

Delphi

Coluna Borland - .NET - BorCon - ASP.NET - Delphi 2005 - Cartas

www.clubedelphi.net

feita para desenvolvedores

# ClubeDelphi

5 ano

Edição 67 R\$ 8,90

Borland



Iniciante

## InterBase/Firebird

Utilitários para administração e manutenção

## Delphi e ASP.NET

Criando Componentes e Controles

## Data Explorer

Gerenciando BDs a partir da IDE

**P00**

# Programação Orientada a Objetos com Delphi

**Crie cadastros e pesquisas usando técnicas de P00**

UML e engenharia reversa de código-fonte

Padrões de projeto e P00

Entrevista sobre P00 com Adail Muniz

Intermediário

## IBReplicator

Replicação de bancos IB/FB

## Novidades da Delphi Language

Entenda o strict private / protected

Avançado

## Expressões Regulares

Pesquisas sofisticadas

Especial:

**Delphi 2005 e .NET  
Compact Framework**

Desenvolvimento Delphi para  
Pocket PCs e Celulares





# Proteja seu Software



ANTIPIRATARIA  
DISTRIBUIÇÃO SEGURA  
AUMENTO DOS SEUS LUCROS  
PROTEÇÃO DA  
PROPRIEDADE INTELECTUAL

## **Compact**500<sup>®</sup>

Segurança. Qualidade. Eficiência.

Isto é a solução Compact-500. Você tem a garantia de proteção do seu software contra pirataria e engenharia reversa, além de controle total do gerenciamento de seu programa. Tudo isto de maneira fácil e extremamente eficiente.

Só uma empresa com um histórico de inovação, bom atendimento e sucesso pode garantir a segurança do seu bem mais valioso: suas informações.

**Compact-500**  
*A segurança do seu software*



São Paulo (11) 3392 4600

[www.safenet-inc.com/brasil](http://www.safenet-inc.com/brasil)

# ClubeDelphi

Edição 67 . Outubro. 2005 . Ano V . R\$ 8,90

## Coordenador editorial

Gladstone Matos  
gladstone@clubedelphi.net

## Editor geral

Guinther Pauli  
guinther@clubedelphi.net

## Editor técnico e Revisor

Luciano Almeida Pimenta  
lucianopimenta@clubedelphi.net  
Vinícius Oliveira de Andrade  
viniciusoandrade@yahoo.com.br

## Comissão Editorial

Adriano dos Santos  
Bruno Lichot  
Gustavo Chaurais

## Contribuíram nesta edição

Laércio Queiroz, Andreano Lanusse,  
Cristiano Caetano, Amandio Aguiar,  
Luciano Pimenta, Everson Volaco,  
Paulo Roberto Quicoli, Guinther Pauli,  
Isaque Pinheiro, Adail Muniz .

## Projeto Gráfico, Ilustrações e Capa

Antonio de Sousa Jr.  
antonio@devmedia.com.br

## Supervisão técnica

Vinícius Andrade  
alfredo@devmedia.com.br

## Publicidade

David Niegeski  
publicidade@devmedia.com.br

## Gerente de Marketing

Kaline Dolabella  
kalined@terra.com.br

## Na Web

www.clubedelphi.net

## Atendimento ao leitor

Cristiany Queiroz  
admin@clubedelphi.net  
(21) 2283-9012

## Distribuição

Fernando Chinaglia Distribuidora S/A

Realização



Apoio



A senha deste mês para o Portal do  
Assinante é: **XCA3ABI**

## Editorial

O desenvolvimento orientado a objetos encanta. Quem não gostaria de ter um software bem projetado, com camadas e classes bem definidas, separação clara entre regras de negócio, acesso a dados e tratamento de interface, ao mesmo tempo desfrutando de uma linguagem elegante, produtiva e robusta?

Já vi muito desenvolvedor, devido a softwares mal projetados (ou mesmo devido a prazos apertados), sair colocando código SQL e regra de negócio em evento *OnClick* de botão. Atire a primeira pedra quem nunca o fez! O apelo RAD da ferramenta as vezes nos leva a utilizar práticas não muito interessantes de programação, que podem dificultar (e muito) a manutenção do código, aumentar o impacto da mudança de um requisito e assim por diante.

Polimorfismo, herança e encapsulamento funcionam muito bem para "bolas, quadrados, carros etc." – afinal, a maioria dos exemplos e materiais sobre o assunto que vemos atualmente usam esses recursos para expressar a POO. Mas como aplicar isso em um exemplo do mundo real, e o melhor de tudo, tirar grande proveito dele?

Amigo leitor, não tenha dúvida que desenvolver aplicações realmente orientadas a objetos vai trazer uma grande flexibilidade para sua solução, facilitar a manutenção, a dependência de tecnologia e a codificação.

Nesta edição, procuramos incentivar você desenvolvedor Delphi, a utilizar a linguagem em sua plenitude, através de uma ferramenta extremamente RAD (e imbatível) sem deixar de lado o poder da orientação a objetos, em aplicações reais.

## Boa leitura!

Guinther Pauli

guinther@clubedelphi.net

Editor



## Índice

### 06 - Interbase/Firebird

Ferramentas e utilitários  
LUCIANO PIMENTA

### 08 - Database Explorer

Configurando e Gerenciando Banco de Dados  
EVERSON VOLACO

### 11 - Borland Conference

Retrospectiva  
GUINThER PAULI

### 12 - Expressões

Conceito e técnicas com Regex no Delphi  
CRISTIANO CAETANO

### 15 - IB Replicator

Replicação de dados para IB/FB  
EVERSON VOLACO

### 21 - POO na Prática

Criando classes, regras de negócios e aplicações na prática  
ISAUQUE PINHEIRO

### 29 - OOrigem

Entrevista sobre Orientação a Objetos com  
ADAIL MUNIZ

### 34 - Padrões de Projetos e POO

Parte I - Preparando-se para mudanças  
PAULO ROBERTO QUICOLI

### 38 - Novidades do Delphi Language

Parte IV - Os novos especificadores de acesso/visibilidade  
LAERCIO QUEIROZ

### 40 - Criando controles ASP.NET

User Controls e Web Controls  
LAERCIO QUEIROZ

### 43 - Compact Framework

Desenvolvendo aplicações Delphi para celulares e pockets PCs  
ANDREANO LANUSSE

### 48 - ESS-Model

Obtenha diagrama de classes por meio de engenharia reversa  
CRISTIANO CAETANO

### 50 - Coluna Borlnad

Borland Conference IV - eXtreme Performance  
AMANDIO AGUIAR

## Downloads relacionados as matérias

[www.devmedia.com.br/clubedelphi/downloads](http://www.devmedia.com.br/clubedelphi/downloads)

# Cartas

Tenho uma dúvida com relação a edição 65 da ClubeDelphi, sobre o desenvolvimento para .NET. No artigo é mostrada uma figura sobre as camadas envolvidas no desenvolvimento .NET e pelo que se percebe é que o .NET Framework funciona sobre uma camada API Win32, isso é fato? Pois sempre pensei que o .NET seria um substituto da API Win32, mas pelo que entendi na figura o .NET "depende" da existência da API Win32, essa informação está correta? Caso essa informação seja verdadeira, a API Win32 "sempre" vai existir no ambiente Windows, desmentindo assim boatos de que futuras versões do Windows não suportariam aplicações Win32, apenas .NET?

Ricardo Kazuo

Olá Ricardo. Essa é realmente uma dúvida muito comum e que gera confusão. Para facilitar a explicação e entendimento, coloquei a referida figura junto a esta sessão.

O .NET Framework depende sim do Win32, assim como a VCL ainda depende hoje. No entanto, o .NET foi totalmente construído para operar em cima da API Win32 (sobre ela), de forma que os desenvolvedores não mais "enxerguem" detalhes do Sistema Operacional. Por exemplo, quando você configura o *Caption* de um *TForm* (VCL) ou o *Text* de um *WinForm* (FCL), o framework "mapeia" essa chamada para uma API do S.O. (que nem eu nem você precisamos enxergar).

O .NET é sim o sucessor do modelo Win32, mas esse último ainda continua "embaixo". Ou seja, .NET "esconde" a API, e é claro oferecendo um modelo bem mais elegante de desenvolvimento, orientado a objetos, sem tipos esotérico (*PChars*, *Handles* etc.) ou rotinas estruturadas impostas pela atual API do Windows. Isso não é novidade para nós Delphianos: a VCL durante anos nos abstraiu de detalhes específicos da API.

Então, qual a jogada? Quando todos estiverem trabalhando diretamente sobre a plataforma .NET e não mais com Win32, a Microsoft pode "retirar" o Win32 como camada base, implementando assim os objetos do framework diretamente no Kernel (isso está prometido para o Windows Vista). Seu código e sua aplicação não serão afetados, pois conforme comentei, ele não usará mais a o Win32 diretamente (apesar de hoje ainda fazer uso em nível mais baixo).

Além disso, como o seu código executável compilado não é mais específico do sistema operacional e da CPU, é possível que ele seja executado em outro S.O. (é o fim da dependência binária). A especificação do .NET é aberta para quem quiser ver, incluindo detalhes da CLR (*Common Language Runtime* – a máquina virtual do .NET) e código intermediário (IL – *Intermediate Language*). Assim, é possível que seja fornecida uma implementação do .NET para um novo S.O. (leia-se mapeamento do código intermediário para uma API específica). A implementação mais conhecida hoje se chama *Mono* ([www.mono-project.com](http://www.mono-project.com)).

Por exemplo, considere o seguinte código Delphi:

```
procedure TWinForm7.TWinForm7_Load(  
    sender: System.Object; e: System.EventArgs);  
begin  
    Text := 'Olá Mundo';  
end;
```

Quando você compilar isso com o Delphi for .NET, será gerado um código intermediário, independente. Para o código anterior o equivalente em IL é o seguinte (observe que a sintaxe é bem parecida com Assembler):

```
.method private instance void TWinForm7_Load(  
    object sender, class [mscorlib]System.EventArgs e)  
cil managed  
{  
    // Code size      12 (0xc)  
    .maxstack 2  
    IL_0000: ldarg.0  
    IL_0001: ldstr      bytearray (4F 00 6C 00 E1  
    00 20 00 4D 00 75 00 6E 00 64 00  
        // 0.1... .M.u.n.d. 6F 00 )  
        // o.  
    IL_0006: callvirt instance void  
    [System.Windows.Forms]System.Windows.Forms.  
    Control::set_Text(string)  
    IL_000b: ret  
} // end of method TWinForm7::TWinForm7_Load
```

Para descompilar seu código EXE, você pode utilizar *ildasm.exe* que acompanha o SDK do .NET.



.NET FCL, VCL e Win32

Cartas publicadas podem ser editadas por motivos de clareza ou extensão.

**Guinther Pauli** - Redação  
**Luciano Pimenta** - Redação  
[revista@clubedelphi.net](mailto:revista@clubedelphi.net)

Problemas de performance ?

Corrupções frequentes ?

Se o problema é banco de dados,  
consulte nossos especialistas



**InterBase® 7.5**

Distribuidor  Oficial

**Borland®**

INTERBASE  SPECIALIST

Otimização de performance - Consultoria - Treinamento - Configuração - Monitoramento

## Conheça nossos produtos e serviços

[www.presence.com.br](http://www.presence.com.br)

### ■ Sistema para software-houses

- Gerenciamento completo de Help-desk
- Produção de software
- Time-sheet
- Controle de mudanças

### ■ Desenvolvimento de projetos

- Elaboramos projetos para Web Services
- Asp .net
- Windows forms .Net
- Win32

### ■ Aplicações móveis

- Desenvolvimento de aplicações para Palms / Celulares
- Acesso a bancos de dados

### ■ IBPartner

- Conheça o programa de apoio e incentivo a desenvolvedores com aplicações com InterBase
- Treinamentos
- Workshops
- Preços especiais

### Vendemos toda a linha Borland

Ferramentas de desenvolvimento  
Delphi 2005/ JBuilder / C# / Kylix  
Ciclo de desenvolvimento  
StarTeam / Caliber / Together / Optimizeit  
Consultoria em CMMi  
Borland TeraQuest  
Bancos de dados  
InterBase / JDataStore

Consulte nossos preços e parcelamentos



Informações e vendas: (11) 3862-7471

[bsp@presence.com.br](mailto:bsp@presence.com.br)

[www.presence.com.br](http://www.presence.com.br)



# InterBase/Firebird

Luciano Pimenta

## Ferramentas e utilitários

Confirma neste artigo uma lista de importantes utilitários para auxiliar no seu dia a dia como desenvolvedor Delphi para InterBase / Firebird. Para cada ferramenta, coloquei uma breve descrição, a empresa fabricante e o endereço para download. Bom proveito!

### Advanced Data Generator

Da empresa Upscene ([www.upscene.com](http://www.upscene.com)), a ferramenta gera valores para os campos da tabela do banco de dados, ideal para testes de aplicações com dados no banco ainda em período de desenvolvimento. Possui entre outras características: geração de dados randômicos, utiliza dados de arquivos externos, suporte a *Blobs*, conexão via ODBC/ADO. Possui versões para IB e FB separadas.

### Database Comparer

Produzida por Clever Components ([www.clevercomponents.com](http://www.clevercomponents.com)), ajuda a comparar e atualizar a estrutura de um banco de dados. Você pode comparar arquivos de banco de dados ou scripts. A ferramenta suporta IB, FB, Yaffil e SQL Anywhere.

### IB Log Manager

Também da Upscene ([www.upscene.com](http://www.upscene.com)), é utilizada para log de alteração de dados em bancos IB/FB. Uma excelente integração de *log* com auditoria do banco de dados.

### MS SQL and Access To InterBase Migration

Como o próprio nome diz, a ferramenta faz a migração de base de dados SQL Server e Access para IB/FB. Desenvolvida por Marcelo Lopez Ruiz ([www.ibphoenix.com/main.nfs?a=ibphoenix&page=sql2gdb](http://www.ibphoenix.com/main.nfs?a=ibphoenix&page=sql2gdb)), é muito prática em sua utilização, realiza a migração inteira do banco Access/SQL Server para IB/FB.

### IB DataPump

Produzida por Clever Components ([www.clevercomponents.com](http://www.clevercomponents.com)), realizada a migração de diferentes bases de dados (utilizando conexões BDE/ADO/ODBC), para IB/FB. Você pode gerar o script para o banco IB/FB, como também migrar diretamente os dados de um banco para outro.

### Mais

Veja a seguir a lista de mais alguns interessantes utilitários:

- **IBBackupSurgeon** e **IBFirstAID** ([www.ibsurgeon.com](http://www.ibsurgeon.com)) - utilitários que auxiliam na recuperação de BD corrompidos, fazendo diagnósticos, validações, testes de integridade etc.
- **DeZign For Databases** ([www.datanamic.com](http://www.datanamic.com)) - criação de modelos E/R e geração de código SQL;
- **Case Studio 2** ([www.casestudio.com](http://www.casestudio.com)) - ferramenta case para geração de diagramas e código SQL para vários bancos, incluindo FB;
- **SQL Debugger & SQL Editor** ([www.in-forms.co.za/ifsdownload.htm](http://www.in-forms.co.za/ifsdownload.htm)) - depurador de Stored Procedures para IB;
- **MiTeC InterBase Performance Monitor** ([www.mitec.cz/ibtools.htm](http://www.mitec.cz/ibtools.htm)) - Monitor de performance;
- **IBHeartbeat** ([www.fleetriver.demon.co.uk](http://www.fleetriver.demon.co.uk)) - Profiling e Performance;
- **Set Statistics** ([www.clientel.at/public](http://www.clientel.at/public)) - Otimização de bancos Firebird;
- **IBSite** ([www.madrigal.com.au/products/ibsite/default.htm](http://www.madrigal.com.au/products/ibsite/default.htm)) - análise de Stored Procedures e Triggers; **nops**
- **Report Master** ([www.studioplus.com.ua/ReportMaster/reportmaster.phtml](http://www.studioplus.com.ua/ReportMaster/reportmaster.phtml)) - para criação de relatórios a partir de BD;
- **InterConnect** ([www.lumensoft-int.com/index.php?D=1](http://www.lumensoft-int.com/index.php?D=1)) - para exportação / importação de dados TXT a partir de bancos IB/FB;
- **IBAnalyst** ([www.ibsurgeon.com](http://www.ibsurgeon.com)) - para análise e estatísticas de bancos IB/FB;

Para mais informações de utilitários, consulte os links:

- [bdn.borland.com/article/0,1410,30126,00.html](http://bdn.borland.com/article/0,1410,30126,00.html)
- [www.ibphoenix.com/main.nfs?a=ibphoenix&page=ibp\\_contrib\\_download](http://www.ibphoenix.com/main.nfs?a=ibphoenix&page=ibp_contrib_download)



**Luciano Pimenta** ([lucianopimenta@clubedelphi.net](mailto:lucianopimenta@clubedelphi.net)) é Técnico em Processamento de Dados, Editor Técnico da Revista ClubeDelphi e Editor do Portal ClubeDelphi.NET

([www.clubedelphi.net](http://www.clubedelphi.net))



**O** que é precioso você não entrega nas mãos de qualquer um. Por isso, quando pensar em Internet Data Center, procure a líder em número de clientes no Rio. Procure a ALOG. Utilizamos as mesmas estruturas da antiga filial carioca da .comDominio, o que lhe garante uma superestrutura em um prédio-cofre com sistemas de controle de acesso, de detecção e combate a incêndio com gás FM200, ambiente climatizado, no breaks e geração própria de energia de 1200KVA's, infra-estrutura de redes e acesso redundantes com monitoramento 24x7. Tudo para atender aos sistemas de missão crítica dos nossos clientes. Mas não é só isso. A ALOG ainda dispõe de atendimento personalizado, eficiente e ágil, com capacidade para atender aos nossos mais de 240 clientes corporativos com rapidez. Saiba tudo o que podemos fazer pela sua empresa. Ligue para nós, (21) 3083-3333.

