

Embarcadero Conference 2024

Inovação faz parte do nosso DNA!

Aumentando a produtividade com RTTI

{Wesley Capelari





Fiorilli S/C Software

Atuando há mais de 48 anos

Uma das maiores empresas do Brasil no desenvolvimento de softwares de gestão pública

Presentes em mais de 1000 municípios facilitando a rotina de mais de 2500 órgãos públicos

Líder de mercado, é a empresa que mais cresce em seu segmento

O que é RTTI?

- RTTI (Runtime Type Information)
- Biblioteca de reflexão/introspecção
- Consegue em tempo de execução obter informações sobre a estrutura dos objetos instanciados ou classes
- Consegue em tempo de execução instanciar um objeto, destruí-lo ou modificar suas propriedades



O que preciso para usar o RTTI?

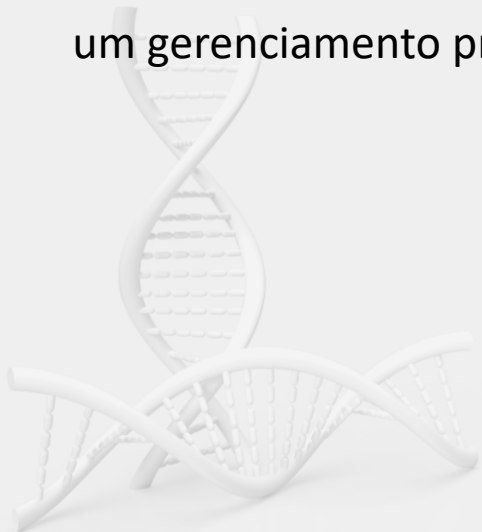
- Importar a biblioteca **System.RTTI** na unit
- Criar um “contexto” (TRTTIContext)
- Buscar o “tipo” (TRTTIType) da instância ou classe desejada



Estrutura simples do uso do RTTI

Aqui vemos um exemplo básico de como utilizar o RTTI dentro do nosso código.

Obs: não é necessário a destruição das instâncias criadas com as classes do RTTI, pois eles possuem um gerenciamento próprio de memória.



```
.
- uses
.   System.Rtti;
.
.   { TFrmPrincipal }
.
30 procedure TFrmPrincipal.MinhaProcedure;
.   var
.       LContext: TRttiContext;
.       LType    : TRttiType;
.   begin
.       LContext := TRttiContext.Create;
.       LType    := LContext.GetType(TFrmPrincipal);
.   end;
.
.   end.
40
```

```
40
.   end;
.   end;
```

Como “conhecer” a instância ou classe através do RTTI

Primeiramente devemos instanciar nosso contexto e tipo, com o tipo podemos pegar seus:

- Campos (Fields)
- Métodos (Methods)
- Propriedades (Properties)
- Atributos Personalizados (Custom Attributes)

Sejam eles publicados, privados, protegidos ou públicos*

*Desde que habilitado a diretiva em sua classe

```
type
0  ▢ MeuAtributo = class(TCustomAttribute)
    .
    .
    .
    [MeuAtributo]
    ▢ TFrmPrincipal = class(TForm)
        published
        { Published declarations }
        MeuCampo: TStringList;
        private
        { Private declarations }
0     [MeuAtributo]
        FMeuCampo: string;
        protected
        { Protected declarations }
        procedure SetMeuCampo(const AValue: string);
        public
        { Public declarations }
        procedure MinhaProcedura;

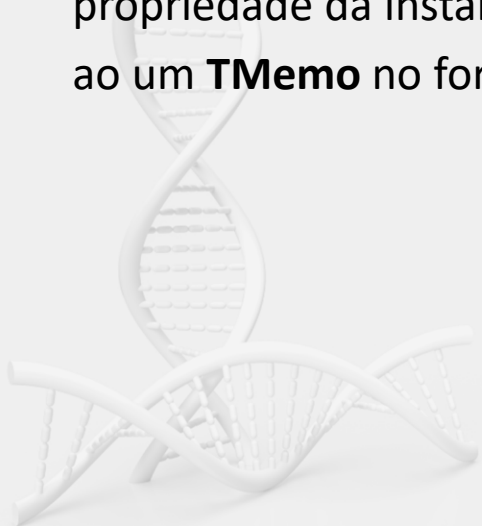
        [MeuAtributo]
0     property MinhaPropriedade: string read FMeuCampo write SetMeuCampo;
    .
    .
    .
end;
```

```
0  sup?
    property MinhaPropriedade: string read FMeuCampo write SetMeuCampo;
    [MeuAtributo]

    procedure MinhaProcedura?
    .
    .
    .
```


