação em Delphi. Embora a maioria dos componentes representam partes visíveis da interface de usuário, também existem os componentes não-visuais, como por o exemplo o objeto Cronômetro e Banco de dados.

Um componente, em sua definição mais simples mais é um objeto descendente tipo TComponent. Todos os componentes descendem na forma mais primitiva do TComponent, desde que TComponent provê as características básicas que todo o componente deveria ter: capacidade de ser mostrado na paleta de componentes como também de operar em "Design-time" .

Os componentes fazem a programação em Delphi ficar mais facil. Em vez de ter que operar a nível de unidades, o usuário de um componente tem que preencher as propriedades dele e localizar isto na posição querida da forma dele simplesmente. Isso é tudo:O Delphi se encarrega do resto.

Todos os componentes são parte da hierarquia de Biblioteca de Componente de objetos denominada Visual (VCL). Quando um componente novo é criado, este é derivado a partir de um componente existente (bem como o TComponent ou algum outro especializado) e é somado ao VCL.

Anatomia de um componente. Propriedades, métodos e eventos.

- Como já foi mencionado, um componente é um objeto, e como tal, consiste em código e dados. Mas quando se referindo a estes nós não usaremos estas condições, mas nós falaremos bastante de propriedades , métodos e eventos. Ao longo deste curso iremos estudar todos estes aspectos a fundo, mas vamos fazer um pequeno resumo:

Propriedades

> As propriedades provêem ao usuário do componente um acesso fácil para o mesmo. Ao mesmo tempo, permite para o programador do componente esconder a estrutura subjacente de dados.

Entre as vantagens de usar propriedades para consentir ao componente pode fazer um compromisso:

- As propriedades estão você pode alterar em design-time. Deste modo o usuário do componente pode inicializar os valores das propriedades sem necessidade de escritura uma linha de código.
- As propriedades permitem a validação dos dados para o mesmo tempo de ser introduzidos. Podem ser prevenidos erros deste modo causado por valores inválidos.
- Eles nos asseguram que o primeiro valor que colocarmos nas propriedades terão que ser certas e evitarão o erro.

- Eventos

  Os eventos são as conexões existentes entre um certo evento e o código escrito pelo programador de componentes. Deste modo por exemplo, antes do clique de evento do mouse poderia se mostrar uma mensagem em tela. Para o código que é executado quando um certo evento o leva a um lugar é denominado Gerenciador de Eventos (o Gerenciador de Eventos) é normalmente escrito pelo usuario do componente. Os eventos mais comuns já são parte dos próprios componentes de Delphi (eventos do mouse, teclado...), mas também é possível definir eventos novos.

- Métodos

  Os métodos são as funções de E/S de procedimentos que eles são parte do componente. O usuário do componente os usa para obter uma certa ação ou obter um certo valor para o qual você não pode conseguir por meio de uma propriedade. Considerando que eles requerem execução de código, os métodos estão disponíveis só em run-time.

Controle de acesso para um componente. Declarações: Private, Protected, Public e Pubished.

O Object Pascal tem quatro níveis de controle de acesso para os campos, propriedades e métodos de um componente. Este controle de acesso permite o programador de componentes em que partes do código especificar as declarações do objeto. Deste modo é definido a interface do componente. É importante planejar bem esta interface, deste modo nossos componentes serão facilmente programáveis e reutilizaveis.

A menos que o oposto seja especificado, os campos, propriedades e métodos que são somados a um objeto são de tipo published. Todos os níveis de controle de acesso operam a nível de unidades, quer dizer, se uma parte de um objeto é acessível (ou inacessível) em uma parte de uma unidade, é também acessível (ou inacessível) em qualquer outra parte da unidade.
Logo os tipos de controles de acesso são detalhados:

Private: escondendo os detalhes de implementação.

Declarando uma parte de um componente (bem é um campo, propriedade ou método) private (você **se priva**). Aquela parte do objeto é invisível ao código externo para a unidade no o qual o objeto é declarado. Dentro da unidade que contém a declaração, o código como o que pode consentir àquela parte do objeto se fosse público.

A utilidade principal das declarações privadas é que eles permitem esconder os detalhes de

implementação do componente para o usuário final do mesmo, desde estes eles não podem consentir à parte privada de um objeto. Deste modo você pode mudar o implementação interno do objeto sem afetar ao código que o usuário escreveu.

- *Protected: definindo a interface do programador.*

Declarar uma parte de um componente como sendo protegido é o mesmo que declarar isto em privado. (é escondido ao código externo à unidade). A diferença principal entre declarar uma parte de um objeto protegido ou fazer isto privado são que os descendentes do componente poderão fazer referência àquela parte.

Este parte é especialmente útil para o criação de componentes.