## ALOG, SEUS DADOS BEM-CUIDADOS.



Controle de incêndio com FM200



1200KVA's de geração própria



Controle de acesso



Portas blindadas

**ALOG**  
data centers  
[www.alog.com.br](http://www.alog.com.br)



# Database Explorer

Everson Volaco

## Configurando e gerenciando banco de dados

**A** ferramenta Database Explorer que acompanha o Delphi 2005 traz diversas opções para acesso e gerenciamento de banco de dados. O utilitário é semelhante ao antigo SQL Explorer, que utilizava BDE, porém faz acesso a dados através do BDP (Borland Data Provider). Temos também uma versão “embutida” da ferramenta, que pode ser acessada dentro da própria IDE: o Data Explorer permite gerenciar conexões e manipular dados sem sair do ambiente de desenvolvimento.

Uma vez conectado a um banco de dados, através do utilitário você pode criar novas tabelas, acessar e modificar tabelas existentes, acessar views e até visualizar e testar stored procedures armazenadas no banco. Neste artigo conheceremos e abordaremos todas as funcionalidades disponíveis no Database Explorer que acompanha o Delphi 2005.

**Nota:** Neste artigo utilizarei o termo *Database Explorer* para fazer referência ao utilitário, porém, lembre-se que a maioria das funcionalidades do *Database Explorer* apresentadas neste artigo também funciona para o *Data Explorer*, integrado à IDE.

### Conhecendo o Database Explorer

O *Database Explorer* pode ser acessado diretamente a partir do menu iniciar do Windows, através da opção *Iniciar|Programas>Borland Delphi 2005>Database Explorer* ou ainda através da IDE do Delphi, através do menu *View|Data Explorer*. Ao acessá-lo fora do ambiente de desenvolvimento, a janela *Database Explorer* é aberta, e nela temos várias operações disponíveis para serem executadas no BD.

A instalação do Delphi 2005 traz consigo alguns drivers para acesso

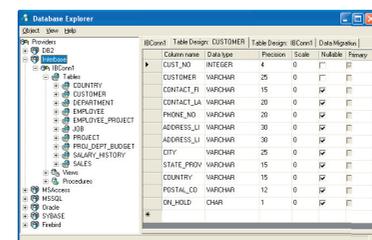
via BDP, como: DB2, InterBase, MS Access, MS SQL Server, Sybase e Oracle. Você pode ainda instalar novos *drivers* para serem utilizados com o BDP, como por exemplo, para acesso ao Firebird ou MySQL, usando normalmente dentro do *Database Explorer* ou em suas próprias aplicações BDP.

Dependendo do objeto que é selecionado dentro da janela, diferentes opções ficam disponíveis no menu de contexto e no menu *Object*. Os comandos disponíveis estão avaliados para os seguintes “nós” do *Treeview*, quando selecionados:

- um driver (DB2, Oracle, InterBase etc.);
- uma conexão;
- o nó *Tables*;
- uma tabela específica;
- uma *View* específica;
- uma *Stored Procedure* específica.

Você pode executar, ao mesmo tempo, várias operações em diferentes bancos de dados dentro do *Database Explorer*. Cada opção selecionada fica disponível no lado direito da janela dentro de uma aba específica (**Figura 1**).

**Nota:** O *driver* para Firebird que aparece na **Figura 1** não acompanha a instalação do Delphi 2005. Você pode baixar o driver a partir do site [www.firebirdsql.com](http://www.firebirdsql.com). Na opção *downloads*, baixe e instale o arquivo *Borland Data Provider for Firebird*. Observe que no mesmo endereço existe o *Data Provider* para ADO.NET, nativo do FB, que nesse caso, não usa o BDP e não pode ser integrado ao *Database Explorer*. No Portal do Assinante da Revista ClubeDelphi ([www.clubedelphi.net/portal](http://www.clubedelphi.net/portal)) existe um artigo de Luciano Pimenta que mostra como realizar a instalação e configuração do BDP para acesso ao FB.



**Figura 1.** Executando várias operações ao mesmo tempo no Database Explorer

Como comentado anteriormente, para acessar o utilitário a partir da IDE do Delphi, basta selecionar o menu *View|Data Explorer*. Selecionando essa opção você pode executar todas as funcionalidades do *Database Explorer* sem precisar sair do ambiente do Delphi, onde cada opção selecionada fica disponível em abas dentro da própria IDE!

Ao selecionar os objetos na janela do utilitário, um menu (chamado *Data Explorer*) é adicionado no menu da IDE, mostrando as opções disponíveis do objeto selecionado. Veja na **Figura 2** a janela *Data Explorer* dentro da IDE do Delphi 2005.



**Figura 2.** Data Explorer na IDE do Delphi 2005 para conexão e gerenciamento de bancos de dados

### Criando e manipulando uma conexão

Para criar uma nova conexão, basta clicar no *driver* desejado e selecionar a opção *Add New Connection* a partir do menu de contexto. Na janela *Add New Connection*, você tem a opção de alterar o *driver* que deseja utilizar através da opção *Provider Name*. No campo *Connection Name* digite o nome da conexão a ser criada e clique no botão OK.

Uma nova conexão é adicionada abaixo do *driver* selecionado. Para configurar os parâmetros da nova conexão, basta selecioná-la e clicar sobre a opção *Modify Connection* a partir do menu de contexto ou no menu *Data Explorer|Modify Connection*.

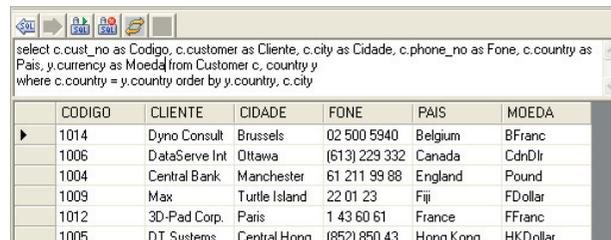
Na janela *Connections Editor*, devemos informar os parâmetros para conexão com o banco de dados. Essa janela mostra parâmetros diferentes, dependendo do *driver* que está sendo utilizado. Antes de clicar no botão OK para fechar a janela, você pode testar a conexão com o banco utilizando o botão *Test*.

Com a conexão selecionada no *Database Explorer*, você tem ainda a opção de atualizar, renomear ou apagar a conexão, ou chamar o editor de SQL para execução de instruções DDL ou DML (opção *SQL Window*).

**Nota:** Instruções DDL (*Data Definition Language*) são instruções SQL para definição de objetos para o banco de dados como *Create*, *Alter* e *Drop*. Instruções DML (*Data Manipulation Language*) são instruções para manipulação de dados usando *Select*, *Insert*, *Update* e *Delete*.

Para abrir o editor SQL, basta selecionar a opção *SQL Window* disponível no menu da conexão selecionada. Dentro desse editor você pode executar instruções SQL de qualquer tipo e com a opção de executá-la dentro de uma transação, utilizando a opção *Begin transaction* disponível na barra de botões.

Após digitar a instrução, basta clicar no botão *Execute SQL*. Veja o editor SQL na **Figura 3**. Você pode ainda navegar entre as instruções SQL executadas através das opções *Previous SQL* e *Next SQL*.

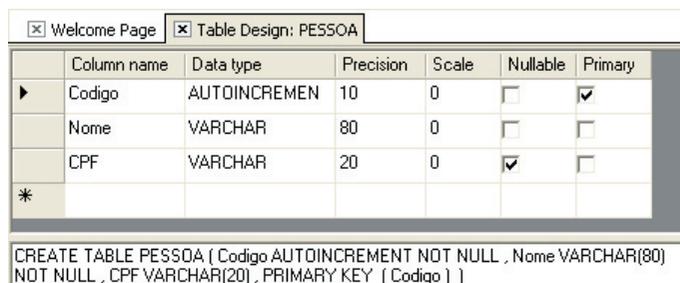


**Figura 3.** Utilizando o editor SQL do Database Explorer para execução de instruções SQL

### Manipulando tabelas

O *Database Explorer* permite que você crie ou altere tabelas do banco de dados, de forma visual, através do *Table Design*. Selecione o nó *Tables* presente dentro da conexão e a partir do menu de contexto clique sobre a opção *New Table* para abrir a janela *Table Design*. Dentro dessa janela você pode definir de forma visual todas as colunas da nova tabela, especificando o nome, tipo, tamanho, campos obrigatórios e a chave-primária.

Na coluna *Data type* os tipos de dados listados dependem do *driver* usado na conexão, isso é, caso estejamos criando uma nova tabela para um banco de dados InterBase, então os tipos disponíveis serão os do InterBase, se for para o MS Access, serão listados os suportados pelo Access, e assim por diante. Terminada a definição da tabela, basta clicar na opção *Save Table* a partir do menu de contexto, para criar a nova tabela no banco de dados. Selecionando a opção *Show DDL* é possível visualizar a instrução SQL para criação da tabela antes de criá-la fisicamente no banco de dados (caso você adicione ou remova alguma coluna, é necessário utilizar a opção do menu novamente para atualizar a visualização do comando SQL). Veja na **Figura 4** a criação de uma tabela.



**Figura 4.** Criando uma nova tabela em um banco de dados Access através do Database Explorer



**Nota:** Após salvar a nova tabela, caso ela não esteja aparecendo na lista, basta selecionar o nó *Tables* e executar a opção *Refresh* disponível no menu.

O *Database Explorer* permite ainda a você alterar a estrutura de tabelas já existentes no banco de dados. Para isso, selecione uma tabela específica e clique sobre a opção *Alter Table* disponível no menu de contexto. A janela *Table Design* é aberta trazendo a estrutura da tabela selecionada, onde é possível adicionar, alterar ou excluir campos. Ainda com a tabela selecionada, você pode apagá-la do banco de dados usando a opção de menu *Drop Table*.

Para visualizar os registros da tabela selecionada, basta clicar na opção *Retrieve Data From Table*. Você pode adicionar, alterar ou apagar os registros da tabela. Utilizando as opções *Update* e *Rollback* do menu de contexto da janela, você pode optar em salvar ou cancelar as operações.

Dentro do nó *Views* é possível visualizar os dados retornados por uma *view* do BD. Para isso, basta utilizar a mesma opção de visualização dos dados de uma tabela.

### Visualizando e testando Stored Procedures

O *Database Explorer* permite visualizar e testar as *stored procedures* armazenadas no BD. Expandido o nó *Procedures* de uma conexão, é possível visualizar todas as *stored procedures* armazenadas, onde através da opção de menu *View Parameters* podemos visualizar informações sobre cada parâmetro de entrada e saída, como também entrar com valores de teste para cada parâmetro.

Ao clicar na opção *View Parameters*, os parâmetros de entrada e saída da *stored procedure* selecionada são listados, trazendo informações sobre cada parâmetro, como nome, tipo, direção (entrada/saída) etc. Para testar uma *stored procedure*, basta entrar com um valor para os parâmetros de entrada através do campo *Value*, clicando a seguir sobre o botão *Execute*.

**Nota:** Caso a *stored procedure* que esteja sendo testada retorne um conjunto de dados, você pode marcar a opção *Stored procedure has one or more cursors* para visualizar o *Result Set*.

### Migração com "Copy/Paste"

Uma opção bastante interessante dentro do *Database Explorer* é a possibilidade de se fazer uma cópia de uma tabela e seus registros. Você pode selecionar uma tabela específica de um banco de dados, por exemplo do InterBase, copiar e colar dentro de um banco SQL Server, por exemplo.

Para isso, basta clicar na opção *Copy Table* (menu de contexto da tabela selecionada), selecionar a conexão onde a tabela será colada e entrar no nó *Tables*. A partir do menu de contexto, clicar sobre a opção *Paste Table* para colar a tabela e seus registros. No momento de colar a tabela, o *Database Explorer* solicita o nome para a tabela destino.

**Nota:** A opção de copiar e colar uma tabela funciona desde que o banco de dados de destino, que irá receber a tabela, possua suporte aos tipos de dados definidos na tabela que está sendo copiada.

### Migração com o "Migrate Data"

Selecionando um *driver* qualquer dentro da janela *Database Explorer*, podemos acessar a opção *Migrate Data* a partir do menu de contexto. O *Data Migration* utiliza a classe *BdpCopyTable* para fazer a cópia das tabelas entre os bancos de dados selecionados. Na paleta de componentes *Borland Data Provider* existe um componente chamado *BdpCopyTable*, que permite fazer migração de dados em tempo de execução através da sua própria aplicação.

Para fazer a migração de várias tabelas ao mesmo tempo entre bancos de dados diferentes, basta selecionar a conexão de origem em *Source Connection* e a de destino em *Destination Connection* dentro da janela *Data Migration*. Na lista de tabelas da conexão de origem, você pode selecionar as tabelas a serem migradas.

Clique no botão *Migrate* para fazer a migração dos dados. As operações que não puderem ser executadas devido às diferenças entre os bancos de dados, são mostradas em um grid durante o processo de migração, juntamente com os comandos realizados com sucesso. Para visualizar as tabelas copiadas dentro da base de dados de destino, basta selecionar a conexão e clicar na opção *Refresh* para que os objetos da conexão sejam atualizados.

Para aplicações *Windows Forms* ou *ASP.NET* desenvolvidas tanto em Delphi como em C#, o *Database Explorer* integrado à IDE (o *Data Explorer*) disponibiliza uma funcionalidade para acelerar e facilitar o acesso a banco de dados. Após configurar uma conexão no *Data Explorer*, você pode selecioná-la e arrastá-la para dentro de um formulário *Windows* ou *Web*, para que seja adicionado o *BdpConnection* já configurado com os parâmetros da conexão.

Você pode ainda selecionar e arrastar uma tabela diretamente do *Data Explorer* para a aplicação, onde é criado, além do componente de conexão *BdpConnection* (caso ainda não exista), um *BdpDataAdapter* já configurado de acordo com a tabela arrastada.

**Nota:** Aplicações *VCL Forms Application - Delphi for .NET* não suportam BDP e por esse motivo não possuem a integração citada com a janela *Data Explorer*.

### Conclusões

Neste artigo conhecemos o *Database Explorer* e todas as suas funcionalidades no Delphi 2005. Vimos que através dele é possível definir e gerenciar conexões com bancos de dados diretamente da IDE, sem a necessidade de sair do ambiente de desenvolvimento, ganhando assim maior produtividade dentro da ferramenta. Um abraço e até a próxima!



**Everson Borges Volaco** ([everson@rhealeza.com.br](mailto:everson@rhealeza.com.br)) é desenvolvedor e instrutor certificado Borland, com experiência em aplicações cliente/servidor, usando Delphi, Interbase e Oracle. Possui três certificações oficiais Borland: Borland Delphi 7.0, Borland CaliberRM 6.0 e Borland StarTeam 6.0.