Como buscar o valor de uma propriedade da instância

Esta função ao lado está presente no Exemplo 02 do repositório, é um exemplo de código que pega todas as propriedades da classe **TFrmPrincipal** através do método **GetValue**, e caso ela seja do tipo string, o RTTI pega o valor contido na propriedade da instância **FrmPrincipal** e adiciona ao um **TMemo** no formulário



```
procedure TFrmPrincipal.btnGetPropertiesClick(Sender: TObject);
var
  LContext : TRttiContext;
  LType     : TRttiType;
  LProperty: TRttiProperty;
  LValue    : string;
begin
  LContext := TRttiContext.Create;
  LType     := LContext.GetType(TFrmPrincipal);

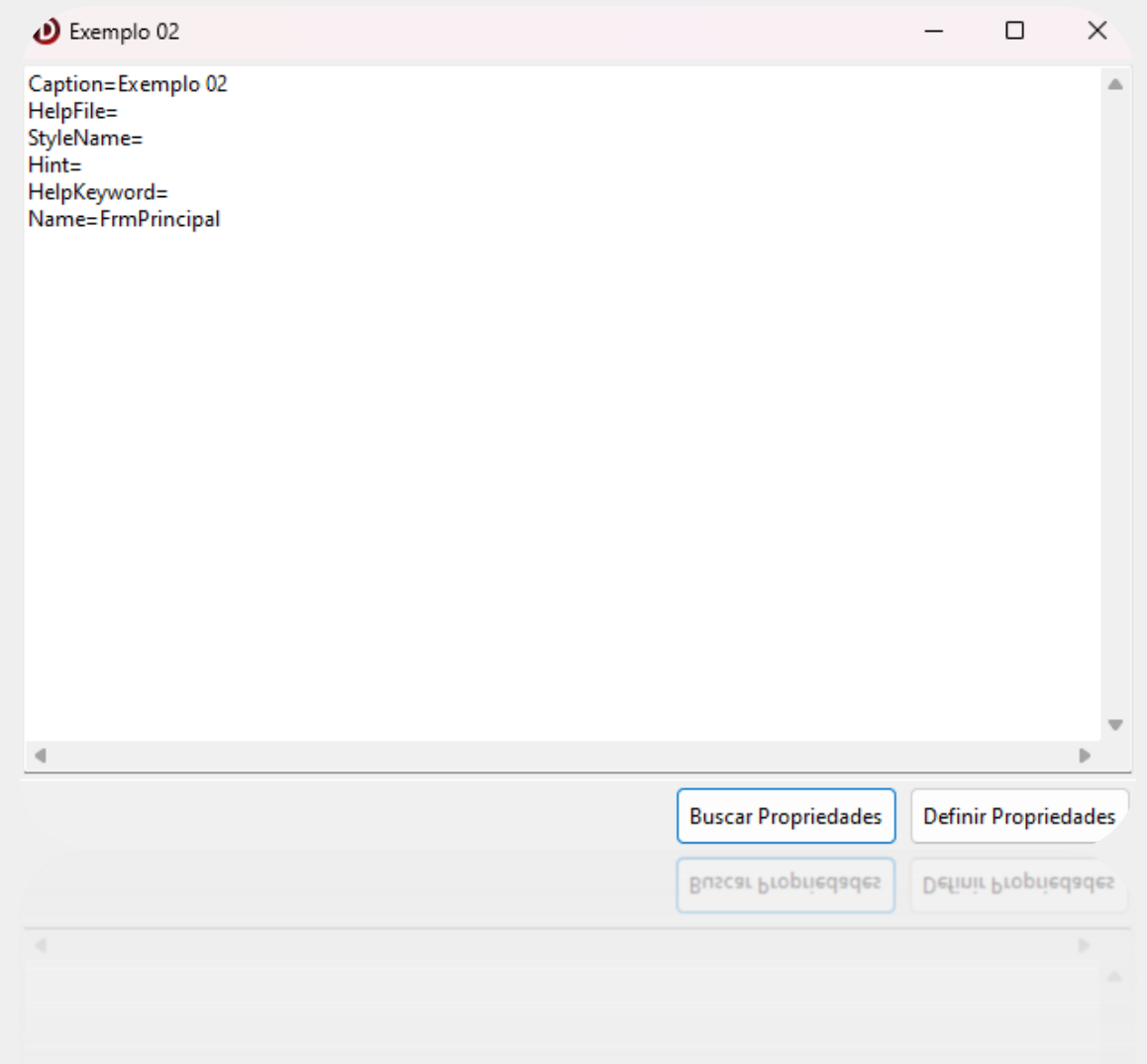
  mmoProperties.Lines.Clear;
  for LProperty in LType.GetProperties do
  begin
    if LProperty.DataType.TypeKind in C_VALID_TYPES then
    begin
      LValue := LProperty.GetValue(FrmPrincipal).AsString;

      if not mmoProperties.Lines.ContainsName(LProperty.Name) then
        mmoProperties.Lines.AddPair(LProperty.Name, '');

      if not LValue.IsEmpty then
        mmoProperties.Lines.Values[LProperty.Name] := LValue;
      end;
    end;
  end;
end;
```