- *Public: definindo a interface a tempo de execução.*

As partes inteiras de um objeto que nós declaramos público, eles poderão ser indexado por qualquer código interno ou externo para a própria unidade. Deste modo, a parte pública identifica a interface a tempo de execução de nosso componente. Os métodos que o usuário do componente deveria chamar os públicos deveria ser declarado, como também as propriedades de read-only, para ser só válido a tempo de execução.

O público declara as propriedades que não aparecerão no Object Inspector.

Esta seção é talvez mais importante considerar quando estamos projetando um componente. Quando são projetados componentes, deveriam ser considerados cuidadosamente que métodos e propriedades deveriam ser públicas. Se o código está correto, isto permitirá a alteração as estruturas de dados e métodos internos do componente sem ter que jogar o público da interface que continuará sendo o mesmo para o usuário do componente.

- *Published: definindo a interface a design-time:*

Quando declarando parte de um objeto publicado é o mesmo que a pública e também gera a informação em run-time.

As propriedades declaradas publicadas aparecem no Object Inspector. Estas partes definem a interface a design-time de nosso componente.

Passos necessários para criar um componente. O expert em componentes.

As grandes características, os passos necessários para criar um componente novo são o seguinte:

1. Criar uma unidade para o componente novo.
2. Derivar o componente novo a partir de outro existente, o o qual servirá como base para somar as características do novo componente.
3. Somar as propriedades, eventos e métodos necessários para o componente novo.
4. Registrar o componente, inclusive o bitmaps adicional, arquivos de ajuda, etc.

5. Instalar o componente novo na paleta de componentes.

De todos os passos mencionados, há um que especialmente é que você tem que se preocupar: a escolha do componente ao qual vai se derivar. Este passo é crucial, com uma boa escolha do ascendente nós podemos economizar muito código e chegar facilmente ao nosso objetivo.
Como base para a escolha do objeto ascendente, ele serve para fazer para advertência as normas seguintes:

- TComponent - O ponto de partida para componentes não visuais.
- TWinControl - O ponto de partida se é necessário que o componente tenha "handles".
- TGraphicControl - Um ponto de partida bom para componentes visuais que não tenham "handles". Esta classe tem os metodos Paint e Canvas.
- TCustomControl - O ponto de partida mais comum. Esta classe tem "handle" de janela, eventos e propriedades comuns e, principalmente, os metodos Canvas e Paint.

Bem, nós já sabemos como determinar o ponto de partida. Deixe nos ver agora como criar a unidade que abrigará o componente. Há duas opções, criar a unidade manualmente ou deixar que o Delphi faça o trabalho sujo. (Use o Expert de componentes). Se nós escolhermos para a primeira opção, teriamos que fazer tudo na mão, mas é aconselhavel utilizar o expert do Delphi.

Abrir o expert de componentes no menu Component e New Component...
Aparecerá uma caixa de diálogo na qual nós devemos preencher os campos:

- *Class Name: Aqui nós deveríamos especificar o nome do componente novo.*
- *Ancestor Type: Nós introduziremos o ascendente aqui a partir do o qual nós derivaremos nosso componente.*
- *Pallete page: Nós indicaremos a página da paleta aqui no o qual nós queremos que o componente novo apareça.*

Uma vez introduzido estes campos, quando pressionamos OK você verá o código de nossa unidade. Se por exemplo nós introduzimos os dados seguintes no expert de componentes:
Class Name: TMiComponente
Ancestor Type: TComponent
Pallete Page: Curso

Quando clicamos aceitando,O Delphi nos geraria a unidade seguinte, listada abaixo para introduzir as propriedades e métodos de nosso componente:

```
unit Unit1;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs;

type
  TMiComponente = class(TComponent)
  private
    { Private declarations }
  protected
    { Protected declarations }
```

```
  public
    { Public declarations }
  published
    { Published declarations }
  end;

procedure Register;

implementation

procedure Register;
begin
  RegisterComponents('Curso', [TMiComponente]);
end;

end.
```

A partir daqui tudo o que consiste em introduzir as propriedades e métodos necessários para a operação de nosso componente. Mas antes de qualquer coisa deve se fazer algumas advrtências em alguns aspectos:

Na cláusula uses, o Delphi soma por padrão as unidades standards. Se nosso componente não usa alguns deles que nós podemos eliminar isto desta cláusula. Da mesma maneira, se nós usamos algum procedimento ou função localizada em outra unidade, nós deveríamos adicionar esta unidade para a cláusula que uses.

As declarações das propriedades, campos e métodos que nós vamos definir, os localizará na seção apropriada da interface como corresponde, quer dizer nós os declararemos private, protected, public ou published.

O Delphi declara e define o registro de procedimento automaticamente para registrar o componente na paleta.

Luis Roche revueltaroche@redestb.es

Ultima modificación 7.12.1996