# Borland Conference

Guinter Pauli

## Retrospectiva

A 4ª Borland Conference está próxima! E nada melhor que lembrar um pouco as edições anteriores deste que sem dúvida é o maior evento Borland do país! Quando a Borland me convidou para contar nesta coluna um pouco sobre as edições passadas do evento, fiquei bastante entusiasmado, já que pude acompanhar de perto a BorCon desde a primeira versão.

A 1ª BorCon aconteceu em 2001 e foi primeira versão brasileira do evento que já acontecia há anos nos Estados Unidos. Acredito que foi a primeira grande oportunidade de reunir toda a comunidade Delphi do Brasil. A seguir um comentário que fiz na época:

*"Nos tempos da universidade, sempre participávamos de congressos e simpósios de informática, e ficávamos ansiosos para assistir aquela única palestra sobre Delphi, Desenvolvimento Internet ou Banco de Dados. Agora você já imaginou um evento onde o único assunto era tecnologias Borland, Delphi, Internet, Interbase, XML, .NET e Web Services? Onde os palestrantes eram as maiores autoridades da tecnologia tanto do Brasil como do exterior? Sim, a Borcon foi isso."*

Para quem esteve no evento e quiser matar a saudade, coloquei algumas fotos no endereço:

[www.clubedelphi.net/artigos/quintherborcon.html](http://www.clubedelphi.net/artigos/quintherborcon.html).

A segunda edição, que aconteceu em 2002, também foi um grande sucesso. Quem não lembra do "Borland Car" na abertura! Foi um grande evento onde tivemos ótimas palestras, principalmente sobre ALM. A seguir um trecho da cobertura que fiz na época:

*"O congresso começou com o diretor da Borland Latin América, José Rubens Tocci (o "Zé", como gosta de ser chamado) falando sobre a importância que nós desenvolvedores temos para a Borland, e que somos uns dos grandes motivos pelo atual sucesso da Borland. Falou ainda que a BorCon é um evento de desenvolvedores para desenvolvedores. Estávamos começando a sentir o espírito Borland!"*

*A seguir "Zé" perguntou quantos desenvolvedores presentes trabalhavam com Delphi. Para minha surpresa (e alegria), quase todos levantaram o braço! Senti que, sem dúvida nenhuma, o Delphi foi, é, e vai continuar sendo para a Borland a sua ferramenta de maior sucesso."*

Coloquei algumas fotos em:

[www.clubedelphi.net/artigos/quintherborcon2.htm](http://www.clubedelphi.net/artigos/quintherborcon2.htm) ClubeDelphi também acompanhou de perto a realização da 3ª Borland Conference Brasil. Além de excelentes palestras, muitas atrações

interessantes (e inusitadas!). Lembram dos simuladores de realidade virtual, construídos em Delphi?! A seguir um trecho da cobertura:

*"Na abertura do evento, José Rubens Tocci, diretor da Borland Latin América e Dormevilly Tertius agradeceram a participação de todos, ressaltando que a Borland só é uma empresa de sucesso hoje por um único motivo: nós desenvolvedores. Reafirmou que a empresa tem uma missão: ajudar seus clientes a moverem-se para o futuro sem abandonar o passado. Foram feitos questionamentos do tipo: "Quantas linhas de código vocês já digitaram?"; "Com o surgimento de novas tecnologias, como fica seu código, é preciso jogar tudo fora?"; Tocci fez ainda uma colocação bastante interessante com relação a esse ponto: "Programa é como um filho". A Borland sabe disso, e sabe que o uso de novas tecnologias exige responsabilidade, e que a maneira como se desenvolve software atualmente é bastante diferente de como fazíamos há anos atrás. Não são permitidas mais falhas com antigamente. Lembrou que, mais do que sistemas e novas tecnologias, o sucesso de um projeto de software depende de **pessoas**"*

Algumas fotos: [www.clubedelphi.net/artigos/BorCon2004.asp](http://www.clubedelphi.net/artigos/BorCon2004.asp).

Nas três edições tivemos muito Delphi, do melhor nível técnico, onde novas tecnologias suportadas no produto foram apresentadas sempre em primeira mão: .NET, XML, Web Services, UML e muito mais. Prezados leitores e desenvolvedores Delphi, não tenham dúvida que a 4ª Borland Conference vai ser um sucesso! Conto com a presença de você por lá! Mais informações em:

[borcon.borland.com.br](http://borcon.borland.com.br).

Um grande abraço!



**Guinter Pauli** é autor de mais de 100 artigos publicados e do livro "Delphi – Programação para Banco de Dados e Web". É Bacharel em Sistemas de Informação pelo Centro Universitário Franciscano (Unifra, Santa Maria – RS). Detém quatro certificações oficiais Borland: Delphi Advanced pela Borland dos Estados Unidos, Delphi Web Development Certified, Kylix Product Certified e Delphi Product Certified. É desenvolvedor 5 estrelas Microsoft, MCP, MCAD e MCSD.NET. Já ministrou palestras para mais de 5 mil pessoas em todo o país. É Editor Geral da revista ClubeDelphi ([www.devmedia.com.br/clubedelphi](http://www.devmedia.com.br/clubedelphi)) e Editor Técnico da Revista Web Mobile Magazine ([www.devmedia.com.br/webmobile](http://www.devmedia.com.br/webmobile)). Colunista dos portais ClubeDelphi, MSDN Magazine e Web Mobile Magazine. Pode ser contactado pelos endereços [guinther\\_pauli@hotmail.com](mailto:guinther_pauli@hotmail.com) ou [guinther@devmedia.com.br](mailto:guinther@devmedia.com.br).



# Expressões Regulares

Cristiano Caetano

## Conceitos e técnicas com REGEX no Delphi

Uma expressão regular, também chamada de RE ou REGEX, é uma ferramenta de pesquisa e substituição de textos extremamente sofisticada e amplamente utilizada. Usualmente, uma expressão regular é uma composição de caracteres e símbolos chamados *meta-caracteres*, cuja principal função é pesquisar um padrão conforme as condições fornecidas; nesse caso, diz-se que o texto encontrado “casou” com a expressão regular.

Muitas vezes, as expressões regulares são confundidas com caracteres curingas que normalmente utilizamos em pesquisas de arquivos, como por exemplo, “\*.doc”. Na verdade, as expressões regulares também são capazes de realizar pesquisas desse tipo, no entanto, elas são imensamente mais poderosas do que isso, permitindo escolher se a expressão a ser pesquisada está no começo ou no final da linha, quais os caracteres são permitidos, quantas vezes a expressão deve se repetir, entre outros tipos de pesquisas.

O tema “Expressões regulares” é tratado com certo receio e, às vezes, deixado um pouco de lado pelos desenvolvedores Delphi, em virtude de que poucas pessoas conhecem o seu real potencial.

Esse cenário ocorre, provavelmente, pela interpretação errônea de que expressões regulares são usadas somente pelos administradores de sistemas Linux/Unix ou por desenvolvedores de linguagens de script, tais como Perl, Ruby, entre outras.

Além disso, ao nos depararmos pela primeira vez com as expressões regulares, a primeira reação é de rejeição e desconfiança; afinal, quem é capaz de entender essas expressões “esquisitas”, como podem ser vistas no código a seguir:

```
^[a-z]{1,8}\.[a-z]{1,3}$
[_a-zA-Z\d\-\.\.]+@([\_a-zA-Z\d\-\.\.]+\.[\_a-zA-Z\d\-\.\.]+)
(FTP|HTTP):\/\/([\_a-z\d\-\.\.]+)(\.[\_a-z\d\-\.\.]+)
```

Como discutido anteriormente, podemos destacar três cenários onde as expressões regulares são comumente utilizadas:

- **Extração de dados:** Normalmente, esse cenário ocorre quando desejamos exportar dados de sistemas legados para arquivos intermediários (XML, CSV, TXT);
- **Pesquisa e substituição de textos:** Esse recurso é notoriamente utilizado por administradores de sistemas para pesquisar informações específicas em arquivos de log ou para realizar modificações em lote de textos localizados nos arquivos de configurações do sistema;
- **Validação de dados:** Esse cenário é utilizado em aplicações

Web ou aplicações convencionais para a validação de campos de entrada quando a utilização de máscaras de entrada não são o suficiente para a validação do dado digitado pelo usuário.

### Meta-Characteres

Assim como no aprendizado de uma língua estrangeira, você primeiro aprenderá o significado de cada palavra individualmente e, em seguida, aprenderá a unir essas palavras em frases mais complexas e coerentes no contexto de um diálogo. Seguindo essa técnica, o leitor poderá aprender a criar expressões regulares mais complexas e sofisticadas em virtude do completo entendimento de cada meta-caractere individualmente.

Cabe lembrar que, para você usufruir dos benefícios das expressões regulares, você deverá utilizar um “motor” de expressões regulares. Normalmente, esse motor já está integrado nas linguagens de programação atuais, caso contrário você poderá utilizar um motor de expressões regulares utilizando programas externos ou até mesmo bibliotecas criadas por terceiros. O motor de expressões regulares, dito em outros termos, é o compilador e interpretador das expressões regulares. Nesse caso, é ele quem analisa, interpreta e fornece os resultados que casaram com a expressão fornecida.

Para discutir esse assunto numa abordagem didática, os meta-caracteres serão identificados nas seções a seguir por seus respectivos *mneumônicos* e uma breve descrição. Além disso, você poderá ver alguns exemplos de utilização dos meta-caracteres e os seus casamentos possíveis.

### Ponto

O *ponto* (.) é um meta-caractere curinga, normalmente utilizado em expressões quando não se sabe exatamente qual o caractere que se quer pesquisar. Em resumo, o *ponto* casa com qualquer caractere existente, imprimível ou não, inclusive com o caractere literal *ponto*.

Mas, no entanto, devemos lembrar que, apesar de casar com qualquer caractere, o *ponto* é obrigatório e deverá existir no texto a ser pesquisado, caso contrário, a expressão regular não casará.

Expressão	Casamentos possíveis
n.o	não, nao, neo
bo.o	bozo, bolo, boto



## Asterisco

O *asterisco* (\*) é um tipo de caractere curinga que casa com zero, uma ou muitas ocorrências do caractere anterior a ele.

<i>Expressão</i>	<i>Casamentos possíveis</i>
6*0	0,60,666666660
bi*p	bp, bip, biiiiip

## Lista

A *lista* ([ ]) indica classes de caracteres que devem existir no texto a ser pesquisado, a fim de que a expressão regular case. Além disso, você poderá pesquisar classes de caracteres utilizando o caractere *hífen* para separar a faixa de caracteres que serão pesquisados (como por exemplo: 0-9, a-z, A-Z, entre outras).

Adicionalmente, muitos motores de expressões regulares também aceitam a utilização de palavras reservadas especiais que definem classes de caracteres padronizadas. A essas palavras reservadas, dá-se o nome de classes de caracteres POSIX.

A principal vantagem desse recurso é a legibilidade da expressão regular e o fato de que essas classes podem ser utilizadas em sistemas que precisem realizar pesquisas ou extração de textos escritos em outras línguas, porque elas abrangem caracteres especiais e acentuação. Você poderá ver alguns exemplos de classes de caracteres POSIX na **Tabela 1**.

<i>Expressão</i>	<i>Casamentos possíveis</i>
n[ãa]o	não, nao
1[.:]45	1:45, 1.45, 1 45

Classe POSIX	Expressão regular similar	Descrição
[[:upper:]]	[A-Z]	Letras maiúsculas
[[:lower:]]	[a-z]	Letras minúsculas
[[:alpha:]]	[A-Za-z]	Maiúsculas/minúsculas
[[:alnum:]]	[A-Za-z0-9]	Letras e números
[[:digit:]]	[0-9]	Números
[[:xdigit:]]	[0-9A-Fa-f]	Números hexadecimais
[[:punct:]]	[.,!?:...]	Sinais de pontuação
[[:blank:]]	[ \t]	Espaço e TAB
[[:space:]]	[ \t\n\r\f\v]	Caracteres brancos
[[:cntrl:]]	-	Caracteres de controle
[[:graph:]]	[^\t\n\r\f\v]	Caracteres imprimíveis
[[:print:]]	[^\t\n\r\f\v]	Imprimíveis e o espaço

**Tabela 1.** Exemplos de caracteres POSIX suportados por alguns motores de REGEX

## Lista Negada

A *lista negada* ([^ ]) tem o mesmo comportamento da lista, porém indica as classes de caracteres que não devem existir no texto a ser pesquisado a fim de que a expressão regular case.

<i>Expressão</i>	<i>Casamentos possíveis</i>
[^0-9]	a, b, c
[^a-z]	1, ?, #

www.clubedelphi.net

## Mais

O *mais* (+) tem um comportamento semelhante ao asterisco. A única diferença é que o *mais* não é opcional, nesse caso, o caractere anterior ao mais deve casar pelo menos uma vez.

<i>Expressão</i>	<i>Casamentos possíveis</i>
6+0	60,660,6660
bi+p	bip, biip, biiip

## Opcional

O *opcional* (?) é um meta-caractere curinga semelhante ao *ponto*. A principal diferença entre o *opcional* e o *ponto*, é que o caractere anterior ao *opcional* não é obrigatório, ou seja, pode ter ou não ter a ocorrência desse caractere para que a expressão regular case. Além disso, o *opcional* também é capaz de tornar outras regras definidas por outros meta-caracteres opcionais.

<i>Expressão</i>	<i>Casamentos possíveis</i>
bol?a	bola, boa
fala[r!]?	falar, fala!, fala

## Escape

O *escape* (\) é o meta-caractere utilizado quando você precisa pesquisar um caractere que é igual a um dos meta-caracteres utilizados pelas expressões regulares. Nesse caso, se você precisar pesquisar um caractere *ponto*, você terá que utilizar o *escape* para informar ao motor de expressões regulares que aquele *ponto* que está descrito na expressão regular não é o meta-caractere *ponto*, é na verdade o caractere que deve ser procurado.

<i>Expressão</i>	<i>Casamentos possíveis</i>
\.	.
\?	?

# PENSE...

QUANTO TEMPO  
VOCÊ GASTARIA  
PARA DESENVOLVER  
COBRANÇA COM BOLETOS  
BANCÁRIOS PARA  
APENAS UM BANCO  
NO SEU SOFTWARE

# COBREBEMX

- 56 BANCOS E MAIS DE 430 CARTEIRAS DE COBRANÇA PARA IMPRESSÃO E/OU ENVIO DE BOLETO BANCÁRIO POR EMAIL;
- GERAÇÃO DE BOLETOS ON LINE;
- GERAÇÃO E LEITURA DE ARQUIVOS (REMESSA/RETORNO) NOS PADRÕES FEBRABAN E CNAB;
- MAIS DE 40 EXEMPLOS EM DIVERSAS LINGUAGENS DE PROGRAMAÇÃO



DOWNLOADS E INFORMAÇÕES EM WWW.COBREBEM.COM

Tecnologia

## Circunflexo

O *circunflexo* (^) é uma espécie de “âncora” que indica os caracteres que devem estar no começo da linha ou texto a ser pesquisado, para que a expressão regular case.

Expressão	Casamentos possíveis
^Delphi	Delphi
^[0-9]	0, 1, 3

## Cifrão

O *cifrão* (\$) é uma espécie de âncora semelhante ao *circunflexo*. Porém, a sua função é indicar os caracteres que devem estar no final da linha ou texto a ser pesquisado, para que a expressão regular case.

Expressão	Casamentos possíveis
\.\$	.
^Delphi 8.\$	Delphi 8.

## Ou

O meta-caractere *ou* (|) é utilizado para criar expressões regulares com mais de uma alternativa possível.

Expressão	Casamentos possíveis
http://ftp://	http://, ftp://
Delphi Borland	Delphi, Borland

O meta-caractere *grupo* (()) tem a função de agrupar caracteres e meta-caracteres em blocos de expressões regulares. Além da criação de blocos de expressões, o grupo tem a função de ampliar a utilização dos outros meta-caracteres. Cabe lembrar que grupos podem conter outros grupos dentro dele.

Expressão	Casamentos possíveis
Delphi (7 8)	Delphi 7, Delphi 8
(oi)+	oi, oioi, oioioi

## Chaves

O meta-caractere *chaves* ({m,n}) tem a função de determinar exatamente a quantidade de vezes que um texto deve repetir para que a expressão regular case.