Como buscar o valor de uma propriedade da instância

Aqui podemos ver o valor retornado pela função no **TMemo** do **Exemplo 02**



Como definir o valor de uma propriedade da instância

O mesmo podemos fazer para definir o valor, mas desta vez pegando as propriedades de **TFrmPrincipal**, e caso encontre um valor para ela no **TMemo** do formulário, ele o define na instância **FrmPrincipal**

Também se encontra no **Exemplo 02**

Obs: No caso o valor que será definido deve ser passado através de um **TValue**, que é a classe do RTTI responsável por representar os valores de uma forma generalizada.

```
procedure TFrmPrincipal.btnSetPropertiesClick(Sender: TObject);
var
  LContext : TRttiContext;
  LType    : TRttiType;
  LProperty: TRttiProperty;
  LValue    : TValue;
begin
  LContext := TRttiContext.Create;
  LType    := LContext.GetType(TFrmPrincipal);

  for LProperty in LType.GetProperties do
  begin
    if LProperty.DataType.TypeKind in C_VALID_TYPES then
    begin
      if not mmoProperties.Lines.ContainsName(LProperty.Name) then
        Continue;↑

      LValue := TValue.From<string>(mmoProperties.Lines.Values[LProperty.Name]);

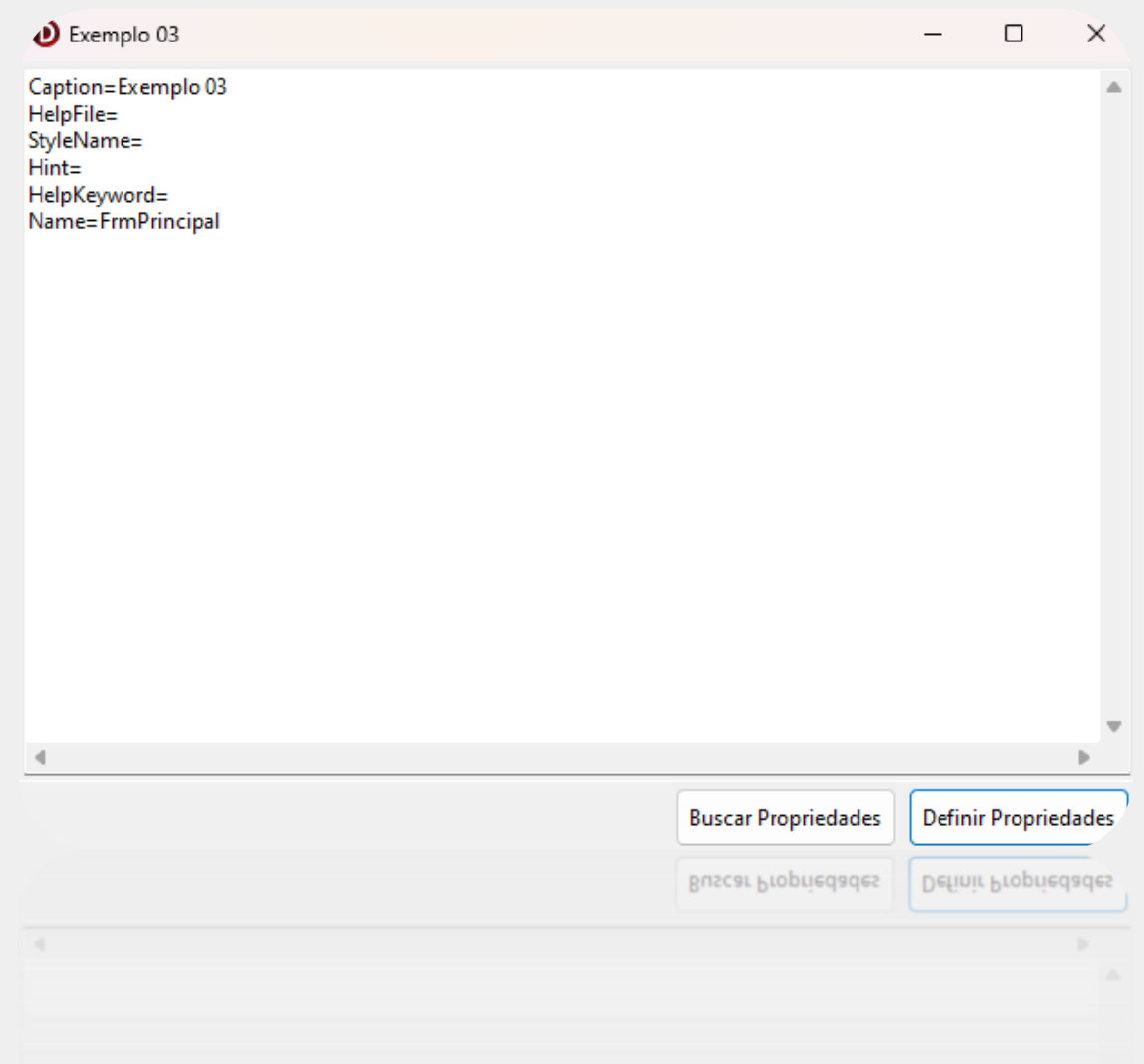
      if LValue.IsEmpty then
        Continue;↑

      LProperty.SetValue(FrmPrincipal, LValue);
    end;
  end;
end;
```

```
cuq?
cuq?
cuq?
[Global::GetLValue(FrmPrincipal, LValue)]
Continue;↑
```

Como definir o valor de uma propriedade da instância

Aqui temos o exemplo de uso onde alteramos o **Caption** do formulário para **Exemplo 03** em tempo de execução através do RTTI



Como aumentar a produtividade com RTTI?

- Software ou biblioteca com objetivo bem claro
- Generalização das funções
- Dependendo das funcionalidades criar uma classe base para conter as funções

Como aumentar a produtividade com RTTI?

