



# Embarcadero Conference

# ENCAPSULAMENTO DE ROTINAS ALTAMENTE REAPROVEITAVEIS

Rodrigo Mourão

Embarcadero

Conference



# Agenda

Uma das grandes dificuldades que programadores e empresas de software tem hoje em dia é de acompanhar a evolução da tecnologia e do mercado.

Como estar preparado para o futuro. Como garantir que nossas rotinas atuais estejam fechadas a mudança porém abertas para expansão?

# Quem aqui lembra dos 4 pilares?

~~Herança, Polimorfismo, Encapsulamento e Abstração~~

Falta de Tempo

Falta de Dinheiro

Falta de Mão de Obra

Excesso de Bugs



# Encapsulamento

Todo objeto dever ser responsável por seus próprios comportamentos, com isto, precisa garantir a integridade daquilo que o faz funcionar de maneira regular.

Encapsulamento é uma técnica OO para ocultar determinados membros de um objeto.



# Encapsulamento

Conhecido como “Data Hidding” o encapsulamento é utilizado como mecanismo regulador de acesso aos dados de um objetos.

O encapsulamento torna as mudanças no comportamento interno de um objeto transparentes para outros objetos e auxilia na prevenção de problemas de efeitos colaterais no código.

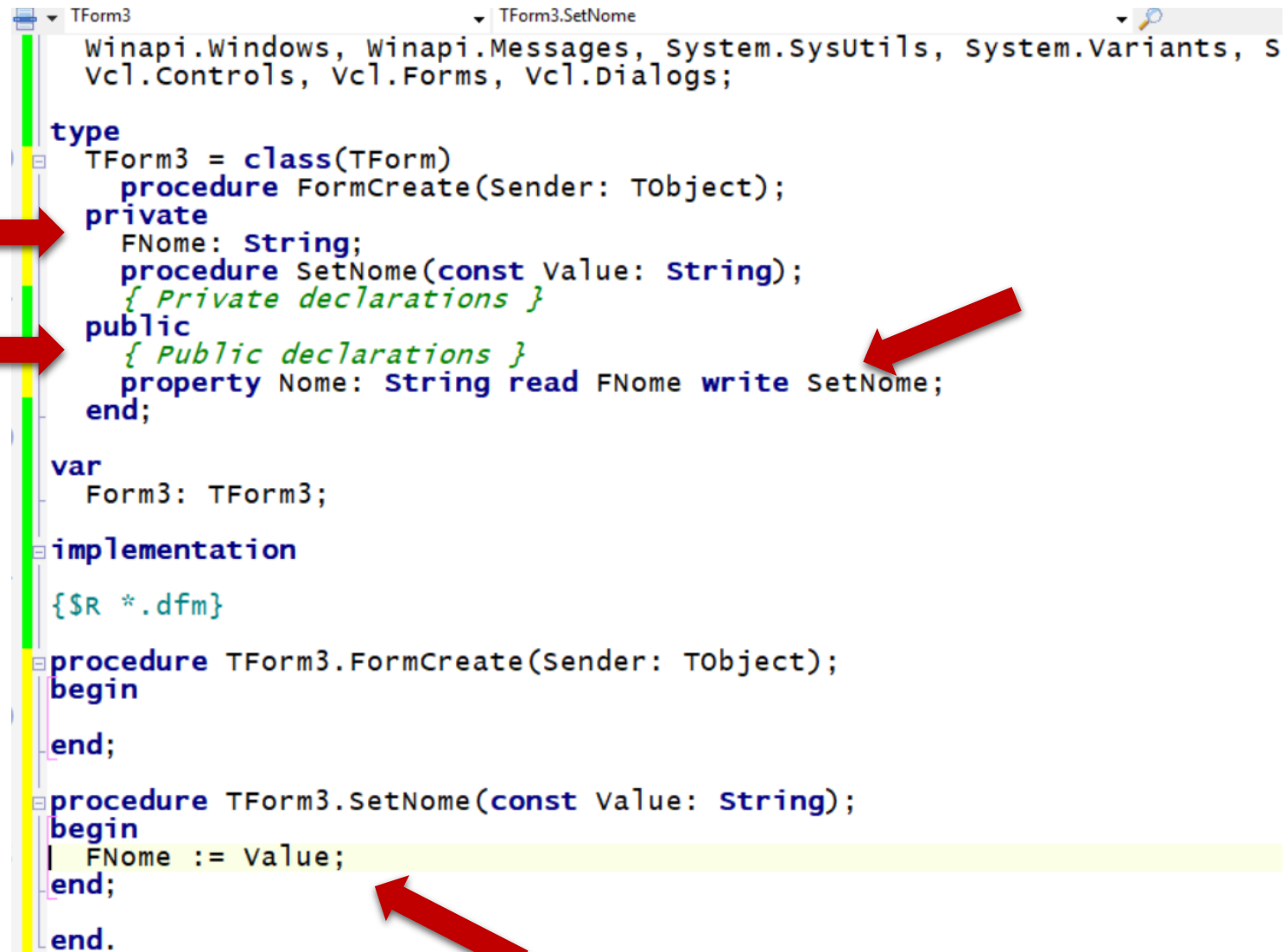


# Esqueça tudo e lembra apenas de uma coisa

Pegue o que varia no seu código e encapsule!



# Esqueça tudo e lembra apenas de uma coisa



```
unit TForm3;
interface
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Controls, Vcl.Forms, Vcl.Dialogs;

type
  TForm3 = class(TForm)
    procedure FormCreate(Sender: TObject);
  private
    FName: String;
    procedure SetNome(const Value: String);
  public
    { Public declarations }
    property Nome: String read FName write SetNome;
  end;

var
  Form3: TForm3;

implementation
  {$R *.dfm}

  procedure TForm3.FormCreate(Sender: TObject);
  begin
  end;

  procedure TForm3.SetNome(const Value: String);
  begin
    FName := Value;
  end;

end.
```