Expressão	Casamentos possíveis
macar{1,2}ao	macarrao, macarao
p{1}ascal	pascal

## Expressões regulares em ação

Infelizmente, o Delphi não incorpora nativamente nenhum motor de expressões regulares. Somente nas versões mais novas você poderá utilizar esse recurso por meio das classes de expressões regulares fornecidas pelo .NET Framework. No entanto, como discutimos anteriormente, você poderá utilizar bibliotecas de terceiros a fim de incorporar as expressões regulares no seu software.

A fim de fornecer uma solução concreta, vamos utilizar a biblioteca *TRegExpr* nos exemplos que vamos apresentar a seguir. A biblioteca *TRegExpr* é de fácil instalação e não precisa de nenhum procedimento trabalhoso para a sua utilização, além disso, não utiliza nenhuma biblioteca externa (DLL) para o seu funcionamento.

Cabe ainda lembrar que essa biblioteca é compatível com várias versões do Delphi e outros ambientes de desenvolvimento (Delphi 2-7, C++ Builder 3-6, Kylix, FreePascal). Você poderá fazer o download e obter maiores informações sobre a biblioteca *TRegExpr* no endereço [regexpstudio.com](http://regexpstudio.com).

A título de exemplo, vamos utilizar a biblioteca *TRegExpr* para validar a digitação do campo e-mail num formulário de cadastro. Nesse cenário, ao sair do campo e-mail (evento *OnExit* do controle), o texto digitado será analisado segundo as regras definidas na expressão regular, como pode ser visto no trecho de código a seguir:

```
if not ExecRegExpr('[_a-zA-Z\d\-\.\.]+@[[_a-zA-Z\d\-\.\.]+\+(\.\[_a-zA-Z\d\-\.\.]+\)+)') then
  TEdit(Sender).Text := ShowMessage('E-mail incorreto. Digite novamente.');
```

Entre outras características, a biblioteca *TRegExpr* fornece procedimentos automatizados que são baseados internamente em expressões regulares, porém você poderá utilizá-los sem precisar escrever uma única linha de expressão regular.

Seguindo esse princípio, você poderá, por exemplo, realizar uma pesquisa e substituição de textos automática utilizando um método chamado *ReplaceRegExpr*, como pode ser visto na **Listagem 1**.

### Listagem 1. Substituindo texto usando expressões regulares da TRegExpr

```
var
  TextoProcurar, TextoSubstituir: string;
begin
  TextoProcurar := 'Java';
  TextoSubstituir := 'Delphi';
  Memo1.Text := ReplaceRegExpr(TextoProcurar,
    Memo1.Text, TextoSubstituir);
end;
```

## Conclusões

As expressões regulares podem e devem ser utilizadas por desenvolvedores Delphi. Elas são fáceis de usar e normalmente poupam muito trabalho, principalmente quando se trata de pesquisa e substituição de textos.

Este artigo abordou somente os conceitos básicos das expressões regulares, no entanto, os motores de expressões regulares normalmente fornecem recursos adicionais que normalmente facilitam ainda mais a vida dos desenvolvedores.

## Links

<http://regexpstudio.com>

Biblioteca *TRegExpr*

<http://aurelio.net/er>

Informações sobre Expressões Regulares e vários Links

[www.oreilly.com/catalog/regex](http://www.oreilly.com/catalog/regex)

Livro sobre Expressões Regulares



**Cristiano Caetano** ([cristiano.caetano@zerog.com](mailto:cristiano.caetano@zerog.com)) autor do livro *CVS: Controle de Versões e Desenvolvimento Colaborativo de Software*, atualmente é o *Quality Assurance Engineer da Proversoft/Macrovision*, empresa líder mundial em *Content & Software Value Management*.



# IBReplicator

Everson Volaco

## Replicação de dados para IB/FB

Neste artigo, conheceremos e abordaremos as principais funcionalidades do IBReplicator (*IBPhoenix*) destinado a replicação de dados para bancos de dados InterBase e Firebird.

Atualmente, na versão 2.1.4 (até o fechamento desta edição), o IBReplicator é a melhor e mais conhecida ferramenta para replicação de dados entre bancos de dados IB/FB.

Através da ferramenta é possível replicar informações de um banco de "origem" para um ou mais bancos de "destino". A replicação dos dados pode ser realizada de várias maneiras, isso é, podemos acioná-la manualmente, de tempos em tempos, ou ainda agendá-la através do próprio IBReplicator.

A ferramenta pode ser extremamente útil para empresas que possuem, por exemplo, filiais espalhadas em diferentes áreas geográficas, e que precisam manter, de forma on-line, ou não, suas bases de dados sincronizadas. No decorrer deste artigo, simularemos através de um exemplo completo, esse tipo de situação.

### Principais características

O IBReplicator é dividido em três aplicativos:

- *Replicator Manager*: Temos acesso a todas as funcionalidades da ferramenta. É nele onde fazemos a configuração e o gerenciamento dos bancos de dados que serão replicados;
- *Replicator Server*: Podemos acionar uma replicação de forma manual, ou de tempo em tempo, como, por exemplo, a cada 5 minutos;
- *Configure Services*: Podemos configurar o "Servidor de Replicação", ou o "Agendador de Replicação" do IBReplicator, como um serviço do *Windows*.

Entre as principais características do IBReplicator, encontram-se:

- *Velocidade de Replicação*: O processo de replicação ocorre diretamente entre os servidores de banco de dados, isso é, não existe nenhuma camada intermediária, como um banco de dados ou um driver, por exemplo;
- *Suporte a vários tipos de dados*: Suporta a replicação de todos os tipos de dados suportados pelo IB/FB, inclusive campos *Blobs* e *Arrays*;
- *Suporte a múltiplos bancos de dados de origem e/ou destino*: Podemos adicionar e configurar vários bancos de dados de origem e destino para replicação dos mesmos;
- *Replicação em várias direções*: O processo de replicação pode ser realizado em várias direções. Podemos realizar a replicação de forma unidirecional (Origem>Destino), bidirecional (Origem>Destino e Destino>Origem), ou ainda de forma n-direcional (Origem>Destino1, Destino1>Destino2 e Destino2>Origem);
- *Replication Monitor*: Podemos monitorar (em tempo real, ou não) através de gráficos, todo o processo de replicação. A ferramenta permite ainda informações estatísticas sobre todas as operações realizadas durante a replicação;
- *Replicação agendada*: Possibilita o agendamento da replicação, que pode ser realizada em diferentes intervalos de tempo, como dias, horas, determinados dias da semana etc.;
- *Suporte a diversas plataformas*: Apesar de ser for *Windows*, o IBReplicator possibilita a replicação de dados entre bancos de dados IB/FB localizados em diferentes sistemas operacionais, como *Windows*, *Linux*, *Solaris* ou *HP-UX*, por exemplo.

### Download e Instalação

Você pode baixar uma versão de avaliação através do link [www.ibphoenix.com](http://www.ibphoenix.com). Após o download, basta descompactar o arquivo ZIP e iniciar o processo de instalação. O IBReplicator é bastante



simples e rápido de ser instalado, bastando seguir as instruções disponíveis no assistente de instalação.

**Nota:** Para instalar todas as ferramentas e documentações do IBReplicator, basta selecionar a opção *Full installation of server, management tools and documentation* na janela *Select Components*. Terminada a instalação, você poderá acessar as ferramentas através do menu *Iniciar|Programas>IBReplicator* criado no *Windows*.

### Exemplo prático

Para demonstrar as funcionalidades do IBReplicator, simularemos a seguinte situação: suponhamos que nossa empresa precise replicar a cada hora seu cadastro de clientes e produtos da matriz para duas filiais localizadas em diferentes cidades do país (ou do mundo).

A cada hora todos os clientes e produtos adicionados, alterados ou excluídos serão replicados para os bancos de dados das filiais. Neste exemplo faremos a replicação de forma unidirecional, isso é, a partir de um banco de origem (matriz) realizaremos a replicação para dois bancos de destino (filiais).

**Nota:** Nada impede que você configure a replicação bidirecional, realizando também a replicação das filiais para o banco de dados da matriz.

### Definindo os bancos de dados

Para este artigo foi utilizando o InterBase 7.5 e a ferramenta IBExpert. Crie um novo banco de dados e adicione duas tabelas (*Clientes* e *Produtos*) de acordo com o script da **Listagem 1**. O script da listagem anterior pode ser executado diretamente no IBExpert através da opção *Script Executive* disponível no menu *Tools*. Caso queira, como alternativa, você pode utilizar o IBConsole. Após a criação do banco de dados da Matriz e suas tabelas, selecione o arquivo *Matriz.gdb* e faça duas cópias, chamadas *FiliaISP.gdb* e *FiliaRJ.gdb*.

#### Listagem 1. Script de criação do banco e tabelas

```
SET SQL DIALECT 3;

SET NAMES WIN1252;

CREATE DATABASE 'C:\Matriz.gdb'
USER 'SYSDBA' PASSWORD 'masterkey'
PAGE_SIZE 4096
DEFAULT CHARACTER SET WIN1252;

CREATE TABLE CLIENTES (
  CLI_ID INTEGER NOT NULL,
  CLI_NOME VARCHAR(40) NOT NULL,
  CLI_FONE VARCHAR(18),
  CLI_EMAIL VARCHAR(70),
  CONSTRAINT PK_CLIENTES
  PRIMARY KEY(CLI_ID));

CREATE TABLE PRODUTOS (
  PRO_ID INTEGER NOT NULL,
  PRO_DESCRICAO VARCHAR(60) NOT NULL,
  PRO_ABRVIAcao VARCHAR(20) NOT NULL,
  PRO_VALORCUSTO NUMERIC(12,2) NOT NULL,
  PRO_VALORVENDA NUMERIC(12,2) NOT NULL,
  CONSTRAINT PK_PRODUTOS
  PRIMARY KEY (PRO_ID));
```

### Configurando o Replication Manager

Após a criação e distribuição dos bancos de dados, o próximo passo é a configuração no IBReplicator para que o mesmo possa realizar o processo de replicação.

**Nota:** Neste exemplo a ferramenta IBReplicator está instalada na matriz, isso é, na mesma máquina onde reside o banco de dados *Matriz.gdb*.

O primeiro passo é criarmos um banco de dados de configuração, que armazenará as opções e parâmetros que serão utilizados no processo de replicação. Abra o aplicativo *Replication Manager* e a partir do menu *File* selecione a opção *New Configuration* para abrir a janela *Create a new Configuration Database*. Configure a janela de acordo com a **Figura 1**.

**Figura 1.** Definindo o banco de dados de configuração no Replication Manager

Nessa janela podemos criar o banco de configuração localmente ou remotamente, por exemplo, em um servidor *Linux* onde esteja rodando o servidor IB/FB. Clique no botão *Create* para criar a configuração.

Para utilizar a ferramenta, precisamos entrar com a chave de licença do produto. Como essa é uma versão de avaliação (*trial*), selecione a opção *Tools|Licence Manager* e através do botão *Add* entre com o valor "EVAL" para o campo *Licence ID*.

**Nota:** A licença de avaliação é válida por 14 dias a partir do momento que é efetuado o registro.

O próximo passo é registrar os bancos de dados que farão parte da replicação. A partir do menu *Database* do *Replication Manager* selecione a opção *Add* para adicionar um novo banco de dados na seção *Registered databases* localizada na aba *Databases*. Selecione o banco no *Treeview* e configure seus parâmetros localizados no lado direito da janela como mostrado a seguir:

- *Descriptive name:* "Matriz";
- *Server:* "localhost:C:\Matriz.gdb";
- *Administrative user name:* "sysdba";
- *Administrative password:* "masterkey";

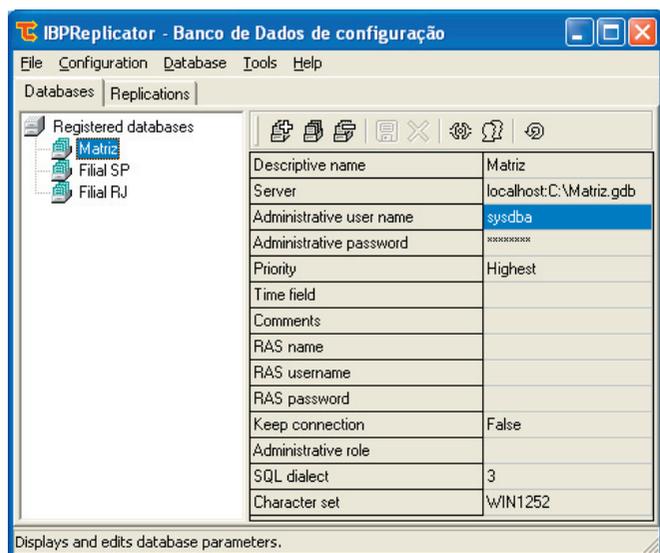
- *Character set:* "WIN1252"

Para os demais parâmetros mantenha seus valores default. Adicione mais dois bancos de dados e configure-os de forma semelhante a mostrada anteriormente. Para o parâmetro *Descriptive name* informe os valores "Filial SP" e "Filial RJ". Para salvar as alterações acesse o menu *Database|Save*, o botão *Save* localizado na barra de ferramentas no painel direito ou as teclas *Ctrl+S*.

No parâmetro *Server* informe a localização dos bancos de dados de cada filial, passando o IP público da máquina de cada filial seguido pelo caminho físico do banco de dados. Exemplo: 200.145.225.45: C:\FilialSP.gdb.

**Nota:** Você pode simular a replicação entre os três bancos de dados em uma mesma máquina. Para isso, basta informar o caminho de cada banco de dados utilizando "localhost" ou o IP local (127.0.0.1).

Os registros dos bancos de dados devem ficar semelhante aos da **Figura 2**. O IBReplicator utiliza o conceito de esquemas (*schema*) para realizar o processo de replicação. É no esquema que definimos quais tabelas e campos serão replicados entre os bancos registrados.



**Figura 2.** Registro dos bancos de dados utilizados no processo de replicação

### Schema

Para criar um esquema, utilizado na replicação dos dados da matriz para as filiais, selecione a aba *Replications* e clique no item *New schema*. Você pode criar um esquema também a partir do menu *Replication>Schema>New* disponível no menu principal do *Replication Manager*. Na janela *Schema properties* entre com o nome "Schema\_Matriz\_Filiais".

Para iniciar a configuração do esquema, selecione o "nó" *Source database* e clique no item *Add source database* para adicionar o banco de dados de origem da replicação. O banco de dados de origem deve ser único em cada esquema, isso é, a replicação dos

dados dentro de um esquema ocorre do banco de origem para um ou mais bancos de destino.

Na janela *Add source database* dentro da aba *Connection*, selecione o banco de dados *Matriz* para o campo *Registered database*. Para os campos *Replication username* e *Replication password* entre com os valores "sysdba" e "masterkey", respectivamente. Você pode testar a conexão com o banco através do botão *Test connection*. Através da aba *Settings* podemos definir algumas configurações a serem utilizadas pelo esquema. São elas:

- *Replication enabled:* Através dessa opção habilitamos ou não o processo de replicação. Essa opção é útil, por exemplo, para desabilitar a replicação enquanto a rede estiver em manutenção ou não disponível;
- *Conflict resolution strategy:* Quando uma chave primária de uma ou mais linhas de uma tabela do banco de dados de origem estiver em conflito com outra linha da tabela no banco de dados de destino, o IBReplicator provê três caminhos para tentar resolver o problema automaticamente:
  - *Priority-based:* Quando registramos um banco de dados dentro do *Replication Manager*, podemos configurar a prioridade (parâmetro *Priority*) a ser utilizada quando um conflito ocorrer, onde, o banco com maior prioridade tem preferência para resolução do conflito. Exemplo: caso o banco de destino possua uma prioridade maior que o de origem, a linha com conflito entre os bancos não é replicada, preservando assim a linha do banco de destino;
  - *Time-stamped:* Nessa configuração, a linha conflitante mais recente é preservada;
  - *Master-slave:* Nessa opção o banco de origem sempre substitui as linhas conflitantes do banco de destino.
- *Use default conflict resolution strategy:* Desmarcando essa opção, podemos definir nossa própria estratégia para resolver possíveis conflitos entre chaves primárias dos bancos replicados. Por padrão essa opção vem marcada, fazendo com que as configurações padrões do *Replication Manager* sejam utilizadas.

**Nota:** Você pode alterar as configurações padrão, as quais impactam em todos os esquemas criados, através do menu *Replication|Default settings*.

Para o nosso *schema*, utilizaremos a estratégia *Master-slave*, a qual já vem definida por default.