Para exemplificar como podemos aumentar a produtividade do desenvolvimento, vamos desenvolver um software que contem primeiramente uma classe base que será herdada por todas as outras classes do projeto.

E para demonstração vamos criar uma classe **TLivro** com as propriedades Título, Autor, Data de Publicação e Ano de Aquisição, que herda a classe **TBaseClass** que tem a propriedade Nome.

```
4 TBaseClass = class(TObject)
    private
        FNome: string;
    public
        property Nome: string read FNome write FNome;
    end;

0 TLivro = class(TBaseClass)
    private
        FTitulo: string;
        FAutor: string;
        FDataPublicacao: TDateTime;
        FAnoDeAquisicao: Integer;
    public
        property Titulo      : string      read FTitulo      write FTitulo;
        property Autor       : string      read FAutor       write FAutor;
        property DataDePublicacao: TDateTime read FDataPublicacao write FDataPublicacao;
        property AnoDeAquisicao : Integer    read FAnoDeAquisicao write FAnoDeAquisicao;
    end;
end;
```



Como aumentar a produtividade com RTTI?

Dentro do meu formulário vou ter um **TMemo** para representar a estrutura que está sendo gerada pelas minhas classes. Assim como um botão que criará a instância de um livro.

No caso vamos adicionar uma função na nossa **TBaseClass** responsável por ler todas as propriedades vindas da nossa instância e retornar uma string com seus nomes e valores.

No deixamos restrito apenas para propriedades que tem o tipo string para serem representadas, posteriormente adicionaremos mais, conforme a necessidade.

```
function TBaseClass.GetRepresentacao: string;
const
  C_TYPE_STRING = [
    tkChar, tkWChar, tkString, tkLString, tkWString, tkUString
  ];
var
  LContext : TRttiContext;
  LType     : TRttiType;
  LProperty: TRttiProperty;
  LResult   : TStringList;
  LValue    : TValue;
  LValueStr: string;
begin
  LContext := TRttiContext.Create;
  LType     := LContext.GetType(Self.ClassType);

  LResult := TStringList.Create;
  try
    for LProperty in LType.GetProperties do
      begin
        LValueStr := '';
        LValue    := LProperty.GetValue(Self);

        if LValue.IsEmpty then
          Continue;

        if LProperty.DataType.TypeKind in C_TYPE_STRING then
          LValueStr := LValue.AsString;

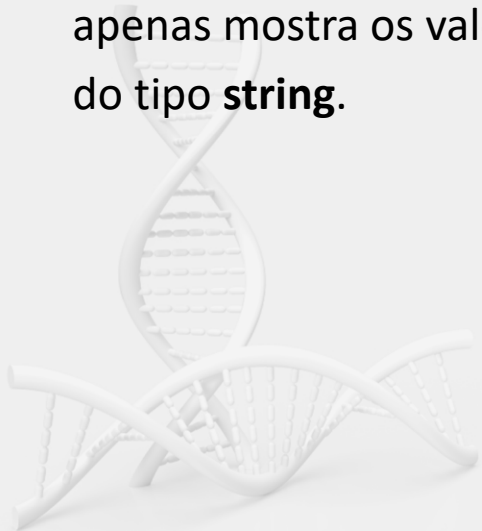
          if LValueStr.IsEmpty then
            Continue;

        LResult.AddPair(LProperty.Name, LValueStr);
      end;
    finally
      Result := LResult.Text;
      FreeAndNil(LResult);
    end;
  end;
```

Como aumentar a produtividade com RTTI?

Aqui temos o código responsável por instanciar a classe **TLivro**, definir suas propriedades e retornar sua representação e adiciona-la ao **TMemo**.