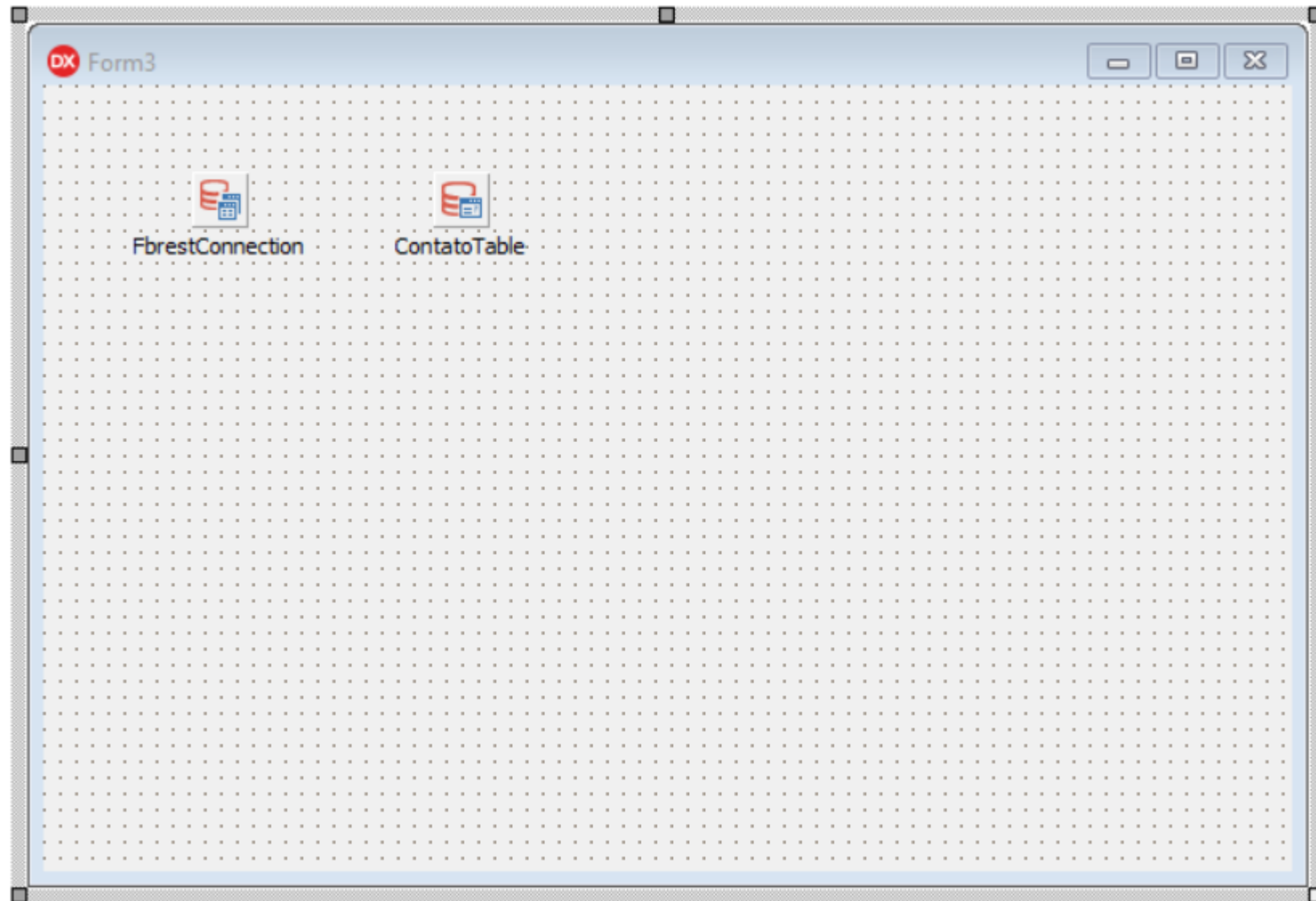


Existe 1000  
maneiras de  
preparar Neston

Embarcadero

Conference

# Simples, rápido mas ordinário



# Tá vai! Valeu a Intenção

```
FNome: String;
procedure SetNome(const Value: String);
{ Private declarations }
public
{ Public declarations }
property Nome: String read FNome write SetNome;
end;

var
    Form3: TForm3;

implementation

{$R *.dfm}

procedure TForm3.FormCreate(Sender: TObject);
var
    Qry: TFDQuery;
begin
    Qry := TFDQuery.Create(Self);
    Qry.SQL.Add('Select * from contatos');
    Qry.Open();
end;

procedure TForm3.SetNome(const Value: String);
begin
    FNome := Value;
end;

end.
```

# Coisa linda de se ver!

```
TDBConnection = class
strict private
  class var FInstance: TDBConnection;
private
  FConnection: TDBXConnection;
  FCommand: TDBXCommand;
  FTransaction: TDBXTransaction;
  FInTransaction: Boolean;
  constructor CreatePrivate;
  function GetSqlCommand: String;
  function GetTransaction: Boolean;
  procedure SetSqlCommand(const Value: String);
  function GerConnection: TDBXConnection;
  function DoTrace(TraceInfo: TDBXTraceInfo): CBRTYPE;
public
  constructor Create;
  class function GetInstance: TDBConnection;
  destructor Destroy; override;
  property SqlCommand: String read GetSqlCommand write SetSqlCommand;
  procedure ExecuteCommand;
  function ExecuteReader: TDBXReader;
  procedure StarTransaction;
  procedure Commit;
  procedure Rollback;
  property InTransaction: Boolean read GetTransaction;
  property Connection: TDBXConnection read GerConnection;
end;
```



Dúvidas?

Embarcadero

Conference

# OBRIGADO



[rodrigomourao@rodrigomourao.com.br](mailto:rodrigomourao@rodrigomourao.com.br)

<http://blog.rmfactory.com.br>

@rm.mourao

(21) 98288-7488

Embarcadero

Conference