- *Separator character:* Essa opção é usada para separar os valores de chaves primárias com múltiplas colunas. Para essa opção mantenha o valor "5";
- *Comments:* Esse campo pode ser utilizado para documentação referente ao banco de dados que está sendo adicionado ao esquema.

Para a aba *Event logging*, ainda na janela *Add source database*, podemos especificar quais informações gostaríamos de visualizar e armazenar no arquivo de *log* do processo de replicação.

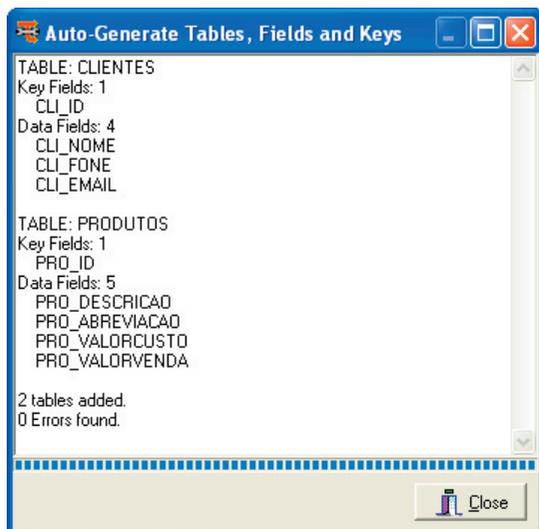
Marcando a opção *Use default logging* passamos a utilizar as opções definidas no *Replication Manager (Replication|Default settings>Event logging)*. Caso prefira, você pode selecionar quais informações deseja visualizar em tela e quais armazenar no arquivo de *log*. Clique no botão OK para fechar a janela e adicionar o banco de origem ao esquema.

### Adicionando os bancos

Para o "nó" *Target databases*, vamos adicionar os bancos de dados das duas filiais. Para isso, clique sobre a opção *Add target database* ou selecione a opção de menu *Replication|Target>Add*. Na janela *Add target database* dentro da aba *Settings* configure o campo *Registered database* apontando para o banco *Filial SP*.

Para as demais opções, configure-as de forma semelhante ao banco de origem (*Matriz*). Na aba *General* temos a opção de configurar o campo *Periodic commit*, onde podemos definir de quantos em quantos registros o IBReplicator fará a operação de *Commit*. Por padrão, o IBReplicator só realiza o *Commit* após toda a operação de replicação ter sido completada, para, dessa maneira garantir a integridade dos dados replicados.

Nessa opção, mantenha o valor "0", que já vem marcado como default. Clique no botão OK para adicionar o primeiro banco de destino a receber a replicação. O próximo passo é configurar as tabelas e colunas que farão parte da replicação. Como neste exemplo os bancos de dados envolvidos possuem a mesma estrutura de tabelas e colunas, podemos então utilizar a opção *Generate Tables, Keys and Fields* disponível quando selecionamos o "nó" *Replicated tables and procedures*. Veja a janela *Auto-Generate Tables, Fields and Keys* na **Figura 3**.



**Figura 3.** Gerando automaticamente o mapeamento das tabelas, colunas e chaves primárias

Caso o banco de dados que você esteja utilizando não contenha a mesma estrutura de tabelas e campos, você pode definir o mapeamento dos objetos de forma manual. Para configurar manualmente as tabelas a serem replicadas, utilize a opção *Replicated tables*. Para cada par de tabelas mapeadas é criado um novo "nó" abaixo do *Replicated tables and procedures*.

Para definir manualmente as colunas que fazem parte da chave primária de cada tabela, basta selecionar a opção *Define primary key* disponível para o "nó" *Key columns*. Selecionando o "nó" *Data columns*, podemos também definir manualmente as colunas que serão replicadas dentro de cada tabela. Para isso, utilize a opção *Define data columns*.

Terminado o mapeamento dos objetos entre os bancos de dados de origem (*Matriz*) e o de destino (*Filial SP*), precisamos agora configurar o segundo banco de destino, o da *Filial RJ*. Para adicionar o banco de destino *Filial RJ*, você pode repetir os passos anteriores ou utilizar a opção *Clone* através do menu de contexto do banco de destino *Filial SP* recém adicionado.

Clique na opção *Clone* para abrir a janela *Change target database*. Dentro dessa janela selecione o banco de dados registrado (*Filial RJ*) e clique no botão OK para realizar a clonagem das configurações.

**Nota:** Na janela *Change target database* são mostrados apenas os bancos de dados registrados dentro do *Replication Manager*. Caso você queira utilizar mais algum banco de dados no processo de replicação, você precisa, antes de tudo, registrá-lo dentro da aba *Databases* do *Replication Manager*.

Depois de efetuada a operação de *Clone*, todas as configurações e mapeamentos feitos no banco de dados de destino *Filial SP* são replicados para o *Filial RJ*. Para finalizar as configurações do esquema, selecione o banco de dados de origem (*Matriz*) e clique na opção *Create system objects* para que o IBReplicator gere os objetos necessários para realizar a replicação dos dados entre os bancos envolvidos no processo de replicação.

**Nota:** Essa opção pode levar algum tempo dependendo do número de tabelas e colunas que serão utilizadas na replicação. No final do processo de criação dos objetos, você receberá uma mensagem de informação. Clique em OK.

### Replicando o banco

Antes de testarmos a replicação entre os bancos de dados, vamos inserir alguns registros nas tabelas *Clientes* e *Produtos* do banco de origem *Matriz*. Você pode utilizar o script da **Listagem 2**, para inserir alguns registros em ambas as tabelas.

O IBReplicator permite que você especifique uma configuração a ser utilizada como default. Para isso, basta selecionar a opção *Configuration>Set as default* a partir do menu principal do *Replication Manager*.

**Nota:** Essa opção faz com que a configuração *Replicacao.gdb* seja utilizada como padrão nas demais ferramentas que constituem o IBReplicator.

Feche o *Replication Manager* e a partir do menu *Iniciar>Programas>IBReplicator* do Windows abra o aplicativo *Replication Server*. Através desse aplicativo podemos executar o processo de replicação de forma manual, ou configurar para que o mesmo seja executado a cada período de tempo.

**Listagem 2.** Script para inserção de dados nas tabelas do exemplo

```
INSERT INTO CLIENTES (CLI_ID, CLI_NOME, CLI_FONE,
  CLI_EMAIL)
VALUES (1, 'Everson Borges Volaco', '41 23325455',
  'everson@rhealeza.com.br');
INSERT INTO CLIENTES (CLI_ID, CLI_NOME, CLI_FONE,
  CLI_EMAIL)
VALUES (2, 'Natalia da Silva', '41 30234576',
  'natalia.volaco@gmail.com');
INSERT INTO CLIENTES (CLI_ID, CLI_NOME, CLI_FONE,
  CLI_EMAIL)
VALUES (3, 'Sandra Mara Martins', '42 32322768',
  'Sandra.Borges@yahoo.com.br');
SET NAMES WIN1252;
INSERT INTO PRODUTOS (PRO_ID, PRO_DESCRICA0,
  PRO_ABBREVIACA0, PRO_VALORCUSTO, PRO_VALORVENDA)
VALUES (1, 'Delphi Architect 2005',
  'Delphi 2005', 2500.50, 3000.68);
INSERT INTO PRODUTOS (PRO_ID, PRO_DESCRICA0,
  PRO_ABBREVIACA0, PRO_VALORCUSTO, PRO_VALORVENDA)
VALUES (2, 'JBuilder Enterprise 2005',
  'JBuilder 2005', 3500.50, 4600.89);
INSERT INTO PRODUTOS (PRO_ID, PRO_DESCRICA0,
  PRO_ABBREVIACA0, PRO_VALORCUSTO, PRO_VALORVENDA)
VALUES (3, 'StarTeam Enterprise Advanced 2005',
  'StarTeam 2005', 8500.50, 11060.45);
```

Como definimos a configuração *Replicacao.gdb* como default, a mesma já vem carregada ao abrir o *Replication Server*. Para executar manualmente a replicação, selecione o botão *Replicate* disponível na barra de botões da janela. Caso você deseje realizar a replicação automaticamente de tempos em tempos, basta entrar com o tempo desejado, em segundos, no campo *Timer Interval* e clicar sobre o botão *Replication Timer*.

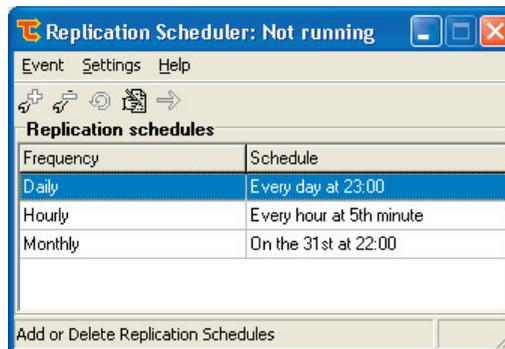
Todas as informações referentes a cada replicação processada são mostradas e armazenadas no arquivo de *log* configurado na aba *Events Logging* no *Replication Manager*. Para confirmar que a replicação dos registros da Matriz foi realizada para os dois bancos, basta registrar os bancos de destino no IBExpert e visualizar os registros das tabelas *Clientes* e *Produtos*.

Você pode também configurar e executar a replicação através do próprio *Replication Manager*. Para isso, abra o aplicativo e a partir do menu *File>Open Configuration* selecione o banco de configuração *Replicacao.gdb*. Para executar de imediato a replicação selecione o menu *Tools>Notify server>Replicate now*. Você pode ainda, se preferir, agendar a replicação, para que a mesma ocorra no dia, hora, ou dias da semana que você quiser.

Selecione a opção de menu *Tools>Scheduler* para abrir a janela *Replication Scheduler*. Através dessa janela, você pode adicionar quantos agendamentos quiser para a replicação. Através do menu *Event*, você pode adicionar, apagar ou carregar um agendamento.

No menu *Settings>Log File Settings* podemos apontar e habilitar

um arquivo de *log* a ser gerado cada vez que uma replicação agendada for executada. Veja na **Figura 4** alguns exemplos de agendamento.



**Figura 4.** Agendando para a replicação

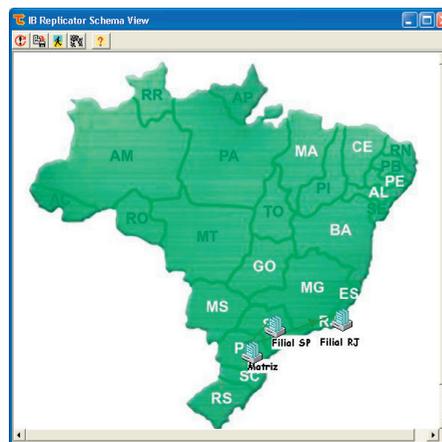
**Serviço do Windos**

O IBReplicator permite que você instale-o como um serviço do Windows tanto o *Replication Server* como o *Replication Scheduler*. Para isso, basta acessar o aplicativo *Configure Services* a partir do menu *Iniciar>Programas>IBReplicator* do Windows. Dentro da janela *Replication Service Control* você pode instalar uma ou ambas as ferramentas como serviço do Windows, além de poder configurá-las para inicializar automaticamente, ou ainda dependendo do serviço do InterBase ou Firebird.

**Schema View**

O IBReplicator possibilita ainda que, para cada esquema criado, possamos representá-lo através de uma imagem, isso é, podemos demonstrar através do *schema view* os bancos de dados relacionados à replicação, suas localizações físicas e relacionamentos.

Para acessar o *Schema view*, basta selecionar a opção de menu *Tools>Schema view* disponível no menu principal do *Replication Manager*. Através da janela *IB Replicator Schema View*, podemos selecionar uma imagem a ser utilizada para representar nosso esquema. Veja um exemplo de representação na **Figura 5**.



**Figura 5.** Schema View demonstrando a distribuição geográfica dos bancos utilizados na replicação



## Utilizando o Replication Monitor

Durante cada processo de replicação, podemos, através do *Replication Monitor*, ferramenta que acompanha o IBReplicator, monitorar todos as operações realizadas. Através do *Replication Monitor*, podemos visualizar graficamente diversas informações sobre o processo de replicação.

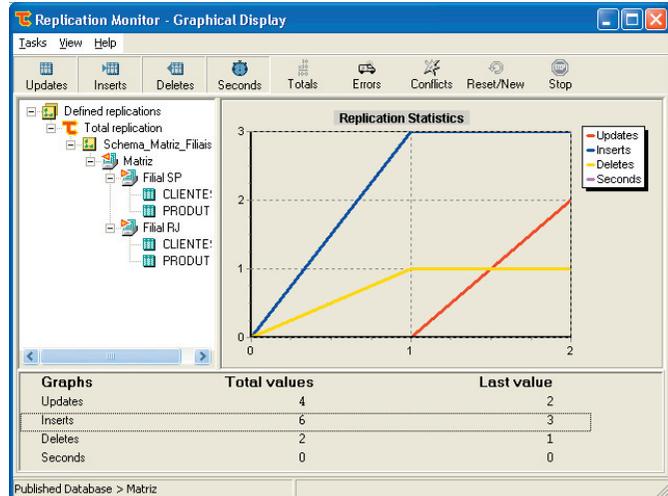
Selecione a opção *Tools>Monitor* a partir do menu principal do *Replication Manager*. Na janela *Replication Monitor – Graphical Display* podemos através do menu *Tasks* selecionar as operações que queremos monitorar.

Entre elas, encontram-se, operações de *Insert, Update, Delete*, tempo da operação, número de erros, conflitos, entre outros. Através da opção de menu *View>Error/Conflict Notification* podemos configurar a ferramenta para avisar quando algum conflito ocorrer. Na janela *Replication – Error/Conflict notification* podemos configurar dois caminhos para aviso sobre conflitos durante o processo de replicação: Caixa de mensagem e/ou envio de e-mail. Veja a janela configurada na **Figura 6**.

Para testar o funcionamento do *Replication Monitor*, selecione as operações que deseja visualizar no gráfico e clique na opção *Start* para iniciar o monitoramento. Quando o monitoramento está ativo, é habilitada a opção de menu *View|Stats*, a qual permite a visualização das informações estatísticas referente à replicação. Veja na **Figura 7** o *Replication Monitor* em execução.



**Figura 6.** Configurando as opções para recebimento de notificações



**Figura 7.** Monitorando as operações realizadas durante o processo de replicação

## Conclusões

Hoje, vemos em muitas empresas aplicativos desenvolvidos para fazerem a replicação de dados entre bancos de dados. Muitos desses aplicativos utilizam arquivos textos (colunados ou CSV), ou XML transferidos através de FTP ou mesmo e-mail.

Vimos neste artigo que, para usuários dos bancos IB/FB, o IBReplicator é uma ótima opção de ferramenta para replicação de dados entre dois ou mais bancos de dados.

Um abraço e até a próxima.

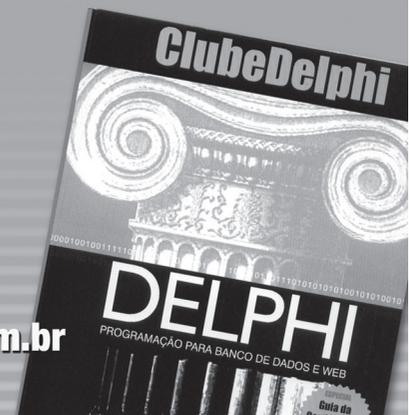


**Everson Borges Volaco** (everson@rhealeza.com.br) é desenvolvedor e instrutor certificado Borland, com experiência em aplicações cliente/servidor, usando Delphi, Interbase e Oracle. Possui três certificações oficiais Borland: Borland Delphi 7.0, Borland CaliberRM 6.0 e Borland StarTeam 6.0.

# Programação para Banco de Dados e Web de Guinther Pauli

Com exemplos práticos, o livro cobre em detalhes as mais importantes tecnologias de programação para Banco de Dados e Web, incluindo DBExpress, Acesso ao IB/FB, Multicamadas, SOAP, COM+, Migração de BDE para DBExpress, ClientDataSet, XML, Web Services, IntraWeb, Delphi for .NET, ASP.NET, BDP, VCL.NET e ADO.NET, além de tópicos avançados como criação de componentes, depuração, programação multithread e segredos do DBGrid. E mais um capítulo especial sobre a certificação Borland Delphi.