Abaixo temos o resultado da nossa função **GetRepresentacao** na instância feita, veja que apenas mostra os valores das nossas propriedades do tipo **string**.



```
procedure TFrmPrincipal.btnGenerateClick(Sender: TObject);
var
  LLivro: TLivro;
begin
  LLivro := TLivro.Create;
  try
    with LLivro do
    begin
      Titulo      := 'APRENDA A PROGRAMAR COM BORLAND DELPHI 7.0';
      Autor       := 'Fabíola Ventavoli';
      DataDePublicacao := StrToDate('04/11/2014');
      AnoDeAquisicao  := 2015;
    end;

    mmoRepresentacao.Lines.Text := LLivro.GetRepresentacao;
  finally
    FreeAndNil(LLivro);
  end;
end;
```

Exemplo 03

Titulo=APRENDA A PROGRAMAR COM BORLAND DELPHI 7.0
Autor=Fabíola Ventavoli

Gerar Instância(s) e Representar

Como aumentar a produtividade com RTTI?

Agora vamos declarar quais são os tipos **Float** e os tipos Inteiros.

Também vamos colocar uma parte para representar os dados desses dois tipos.

Note eu para o tipo **Float** adicionei mais uma verificação, pois o tipo **TDateTime** ele é uma representação do tipo **Float**.

Veja que apenas de adicionar essas representações ele já consegue interpretar vários tipos de propriedades, mesmo que “descobrimo” as propriedades da minha classe em tempo de execução.

```
60 C_TYPE_FLOAT = [  
    tkFloat  
];  
C_TYPE_INTEGER = [  
    tkInteger, tkInt64  
];  
  
if LValue.IsEmpty then  
    Continue;↑  
  
if LProperty.DataType.TypeKind in C_TYPE_STRING then  
    LValueStr := LValue.AsString  
else if LProperty.DataType.TypeKind in C_TYPE_FLOAT then  
begin  
    if LProperty.DataType.QualifiedName.Contains('TDateTime') then  
        LValueStr := FormatDateTime('dd/mm/yyyy', LValue.AsExtended)  
    else  
        LValueStr := FormatFloat('0.00', LValue.AsExtended);  
    end  
else if LProperty.DataType.TypeKind in C_TYPE_INTEGER then  
    LValueStr := IntToStr(LValue.AsInt64);  
  
if LValueStr.IsEmpty then  
    Continue;↑
```

Exemplo 03

Titulo=APRENDA A PROGRAMAR COM BORLAND DELPHI 7.0

Autor=Fabíola Ventavoli

DataDePublicacao=04/11/2014

AnoDeAquisicao=2015

Gerar Instância(s) e Representar

Como aumentar a produtividade com RTTI?

Agora vamos dificultar um pouco mais, vamos criar a classe **TEstante**, classe que terá um **nome** para identificação e um conjunto de livros (**TList<TLivro>**).

Assim como uma classe chamada **TBiblioteca**, que tem um **nome** e um conjunto de Estantes (**TList<TEstante>**).

Obs: Também faremos estas classes herdando de **TBaseClass**.

```
34 TEstante = class(TBaseClass)
    private
        FLivros: TList<TLivro>;
    public
        property Livros: TList<TLivro> read FLivros write FLivros;
        property Nome;
40 end;

TBiblioteca = class(TBaseClass)
    private
        FEstantes: TList<TEstante>;
    public
        property Estantes: TList<TEstante> read FEstantes write FEstantes;
        property Nome;
    end;

end;
library nome;
library 'Estantes': TList<TEstante> library 'Estantes' library 'Estantes';
```

Como aumentar a produtividade com RTTI?