Compre agora o seu através dos sites [www.submarino.com.br](http://www.submarino.com.br)  
[www.saraiva.com](http://www.saraiva.com)





# POO na prática

Isaque Pinheiro

## Criando classes, regras de negócio e aplicações em camadas

A grande dúvida da maioria dos programadores não é somente o que significa Programação Orientada a Objetos e sim como implantá-la em projetos reais. Materiais, teorias e conceitos sobre o assunto são o que mais encontramos na internet.

Neste artigo mostrarei um exemplo prático (para todos aqueles que ainda se perguntam como funciona POO) de como programar realmente Orientado a Objetos no Delphi. POO permite o desenvolvimento de classes e regras de negócios que sejam utilizadas em todo projeto. Usando essa técnica não temos como escapar do desenvolvimento em camadas, que dará ao programa, mais dinâmica e menos manutenção.

Quando falamos em desenvolvimento em camadas, estamos nos referindo na divisão do programa em várias partes (partes essas que terão que ser bem definidas para serem utilizadas em outras "camadas" do projeto).

Quando pensamos em POO, temos que lembrar de alguns pontos fundamentais: Abstração, Polimorfismo, Herança e Encapsulamento, esses explicados em detalhes na edição 58, no excelente artigo de Adail Muniz.

Neste artigo, vamos desenvolver um cadastro de clientes totalmente POO, dando assim uma total realidade aos desenvolvedores de software. O projeto será dividido em camadas, que são: acesso a dados, regras de negócio e interface com o usuário.

### Camada de acesso a dados

Essa camada tratará do acesso ao banco de dados, onde usaremos um Data Module juntamente com um *ClientDataSet* para acessar os dados. No Delphi 2005 crie uma nova aplicação VCL, dê o nome ao formulário de "FrmInterface" e salve-o como "Interface\_OOP.pas". Salve o projeto como "AP\_OOP.dpr".

**Nota:** Este projeto também pode ser desenvolvido em Delphi 7, caso ainda esteja utilizando essa versão da ferramenta.

Adicione um DataModule ao projeto, dando a ele o nome de "dtmDados" e salvando-o na unit "Dados\_OOP.pas". Selecione agora a categoria *Data Access*, adicione um *ClientDataSet* no DataModule e mude seu nome para "Clientes". Agora vamos carregar no *ClienteDataSet* um arquivo XML no evento *OnCreate* do DataModule (usei neste exemplo um arquivo XML para facilitar o acesso, mas pode ser usado qualquer banco de dados como Firebird, SQL Server ou

Oracle). Use o seguinte código:

```
Clientes.FileName := 'Clientes.xml';
```



**Nota:** Você pode baixar esse arquivo XML no endereço para download deste artigo.

Vamos criar um método na seção *public*, para devolver para a camada de regras de negócio o *ClientDataSet* que está acessando a nossa tabela, conforme a **Listagem 1**.

### Listagem 1. Função para retornar o ClientDataSet

```
public
function Cliente_Open: TDataSet;
...
function TDtmDados.Cliente_Open: TDataSet;
begin
Result := Clientes;
end;
```

Criaremos agora duas classes para essa camada, que tratarão desde operações básicas até a validação dos dados digitados na camada de interface.

### Classe de dados

Está dividida em duas:

*Classe de campos:* tratará todos os tipos de campos, como *Data*, *Numeric*, *Integer*, *String* etc., suas características e validações. Crie uma nova unit e salve-a como "Field\_OOP.pas". Crie uma classe chamada "TFields\_OOP" como na **Listagem 2**.

Aqui estamos conhecendo uma *Abstração* (objeto que possui características como propriedade e métodos) e *Encapsulamento* (criar métodos e propriedades para o objeto, ficando ocultos para quem vai usá-los).

Temos a propriedade *AsValue* que guardará o valor do campo e *Required*, que indica se o campo é obrigatório ou não (podemos criar várias propriedades conforme nossa necessidade). Temos uma classe para ser usada em todos nossos projetos.

Ao desenvolver a aplicação final, que usará essa classe, a aplicação terá todo o tratamento de campos digitados pelo usuário. Assim o desenvolvedor pode se preocupar mais com a interface do que com validações de dados.



## Listagem 2. Classe para tratamento de campos

```
unit Field_00P;

interface

uses

Classes;

type
  TFields_00P = class(TObject)
  public
    constructor Create;
    destructor Destroy; override;
    function ValidateCurrency(Value: string): Boolean;
    function ValidateDate(Value: string): Boolean;
    property AsCurrency: Currency read FAsCurrency
      write SetAsCurrency;
    property AsFloat: Double read FAsFloat
      write SetAsFloat;
    property AsInteger: LongInt read FAsInteger
      write SetAsInteger;
    property AsValue: string read FAsValue
      write SetAsValue;
    property AsDate: TDateTime read FAsDate
      write SetAsDate;
    property DisplayFormat: string read FDisplayFormat
      write SetDisplayFormat;
    property DisplayLabel: string read FDisplayLabel
      write SetDisplayLabel;
    property EditMask: string read FEditMask
      write SetEditMask;
    property Required: boolean read FRequired
      write SetRequired;
    property List: TStringList read FList
      write SetList;
      { Aperte Shift+Ctrl+C para gerar a implementação }
  end;

implementation

uses SysUtils;

constructor TFields_00P.Create;
begin
  { Instancia o objeto, que receberá lista para ser
  usada na interface }
  FList := TStringList.Create;
  FAsValue := '0';
end;

destructor TFields_00P.Destroy;
begin
  { Libera o objeto da memória e limpa o ponteiro }
  FreeAndNil(FList);
end;

function TFields_00P.ValidateCurrency(
  Value: string): Boolean;
var
  curValue: Currency;
begin
  Result := False;
  try
    if Value <> '' then
      begin
        curValue := StrToCurr(Value);
        Result := True;
      end;
    except

```

```
    on E: EConvertError do
      raise Exception.Create(
        'Valor monetário inválido!');
    end;
  end;

function TFields_00P.ValidateDate(
  Value: string): Boolean;
var
  datValue: Currency;
begin
  Result := False;
  try
    if Value <> ' / / ' then
      begin
        datValue := StrToDate(Value);
        Result := True;
      end;
    except
      on E: EConvertError do
        raise Exception.Create('Data inválida!');
      end;
    end;
  end;

procedure TFields_00P.SetAsCurrency(Value: Currency);
begin
  try
    if AsCurrency <> Value then
      FAsCurrency := Value;
      { Repassa o conteúdo para a propriedade AsValue }
      FAsValue := CurrToStr(Value);
    except
    end;
  end;

procedure TFields_00P.SetAsDate(const Value: TDateTime);
begin
  try
    if FAsDate <> Value then
      FAsDate := Value;
      { Repassa o conteúdo para a propriedade AsValue }
      FAsValue := DateToStr(Value);
    except
    end;
  end;

procedure TFields_00P.SetAsFloat(Value: Double);
begin
  try
    if FAsFloat <> Value then
      FAsFloat := Value;
      { Repassa o conteúdo para a propriedade AsValue }
      FAsValue := FloatToStr(Value);
    except
    end;
  end;

procedure TFields_00P.SetAsInteger(Value: LongInt);
begin
  try
    if AsInteger <> Value then
      FAsInteger := Value;
      { Repassa o conteúdo para a propriedade AsValue }
      FAsValue := IntToStr(Value);
    except
    end;
  end;
end.
```



*Classe de registros:* terá controle dos registros como os métodos *Next, Prior, Post, Edit* etc. Crie outra unit e salve-a como "DataSet\_OOP.pas". Nessa unit crie uma classe chamada de "TDataSet\_OOP" como na **Listagem 3**.

### Listagem 3. Classe de controle de registros

```
unit DataSet_OOP;

interface

uses
  DBClient;

type
  { Classe TDataSet_OOP, implementação de metodos e
  eventos na classe }
  TDataSet_OOP = class(TObject)
  private
    FDataSet: TClientDataSet;
  public
    constructor Create;
    destructor Destroy; override;
    function Active: Boolean;
    function Apply: Boolean; virtual;
    function Cancel: Boolean; virtual;
    function Close: Boolean; virtual;
    function Delete: Boolean; virtual;
    function Edit: Boolean; virtual;
    function Insert: Boolean; virtual;
    function Next: Boolean; virtual;
    function Open: Boolean; virtual;
    function Post: Boolean; virtual;
    function Prior: Boolean; virtual;
    function First: Boolean; virtual;
    function Last: Boolean; virtual;
    function Bof: Boolean; virtual;
    function Eof: Boolean; virtual;
    property DataSet: TClientDataSet
      read FDataSet write FDataSet;
  end;

implementation

constructor TDataSet_OOP.Create;
begin
  //
end;

destructor TDataSet_OOP.Destroy;
begin
  //
end;

function TDataSet_OOP.Apply: Boolean;
begin
  Result := True;
  try
    if (DataSet <> nil) and (DataSet.Active) and
      (DataSet.ChangeCount > 0) then
      DataSet.ApplyUpdates(0);
  except
    Result := False;
  end;
end;
```

```
end;

function TDataSet_OOP.Cancel: Boolean;
begin
  Result := True;
  try
    if (DataSet <> nil) and (DataSet.Active) then
      DataSet.Cancel;
  except
    Result := False;
  end;
end;

function TDataSet_OOP.Close: Boolean;
begin
  Result := True;
  try
    if (DataSet <> nil) and (DataSet.Active) then
      DataSet.Close;
      DataSet := nil;
  except
    Result := False;
  end;
end;

function TDataSet_OOP.Delete: Boolean;
begin
  Result := False;
  try
    if (DataSet <> nil) and (DataSet.Active) then
      begin
        if (DataSet.RecordCount > 0) then
          begin
            DataSet.Delete;
            Result := True;
          end;
        end;
      except
        Result := False;
      end;
  end;

function TDataSet_OOP.Edit: Boolean;
begin
  Result := True;
  try
    if (DataSet <> nil) and (DataSet.Active) then
      DataSet.Edit;
  except
    Result := False;
  end;
end;

function TDataSet_OOP.Insert: Boolean;
begin
  Result := True;
  try
    if (DataSet <> nil) and (DataSet.Active) then
      DataSet.Insert;
  except
    Result := False;
  end;
end;

function TDataSet_OOP.Next: Boolean;
begin
  Result := True;
  try
    if (DataSet <> nil) and (DataSet.Active) then
```





```
        DataSet.Next;
    except
        Result := False;
    end;
end;

function TDataSet_OOP.Open: Boolean;
begin
    Result := True;
    try
        if (DataSet <> nil) and not (DataSet.Active) then
            DataSet.Open;
        except
            Result := False;
        end;
    end;
end;

function TDataSet_OOP.Post: Boolean;
begin
    Result := True;
    if (DataSet <> nil) and (DataSet.Active) then
        DataSet.Post;
    end;
end;

function TDataSet_OOP.Prior: Boolean;
begin
    Result := True;
    try
        if (DataSet <> nil) and (DataSet.Active) then
            DataSet.Prior;
        except
            Result := False;
        end;
    end;
end;

function TDataSet_OOP.Last: Boolean;
begin
    Result := True;
    try
        if (DataSet <> nil) and (DataSet.Active) then
            DataSet.Last;
        except
            Result := False;
        end;
    end;
end;

function TDataSet_OOP.First: Boolean;
begin
    Result := True;
    try
        if (DataSet <> nil) and (DataSet.Active) then
            DataSet.First;
        except
            Result := False;
        end;
    end;
end;

function TDataSet_OOP.Bof: Boolean;
begin
    Result := True;
    try
        if (DataSet <> nil) and (DataSet.Active) then
            Result := DataSet.Bof;
        except
            Result := False;
        end;
    end;
end;

function TDataSet_OOP.Eof: Boolean;
```

```
begin
    Result := True;
    try
        if (DataSet <> nil) and (DataSet.Active) then
            Result := DataSet.Eof;
        except
            Result := False;
        end;
    end;
end;

function TDataSet_OOP.Active: Boolean;
begin
    Result := True;
    try
        if (DataSet <> nil) and (DataSet.Active) then
            Result := DataSet.Active;
        except
            Result := False;
        end;
    end;
end;
end.
```

Essa classe será desenvolvida utilizando *herança*, de forma que outras classes possam herdá-la, criando uma hierarquia. Temos a propriedade *DataSet* que receberá o *ClientDataSet* da camada de acesso a dados e os métodos *Open* e *Post* que executarão os respectivos métodos do *ClientDataSet*.

Essa classe terá propriedades e métodos que poderão ser herdados e sobrescritos por outra classe a ser implementada, para isso, temos que definir seus métodos com a diretiva *virtual*, como visto na **Listagem 3**. Veremos na classe da camada de negócio (que será nossa próxima classe a ser criada), como herdar essas classes.

### Camada de regras de negócio

Na camada de regras de negócio temos as informações necessárias para uma boa programação POO. É essa camada que se comunica com a aplicação final (camada de interface do usuário), implementando assim suas particularidades. Além disso, se tivermos um Cadastro de Clientes, um Cadastro de Vendedores, um Cadastro de Fornecedores, cada um terá sua própria classe de regra de negócio.

Crie uma nova unit ("Cliente\_OOP.pas"), declare uma classe chamada "TCliente" do tipo "TDataSet\_OOP" e algumas variáveis, métodos e propriedades como na **Listagem 4**. Nessa classe vamos utilizar a classe de campos (*TFields\_OOP*), atribuindo a cada uma delas suas características, como *EditMask*, *Required*, *DisplayLabel*, *List* e definindo seus valores *default*. Também vamos usar a classe de registros (*TDataSet\_OOP*). Além de herdá-la, vamos implementar seus métodos e criar novos métodos para a classe filha. É importante frisar que é nessa camada que iremos trocar dados entre a tabela e a interface.