Notem que as classes que estou desenvolvendo agora, elas possuem um **TList** de uma classe diferente, e como herdam de **TBaseClass**, irei usar o **RTTI** para instanciar e destruir estas variáveis. Assim não precisando me preocupar todas as vezes com isso nas classes que forem sendo herdadas.



```
{ TBaseClass }  
  
constructor TBaseClass.Create;  
var  
    LContext : TRttiContext;  
    LType     : TRttiType;  
    LProperty: TRttiProperty;  
    LValue    : TValue;  
    LMethod   : TRttiMethod;  
  
begin  
    inherited;  
  
    LContext := TRttiContext.Create;  
    LType    := LContext.GetType(Self.ClassType);  
  
    for LProperty in LType.GetProperties do  
        begin  
            LValue := TValue.Empty;  
  
            if LProperty.DataType.TypeKind in C_TYPE_CLASS then  
                begin  
                    for LMethod in LProperty.DataType.GetMethods('Create') do  
                        begin  
                            if Length(LMethod.GetParameters) = 0 then  
                                begin  
                                    LValue := LMethod.Invoke(LProperty.DataType.AsInstance.MetaclassType, []);  
                                    Break;  
                                end;  
                            end;  
                        end;  
                    end;  
  
                    if LValue.IsEmpty then  
                        Continue;  
  
                    LProperty.SetValue(Self, LValue);  
                end;  
            end;  
        end;  
    end;  
end;
```

Como aumentar a produtividade com RTTI?

Agora vamos complicar mais ainda, vamos identificar se a propriedade é do tipo **Class**, e contém o nome **TList**, e também deve ser herdado da nossa **TBaseClass**.

Quando isso ocorrer, vamos transformar o **TList** em um Array com o método **ToArray**, fazer um looping por todos os itens deste array e pegar o resultado da função **GetRepresentacao** deste objeto, para adicioná-lo a nossa representação dentro do **TMemo**.



```
else if LProperty.DataType.TypeKind in C_TYPE_CLASS then
begin
  if LProperty.DataType.QualifiedName.Contains('TList<') then
  begin
    LGenClass := LProperty.DataType.QualifiedName;
    LGenClass := Copy(LGenClass, Pos('<', LGenClass) + 1);
    SetLength(LGenClass, Length(LGenClass) - 1);

    LGeneric := LContext.FindType(LGenClass);
    if LGeneric.AsInstance.MetaClassType.InheritsFrom(TBaseClass) then
    begin
      for LMethod in LProperty.DataType.GetMethods('ToArray') do
      begin
        if Length(LMethod.GetParameters) = 0 then
        begin
          LValue := LMethod.Invoke(LValue, []);
          Break;
        end;
      end;

      for LMethod in LGeneric.GetMethods('GetRepresentacao') do
      begin
        if Length(LMethod.GetParameters) = 0 then
        begin
          for LCount := 0 to (LValue.GetArrayLength - 1) do
          begin
            LItemVal := LMethod.Invoke(LValue.GetArrayElement(LCount), []);
            LOtherRes.Add(LItemVal.AsString);
          end;
          Break;
        end;
      end;
    end;
  end;
end;
```

Como aumentar a produtividade com RTTI?

E agora só como um tratamento, vamos adicionar uma indentação nas nossas propriedades que vierem de um objeto herdado de **TBaseClass**.



```
end;  
finally  
  if not LOtherRes.IsEmpty then  
    LResult.Add(TRegex.Replace(LOtherRes.Text.TrimRight([#10]),  
      '^', ' ', [roIgnoreCase, roMultiline]));  
    Result := LResult.Text.TrimRight([#10]);  
    FreeAndNil(LResult);  
    FreeAndNil(LOtherRes);  
  end;  
end;
```

Exemplo 04

Nome=Biblioteca 01
Nome=Estante 01
Titulo=APRENDA A PROGRAMAR COM BORLAND DELPHI 7.0
Autor=Fabíola Ventavoli
DataDePublicacao=04/11/2014
AnoDeAquisicao=2015

Gerar Instância(s) e Representar

Gerar Instância(s) e Representar

Dúvidas?



Embarcadero Conference 2024


Inovação faz parte do nosso DNA!



Quer me ver na
#ECON25?
Acesse o QRCode
e avalie minha palestra!




Wesley Capelari

 [@wesley.capelari](https://twitter.com/wesley.capelari)

 [wesley-capelari](https://www.linkedin.com/in/wesley-capelari)

 wesley.capelari@gmail.com

 [\(18\) 9 9602 4649](tel:(18)996024649)

 github.com/wesleycapelari