### Listagem 4. Classe com as regras de negócio

```
unit Cliente_OOP;

interface

uses SysUtils, Dados_OOP, DataSet_OOP, Field_OOP;

type
```



```
TCliente = class(TDataSet_OOP)
private
...
public
  constructor Create;
  destructor Destroy; override;
  function Cancel: Boolean; override;
  function Close: Boolean; override;
  function Delete: Boolean; override;
  function Insert: Boolean; override;
  function Next: Boolean; override;
  function Open: Boolean; override;
  function Post: Boolean; override;
  function Prior: Boolean; override;
  function First: Boolean; override;
  function Last: Boolean; override;

  function Locate(
    Fields, Value: string): Boolean;
  function Index(Fields: string): Boolean;

  property Codigo: TFields_OOP read FCodigo
    write FCodigo;
  property Nome: TFields_OOP read FNome write FNome;
  property Endereco: TFields_OOP read FEndereco
    write FEndereco;
  property Cidade: TFields_OOP read FCidade
    write FCidade;
  property Bairro: TFields_OOP read FBairro
    write FBairro;
  property Cep: TFields_OOP read FCep write FCep;
  property Telefone: TFields_OOP read FTelefone
    write FTelefone;
  property Sexo: TFields_OOP read FSexo write FSexo;
  property Credito_Limite: TFields_OOP
    read FCredito_Limite write FCredito_Limite;
  property DataCadastro: TFields_OOP
    read FDataCadastro write FDataCadastro;
end;

implementation

uses DB, DBClient;

constructor TCliente.Create;
begin
  { DataSet recebe o objeto que acessa a tabela no
  banco de dados }
  DataSet := TClientDataSet(DtmDados.Cliente_Open);
  FCodigo := TFields_OOP.Create;
  FCodigo.DisplayLabel := 'Código';
  FNome := TFields_OOP.Create;
  FNome.DisplayLabel := 'Nome';
  FEndereco := TFields_OOP.Create;
  FEndereco.DisplayLabel := 'Endereço';
  FCidade := TFields_OOP.Create;
  FCidade.DisplayLabel := 'Cidade';
  FBairro := TFields_OOP.Create;
  FBairro.DisplayLabel := 'Bairro';
  FCep := TFields_OOP.Create;
  FCep.DisplayLabel := 'Cep';
  FTelefone := TFields_OOP.Create;
  FTelefone.DisplayLabel := 'Telefone';
  FSexo := TFields_OOP.Create;
  FSexo.DisplayLabel := 'Sexo';
  FSexo.List.Add('Masculino');
  FSexo.List.Add('Feminino');
  FCredito_Limite := TFields_OOP.Create;
  FCredito_Limite.DisplayLabel := 'Limite Crédito';
```

```
  FCredito_Limite.Required := True;
  FDataCadastro := TFields_OOP.Create;
  FDataCadastro.DisplayLabel := 'Data Cadastro';
  FDataCadastro.Required := True;
  { Inicia as propriedades de nossa classe,
  com os valores padrões }
  Set_Defaults;
end;

destructor TCliente.Destroy;
begin
  { Destroy as instancias da classe usadas nos
  campos do cliente }
  FCodigo.Free;
  FNome.Free;
  FEndereco.Free;
  FBairro.Free;
  FCidade.Free;
  FCep.Free;
  FTelefone.Free;
  FSexo.Free;
  FCredito_Limite.Free;
  FDataCadastro.Free;
end;

function TCliente.Cancel: Boolean;
begin
  inherited Cancel;
  { Posiciona o ponteiro no primeiro registro }
  First;
  { Limpa as propriedades atribuindo conteúdo do
  registro posicionado }
  Get_Values;
end;

function TCliente.Close: Boolean;
begin
  inherited Close;
  { Limpa as propriedades atribuindo conteúdo padrão }
  Set_Empty;
end;

function TCliente.Delete: Boolean;
begin
  inherited Delete;
  { Dispara o metodo da classe de dados }
  if (DataSet.RecordCount > 0) then
  begin
    { Limpa as propriedades atribuindo conteúdo padrão }
    Get_Values;
  end
  else
  begin
    { Limpa as propriedades atribuindo conteúdo padrão }
    Set_Defaults;
  end;
end;

function TCliente.Insert: Boolean;
begin
  inherited Insert;
  { Limpa as propriedades atribuindo conteúdo padrão }
  Set_Defaults;
end;

function TCliente.Next: Boolean;
begin
  inherited Next;
  { Atribui os conteúdos dos campos as propriedades }
```





```
    Get_Values;
end;

function TCliente.Open: Boolean;
begin
    inherited Open;
    { Atribui os conteúdos dos campos as propriedades }
    Get_Values;
end;

function TCliente.Post: Boolean;
begin
    { Atribue as propriedades da classe cliente,
    para os campos da base de dados }
    Set_Values;
    { Verificação se o campo foi definido como obrigatório }
    if (Credito_Limite.Required) and
        (Credito_Limite.AsValue = '') then
        raise Exception.Create(
            'Valor de crédito, não pode ser nulo!');
    { Verificação se o campo foi definido como obrigatório }
    if (DataCadastro.Required) and
        (DataCadastro.AsValue = ' / / ') then
        raise Exception.Create(
            'Data de cadastro é obrigatória');
    inherited Post;
end;

function TCliente.Prior: Boolean;
begin
    inherited Prior;
    { Atribui os conteúdos dos campos as propriedades }
    Get_Values;
end;

function TCliente.Last: Boolean;
begin
    inherited Last;
    { Atribui os conteúdos dos campos as propriedades }
    Get_Values;
end;

function TCliente.First: Boolean;
begin
    inherited First;
    { Atribui os conteúdos dos campos as propriedades }
    Get_Values;
end;

procedure TCliente.Set_Defaults;
begin
    { Aqui atribuímos os valores padrões das
    propriedades de nossa classe }
    FCodigo.AsInteger := 0;
    FNome.AsValue := '';
    FEndereco.AsValue := '';
    FBairro.AsValue := '';
    FCidade.AsValue := '';
    FCep.AsValue := '';
    FTelefone.AsValue := '';
    FSexo.AsValue := 'Masculino';
    FCredito_Limite.AsCurrency := 0;
    FDataCadastro.AsDate := Date;
end;

procedure TCliente.Set_Empty;
begin
```

```
    { Aqui atribuímos os valores padrões das
    propriedades de nossa classe }
    FCodigo.AsValue := '';
    FNome.AsValue := '';
    FEndereco.AsValue := '';
    FBairro.AsValue := '';
    FCidade.AsValue := '';
    FCep.AsValue := '';
    FTelefone.AsValue := '';
    FSexo.AsValue := '';
    FCredito_Limite.AsValue := '';
    FDataCadastro.AsValue := '';
end;

procedure TCliente.Get_Values;
begin
    { Aqui carregamos as propriedades da nossa classe,
    com os valores dos campos que retornaram da
    camada de dados }
    FCodigo.AsInteger :=
        DataSet.FieldByName('CODIGO').AsInteger;
    FNome.AsValue :=
        DataSet.FieldByName('NOME').AsString;
    FEndereco.AsValue :=
        DataSet.FieldByName('ENDERECO').AsString;
    FBairro.AsValue :=
        DataSet.FieldByName('BAIRRO').AsString;
    FCidade.AsValue :=
        DataSet.FieldByName('CIDADE').AsString;
    FCep.AsValue :=
        DataSet.FieldByName('CEP').AsString;
    FTelefone.AsValue :=
        DataSet.FieldByName('TELEFONE').AsString;
    FSexo.AsValue :=
        DataSet.FieldByName('SEXO').AsString;
    FCredito_Limite.AsCurrency :=
        DataSet.FieldByName('CREDITO_LIMITE').AsCurrency;
    FDataCadastro.AsDate :=
        DataSet.FieldByName('DATACADASTRO').AsDateTime;
end;

procedure TCliente.Set_Values;
begin
    { Aqui carregamos os campos, com os valores dos
    Das propriedades da classe }
    DataSet.FieldByName('CODIGO').AsInteger :=
        FCodigo.AsInteger;
    DataSet.FieldByName('NOME').AsString :=
        FNome.AsValue;
    DataSet.FieldByName('ENDERECO').AsString :=
        FEndereco.AsValue;
    DataSet.FieldByName('BAIRRO').AsString :=
        FBairro.AsValue;
    DataSet.FieldByName('CIDADE').AsString :=
        FCidade.AsValue;
    DataSet.FieldByName('CEP').AsString :=
        FCep.AsValue;
    DataSet.FieldByName('TELEFONE').AsString :=
        FTelefone.AsValue;
    DataSet.FieldByName('SEXO').AsString :=
        FSexo.AsValue;
    DataSet.FieldByName('CREDITO_LIMITE').AsCurrency :=
        FCredito_Limite.AsCurrency;
    DataSet.FieldByName('DATACADASTRO').AsDateTime :=
        FDataCadastro.AsDate;
end;

function TCliente.Locate(Fields, Value: string): Boolean;
begin
```



```

[ Método de localização de registros ]
DataSet.Locate(Fields, Value, [loPartialKey,
    loCaseInsensitive]);
[ Atribui os conteúdos dos campos as propriedades ]
Get_Values;
end;

function TCliente.Index(Fields: string): Boolean;
begin
    [ Ordena os dados em memória ]
    DataSet.IndexFieldNames := Fields;
end;
end.

```

Essa classe é para o desenvolvedor de interface uma “caixa preta”, ele só irá desfrutar dos recursos que essa classe oferece, sem saber como foram desenvolvidos. Com isso, estamos tirando a sobrecarga do desenvolvedor para conhecer toda a estrutura de base de dados, bem como cada validação a ser feita para todos os campos, entre outras características.

Poderíamos colocar nessa camada validações de CPF, cálculos de data de aniversário, verificação de limite de compra do cliente, entre vários outros recursos, como regras de negócio. Essa camada terá tudo que a camada de interface precisar e você não quer que o seu desenvolvedor final saiba como foi concebido.

Vamos imaginar que temos que contratar um programador, mas esse não sabe como fazer cálculos de juros, mora, como achar idade de uma pessoa, como validar CPF, como verificar se a data é válida, como fazer um *Select* na base de dados etc. (nossa, convenhamos, isso não está parecendo um programador!). Mas, sabe fazer interface (desenhar tela), ou é um programador com menor experiência na equipe ou menos responsabilidades de negócio, então você cria nessa camada métodos que o programador só iria disparar da camada interface, para fazer o que citamos anteriormente.

Observe que quando criamos uma classe herdando de outra classe, essa passa a ter a mesma característica da classe pai, suas propriedades e métodos. Esses métodos podem ser implementados, desde que no final (na classe pai) tenha a palavra *virtual* e na classe filha tenha a palavra *override*. Com isso, estamos fazendo uso do *polimorfismo*, outra importante técnica de POO.

### Classe de persistência

Será responsável por trocar dados entre a camada de dados e a camada de interface. Para isso, nossa classe foi implementada com alguns métodos, vamos conhecê-los:

- **Get\_Values:** passará o conteúdo de cada campo da tabela, para cada propriedade da classe de persistência;
- **Set\_Defaults:** preencherá as propriedades da classe de persistência com seus valores padrões;
- **Set\_Values:** devolverá os valores que estão nas propriedades da classe de persistência para os campos da tabela;
- **Set\_Empty:** limpará o conteúdo das propriedades da classe de persistência, para quando for chamado o método **Insert**, os campos sejam limpos;
- **Locate:** localizará um registro dentro da tabela e depois

preenche as propriedades da classe de persistência com os valores, para isso esse método, depois de localizar, dispara o método *Get\_Values*;

- **Index:** indexará a tabela pelos campos que forem passados como parâmetro.

No *Create* da classe, vamos disparar o método *Open* da camada de acesso a dados. Esse método devolverá para a propriedade *DataSet* da classe o *ClientDataSet* que está acessando a tabela. Ainda no *Create* definiremos como os campos irão se comportar na camada de interface com o usuário, como *Required* (obrigatório ou não), *DisplayLabel* etc.

**Nota:** No método *Post* são realizadas validações dos dados enviados para essa classe, se houver algum inválido, será disparada uma exceção (com *raise*), não deixando as informações serem salvas na tabela. Essa é a maneira correta de disparar mensagens de erro para a interface de usuário a partir de classes de negócio. Sempre use *raise* e deixe o tratamento e apresentação do erro por conta da interface, jamais use algo como *ShowMessage* nesse caso.

### Camada de interface

Essa é a camada final do projeto, que usufruirá de toda a estrutura criada nas camadas anteriores. Acesse o formulário principal e adicione no *uses* a unit *Cliente\_OOP*. Após, adicione no formulário alguns componentes do tipo *Label*, *Edit*, *ComboBox*, *MaskEdit*, *BitBtn* e *Button*. Tome por base a **Figura 1** para disponibilizar os componentes no formulário. Configure o *Name* dos componentes para “edtNome”, “edtEndereco” etc., conforme o campo que representam. Observe que não estamos utilizando controles *Data-Aware*, o que vai otimizar nossa solução e deixá-la mais abstraída e flexível.

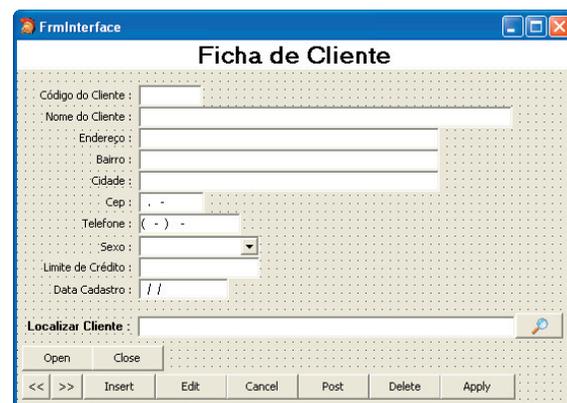


Figura 1. Interface da aplicação

**Nota:** Nos fontes que estarão disponíveis para download temos a aplicação completa, por esse motivo não descreveremos todas as propriedades de cada componente, visto que esse é um procedimento simples de ser realizado.

O próximo passo é acessarmos as classes criadas. Primeiro, declare uma variável do tipo *TCliente* (classe da camada de regra de negócio)

e no evento *OnCreate* do formulário vamos instanciá-la, conforme o código a seguir:

```
private
  oopCliente: TCliente;
...
procedure TFrmInterface.FormCreate(Sender: TObject);
begin
  oopCliente := TCliente.Create;
  { Carrega a lista de opções para o campo sexo }
  cboSexo.Items.Assign(oopCliente.Sexo.List);
end;
```

Temos que adicionar agora algumas funcionalidades ao nosso formulário. Vamos codificar o evento *OnClick* do botão *Open*, onde iremos fazer com que a tabela de clientes seja aberta, chamando o método *Open* de *oopCliente*, com o seguinte código:

```
{ Definimos qual o índice para a ordenação }
oopCliente.Index('NOME');
{ Aqui chamamos o método Open da nossa classe }
oopCliente.Open;
{ Método para mostrar os dados na tela }
Get_Values;
```

**Nota:** Se estiver usando um servidor SQL, você pode adaptar esse método para que utilize uma consulta parametrizada.

Declare os métodos "Set\_Values" e "Get\_Values" no formulário e implemente-os com o código da **Listagem 5**. Para cada botão chame o respectivo método da classe *TCliente* (variável *oopCliente*), seguido do método *Get\_Values*, semelhante ao que fizemos para o botão *Open* (com exceção dos métodos *Edit* e *Apply*). Para o botão *Post*, chame *Set\_Values* antes do método *Post* de *oopCliente*.

No evento *OnChange* do *edtLocateNome* (o *Edit* ao lado da opção de localizar), digite o código a seguir, que fará uma pesquisa na tabela de clientes para encontrar um registro que atenda a condição:

```
oopCliente.Locate('NOME', edtLocateNome.Text);
Get_Values;
```

**Nota:** Antes de executar a aplicação, assegure-se que o *DataModule* seja criado antes dos formulário (*Project>Options*).

No código-fonte disponível para download você pode encontrar também um formulário de busca criado usando técnicas de POO.

#### Listagem 5. Métodos do formulário

```
procedure TFrmInterface.Set_Values;
begin
  { Aqui você, atribui os valores recebidos nas
  propriedades de sua classe, para os componentes
  usados no formulário }
  oopCliente.Codigo.AsInteger :=
    StrToInt(edtCodigo.Text);
  oopCliente.Nome.AsValue := edtNome.Text;
  oopCliente.Endereco.AsValue := edtEndereco.Text;
  oopCliente.Bairro.AsValue := edtBairro.Text;
  oopCliente.Cidade.AsValue := edtCidade.Text;
  oopCliente.Cep.AsValue := edtCep.Text;
  oopCliente.Telefone.AsValue := edtTelefone.Text;
  oopCliente.Sexo.AsValue := edtSexo.Text;
  { Antes de passar o valor para a propriedade da
  classe, se é um valor válido }
  if oopCliente.Credito_Limite.ValidateCurrency(
    edtCredito_Limite.Text) then
    oopCliente.Credito_Limite.AsCurrency :=
      StrToCurr(edtCredito_Limite.Text);
  { Antes de passar o valor para a propriedade da
  classe, se é uma data válido }
  if oopCliente.DataCadastro.ValidateDate(
    edtDataCadastro.Text) then
    oopCliente.DataCadastro.AsDate :=
      StrToDate(edtDataCadastro.Text);
end;

procedure TFrmInterface.Get_Values;
begin
  { Aqui, atribuímos os valores recebidos nas
  propriedades de sua classe para os
  componentes usados no formulário }
  edtCodigo.Text := oopCliente.Codigo.AsValue;
  edtNome.Text := oopCliente.Nome.AsValue;
  edtEndereco.Text := oopCliente.Endereco.AsValue;
  edtBairro.Text := oopCliente.Bairro.AsValue;
  edtCidade.Text := oopCliente.Cidade.AsValue;
  edtCep.Text := oopCliente.Cep.AsValue;
  edtTelefone.Text := oopCliente.Telefone.AsValue;
  edtSexo.Text := oopCliente.Sexo.AsValue;
  edtCredito_Limite.Text :=
    oopCliente.Credito_Limite.AsValue;
  edtDataCadastro.Text :=
    oopCliente.DataCadastro.AsValue;
end;
```

Neste artigo vimos um exemplo prático de desenvolvimento usando Orientação a Objetos com Delphi. Espero assim que muitos de vocês tenham tirado algumas de suas dúvidas e é claro que a programação Orientada a Objetos pode ir muito além do que fizemos aqui. Um grande abraço a todos e até uma próxima oportunidade.



**Isaque Pinheiro** (isaque@neski.com.br) é empresário de uma Software House (NESKI - Nucleus of Enterprise and Software knowledge Integrated), onde exerce a função de Gerente de Novas Tecnologias, gerenciando o desenvolvimento do principal produto da Empresa na tecnologia n-camadas com Firebird e SQL Server 2000, também com conhecimento em IntraWeb e .NET.



## Saiba tudo sobre Banco de Dados

# Assine a SQL Magazine



# “OOrigem”

## Entrevista sobre Orientação a Objetos com Adail Muniz

Para esta edição da ClubeDelphi, sobre Orientação por Objetos, não poderíamos deixar de contar com a participação de um grande colaborador, reconhecido por sua dedicação e conhecimento no assunto. Convidamos para uma entrevista o ilustre colega Adail Muniz Retamal, engenheiro de sistemas da Borland. Atuando nas áreas de engenharia de software, orientação por objetos e metodologias ágeis, Adail conta um pouco de sua história e “migração” do mundo estruturado para o mundo orientado por objetos, quais foram as suas dificuldades, como foi a experiência, desafios e conquistas. Adail, que aqui na revista foi responsável por importantes colaborações (como o curso de POO e Bold – edições 50 a 52 e ECO – edições 58 a 61), procura incentivar o público, através de seus relatos, a adotar boas práticas em seus projetos, dando dicas de como começar e sugerindo alguns livros.

Pedimos perdão à língua pátria, da qual somos admiradores e defensores, pelo título dessa entrevista (“OOrigem”), mas não resistimos!

**Guinther Pauli** – Conte-nos um pouco sobre como você começou a programar, qual foi o seu primeiro computador, primeira linguagem, enfim, como tudo começou.

**Adail Muniz Retamal** - Foi em 1983, o ano em que fiz “contato”. Como presente de aniversário de 13 anos, ganhei um curso de BASIC numa escola no prédio ao lado do meu. Foi minha estréia na Informática. Os computadores eram CP-200, um modelo nacional do Sinclair ZX-81 inglês.

Em março do ano seguinte ganhei um TK-85, que guardo até hoje com cuidado, juntamente com meu “caderno BASIC de programas inocentes”. Para quem não conhece, ou para matar a saudade dos que viveram essa época, segue um exemplo na **Listagem 1**.

Que interessante, não? Linhas numeradas (notou o zero cortado?), somente letras maiúsculas, variáveis globais e com apenas uma letra. E a gente se divertia bastante!

Apesar do BASIC ser uma linguagem não estruturada (pelo menos na época), podíamos usar alguns artifícios para melhorar a organização do programa, como as sub-rotinas. No exemplo da **Listagem 1**, a instrução na linha 40 (GOSUB) chama uma sub-rotina a partir da linha 100, que é a responsável por calcular a idade em dias (lembrese, é um daqueles programas inocentes). A linha 120 faz a execução do programa continuar na instrução posterior ao GOSUB.

Com essa e outras técnicas conseguíamos reutilizar parte da lógica

do programa, também concentrando essas regras de negócio em um único local da listagem, facilitando a manutenção.

**GP** – Quando e como você conheceu o Pascal?

**AMR** – Em 1986/87, trabalhando com um CP-500 M80 (aquele com placa CP/M), conheci várias outras linguagens, incluindo uma tal de Pascal, que achei estranha, mas interessante. Conseguiram um interpretador e alguns programas de exemplo, mas estava muito ocupado com COBOL e, claro, BASIC, mas dessa vez num Unitron (Apple II).

Já no 2º ano de Engenharia Elétrica/Eletrônica (1989), depois de ter lutado com FORTRAN, Assembly e C, comecei a cursar algumas disciplinas com o Turbo Pascal, disponível nos laboratórios nas versões 3 e 4. No ano seguinte, apareceram as versões 5 e 5.5.

**GP** – Quando e como foi seu primeiro contato com o “mundo orientado por objetos”?

**AMR** - Depois de inúmeros programas e bibliotecas de funções (janelas, menus, entrada de dados, arquivos etc.), finalmente começaram as discussões em torno da tal versão 5.5 do Turbo Pascal, que oferecia um novo “tipo de dados” chamado “object” (que para nós era um record com funções e procedimentos) e propunha um modelo diferente de programação, chamado de *object-oriented programming*.

A **Listagem 2** mostra um dos “programas inocentes” da época. Ninguém sabia explicar direito o que significava essa tal de OOP e quais os benefícios, mas era o assunto nas rodinhas de alunos e discussões em sala de aula. Muitos falavam, mas quase ninguém entendia. Eu era um deles.

### Listagem 1. Programa em BASIC

```

1 REM CALCULADOR DE IDADE EM DIAS
10 PRINT "DIGITE SUA IDADE ";
20 INPUT I
30 PRINT I;" ANOS"
40 GOSUB 100
50 PRINT "VOCE JÁ VIVEU ";D;" DIAS"
60 PRINT
70 GOTO 10
100 REM CALCULA IDADE EM DIAS
110 LET D=I*365
120 RETURN

```



## Listagem 2. Programa em Turbo Pascal 5.5

```
program Cadastro;
uses Crt;
{ biblioteca de funções para controle de tela }

type
  { note que não colocavamos o T antes }
  Ficha = object
    Nome: string[50];
    { Olha o bug do milênio ai gente! }
    DataNasc: string[8];
    Telefone: string[12];
    procedure Imprimir;
end;

function FormataData(Data: string): string;
begin
  { não existia o Result ainda }
  FormataData := Copy(Data,1,2)+'/'+
    Copy(Data,3,2)+'/'+
    Copy(Data,5,2);
end;

procedure Ficha.Imprimir;
begin
  WriteLn('Nome      : ', Nome);
  WriteLn('Data Nasc.: ', FormataData(DataNasc));
  WriteLn('Telefone   : ', Telefone);
end;

var
  F: Ficha;
begin
  ClrScr; { limpa a tela }
  { não precisava criar o objeto! }
  F.Nome := 'Aleph';
  F.DataNasc := '281294';
  F.Telefone := '11 1234-5678';
  F.Imprimir;
  ReadLn;
end.
```

Até então, nos virávamos com os *records* e funções/procedimentos que os manipulavam. E em meio a tantos trabalhos e provas, o estudo dessa nova “onda” foi ficando em segundo plano.

No penúltimo período, na disciplina Arquitetura de Computadores, criei um simulador de um microprocessador chamado SAP (*Simple As Possible* – o mais simples possível), onde pude começar a experimentar algumas funcionalidades dessa tecnologia de objetos.

**GP** – Como começou sua experiência profissional? Como foi a evolução e experiência no mundo POO?

**AMR** – Corria o ano de 1993. Já havia me formado, casado e já era papai (da Stella Maris, que fez o banner para o OOPark, lembram?). Não estava empregado, mas fazia uns “bicos” com o Norton Utilities e desenvolvia alguns sisteminhas para um cunhado, o que ajudou muito!

Foi por volta do meio do ano que um grande amigo e guru de Pascal, “Tio Doni”, comprou a versão 7 do Turbo Pascal, que vinha com um *framework* OO chamado *Turbo Vision* (Figura 1).

Como ele havia construído seu próprio *framework* (e até hoje continua fazendo isso com o Delphi), não iria usar o TV (*Turbo Vision*), por isso me deu o manual (original!), que ensinava o que era OO e como usar a biblioteca de classes (que foi a avó da VCL).

Entre um bico e outro, fui lendo o livro, digitando e testando cada exemplo, até que um dia “a ficha caiu”! Percebi que minha mente havia mudado! Compreendi que esse método de programação era muito mais natural que os anteriores e que podia imaginar soluções mais abrangentes e reutilizáveis.

De fato, o *Turbo Vision* foi a VCL para o DOS, em modo texto. Veja algumas classes: *TObject*, *TApplication*, *TView*, *TWindow*, *TStringList*, *TMenuBar*, *TListBox*, *TButton*. Algo familiar? Ah, sim, também já era dirigido por eventos e podíamos criar várias janelas, modais e não-modais. E nas BBS’s era possível encontrar muitos utilitários, como um programa que permitia desenhar a janela, como fazemos no Delphi (WYSIWYG), e que gerava o código para colarmos no nosso programa!

Os objetos no *Turbo Vision* já eram criados dinamicamente (usando ponteiros). Também foram introduzidos o construtor (chamado *Init*) e o destrutor (chamado *Done*), além do *Free*, que já liberava a memória e chamava o destrutor.

Munido dessa “arma”, nesse mesmo ano desenvolvi meu primeiro sistema OO, um controle de convênio médico, para outro cunhado (afinal, para que servem os cunhados, né?).

Quanto tempo de estudo? Cerca de três meses (dia, noite e madrugada), desde quando ganhei o manual do *Turbo Vision* até começar de fato a desenvolver o sistema. Mas foi durante o desenvolvimento que fui entender mais a fundo, criando meu próprio *framework* baseado no TV.

Durante esse período, a paciência e apoio de minha esposa, Éliada, foi fundamental! Ela sempre acreditou que, se eu achava que esse era o caminho a seguir, então o esforço valeria à pena!



Figura 1. Aplicação feita com Turbo Vision

**Treinamento à Distância**

# **E-Commerce com Delphi 2005 e ASP.NET**

## **O que falta para você adquirir O SEU?**

Com este treinamento você aprenderá a construir poderosas aplicações WEB utilizando o Delphi 2005 e as tecnologias ASP.NET, WebBroker, Intraweb e WebSnap. E o que é melhor: você poderá fazer o curso no seu horário, sem precisar freqüentar a sala de aula. Com a nova modalidade de estudos que está se tornando padrão na Internet: curso à distância.



Este treinamento contém três cursos em um só : Desenvolvimento com ASP.NET no Delphi 2005, Desenvolvimento com IntraWeb no Delphi 7 e Desenvolvimento com WebBroker no Delphi 5!

saiba mais em: [www.devmedia.com.br/curso/ecommerce2005](http://www.devmedia.com.br/curso/ecommerce2005)



**-GP** – *Como começou com o Delphi? Soube que há muitos anos você participou de um concurso da Borland – US. Como foi?*

**AMR** - Bom, isso foi em 1994/95. Já empregado numa grande empresa, tive que usar o Visual Basic 3 para desenvolvimento em Windows. Para quem estava acostumado a programar OO, senti falta da herança, por exemplo, mas conseguia fazer algumas coisas legais. Mas quando saiu o Delphi 1.0, fiz o possível para que a empresa o adotasse, mas já era tarde. O que foi que fiz, então? Busquei outra oportunidade onde pudesse usá-lo, claro!

Passei alguns meses num projeto para um hospital e, depois, mais alguns meses com um amigo de infância, Miguel (outro apaixonado pelo Pascal), que havia montado o primeiro provedor de Internet na minha cidade! Lá desenvolvi vários programas, como um medidor de tempo de uso do provedor e o instalador do kit de acesso (configurando os outros programas).

Graças à Internet fiquei sabendo de um concurso promovido pela Borland sobre construção de componentes para o Delphi. Mandei o meu, só para participar. O interessante é que uns seis meses depois, chegou uma caixa da Borland lá em casa. Dentro dela estava o *Developer's Guide* e o *Database Developer's Guide* do Delphi 2, junto com uma carta e um cheque de US\$100,00! Eu tinha sido um dos premiados! Isso virou notícia no jornal e na TV! Interior é assim mesmo, mano!

**GP** – *Você também atuou na área acadêmica como professor, certo? Como foi essa experiência?*

**AMR** - No primeiro semestre de 1995 lecionei Linguagens Comerciais numa faculdade particular. Usamos COBOL, dBase III Plus e Clipper, mas o VB 3 e o Delphi foram usados nos semestres seguintes. Em fevereiro de 1996 passei num concurso para professor substituto no Departamento de Informática da Universidade Federal de Uberlândia, para ministrar disciplinas de Introdução à Informática, linguagens e adivinha? Programação Orientada por Objetos! Além disso, também fiquei responsável pelos laboratórios dos alunos e professores, o que incluía a instalação dos softwares.

Não preciso dizer que a linguagem oficial era o Delphi, né? E na disciplina de POO usamos *Smalltalk* (por motivos históricos) e Delphi, claro! Ah, uns “rebeldes” usaram C++ para o trabalho final, mas eu não levei para o lado pessoal.

Fazíamos das aulas de laboratório um verdadeiro ambiente de desenvolvimento, onde eu propunha um problema e todos participavam da análise, projeto e programação. Foram inesquecíveis sessões de *group programming*!

**GP** – *Quando você montou sua própria empresa e como foi?*

**AMR** – Em 1997, após o término do contrato, saí da Universidade e montei minha própria empresa (com outro cunhado; família grande, hein?), a *Heptagon Projetos Especiais*. Prestava serviços de consultoria, suporte técnico, desenvolvimento e treinamento, especialmente usando o Delphi (e não era pirata, viu?).

Particpei de projetos bastante interessantes e variados, abrangendo comunicação de dados, integração de sistemas, interface com

hardware, computação gráfica, multimídia e até mesmo sistemas comerciais. E em todos eles utilizei os conceitos de orientação por objetos, o que certamente contribuiu muito para a eficácia e eficiência do processo de desenvolvimento.

**GP** – *Como e quando você começou a trabalhar na Borland?*

**AMR** - Como a maioria dos usuários Borland, sempre me imaginei tendo um e-mail @borland.com! Costumava brincar com alguns colegas sobre isso, até mesmo enviando mensagens com o endereço adail@borland.com (usando Telnet diretamente num servidor SMTP). Finalmente, em julho de 2002, graças a uma série de pessoas (não vou citar nomes, mas elas sabem da minha gratidão!) e fatores, finalmente realizei dois sonhos: mudar para São Paulo e trabalhar na Borland!

A partir daí, comecei a me envolver cada vez mais com o Delphi e os treinamentos oficiais, tendo inclusive criado alguns cursos, como o de IntraWeb e Bold.

E já que falei em Bold, aqui começou minha história com os artigos e a ClubeDelphi, onde agora costumo escrever, entre outras coisas, sobre o ECO (o irmão mais novo do Bold). Em breve teremos mais surpresas!

**GP** - *E hoje em dia, quais são suas principais áreas de interesse?*

**AMR** - Meu interesse continua muito grande na tecnologia de objetos. Depois da UML ficou mais simples pensar e escrever OO, despertando o interesse pela modelagem e também pela análise. A descoberta da FDD (*Feature-Driven Development*) em 2003 também me ajudou muito a organizar meu trabalho com objetos. Hoje também atuo com C# e Java, mas sempre tenho o Delphi como referência, principalmente depois do ECO. Seria muito bom ver o ECO disponível para outras versões do Delphi e também para outras linguagens e plataformas, como Java, além de poder funcionar como um verdadeiro servidor de objetos remotos, num esquema similar ao J2EE! Não creio que esse seja um “sonho” impossível...

**GP** - *Para quem está começando no mundo POO, quais livros você indicaria? Quais livros você utilizou em sua “jornada”?*

**AMR** - Além dos manuais do Turbo Pascal 5.5 e do *Turbo Vision*, também li alguns livros sobre Objective-C (alternativa OO ao C++) e Smalltalk. O livro clássico sobre OO é o do Bertrand Meyer, *Object-Oriented Software Construction*, que usei como livro texto quando lecionei POO. O livro do Grady Booch, *Object-Oriented Design with Applications*, também é muito citado.

Peter Coad também foi um dos pioneiros nessa área, lançando excelentes livros e muito práticos. Mas além de falar sobre a programação, ele também propôs métodos para projeto e análise OO, sobre os quais tenho escrito e apresentado nas palestras.

Scott Ambler escreveu *Object Primer*, que é uma ótima introdução a todo o processo de desenvolvimento OO, incluindo boas dicas de modelagem e documentação. Existem boas opções em português também, de acordo com o estilo e necessidade do leitor.





**GP** – Seu último recado

**AMR** – Quero deixar à comunidade Delphi minhas palavras de incentivo, para que os colegas sintam-se encorajados a adotarem disciplinas e hábitos que comprovadamente contribuam para sua eficácia e eficiência. Os desafios são cada vez mais complexos, mas temos à nossa disposição um verdadeiro arsenal, tanto de fundamentos quanto de ferramentas, capaz de nos elevar a níveis cada vez maiores na nossa busca pela excelência. Eu vejo esse compromisso tanto na Borland quando na ClubeDelphi!

E como diz um ditado gaúcho: "E se não foi bem assim, foi bem mais ou menos assim".

Um grande abraço e até a próxima!

## Links

[bdn.borland.com/museum](http://bdn.borland.com/museum)

Museu da Borland, onde podem ser baixadas verdadeiras relíquias como o Turbo Pascal 1.0 e 5.5.

[oldcomputers.net/zx81.html](http://oldcomputers.net/zx81.html)

Quer (re)ver o ZX-81?

[adailmr.sites.uol.com.br/Delphi/sap.zip](http://adailmr.sites.uol.com.br/Delphi/sap.zip)

Link para baixar o SAP (é bem antigo)



**Adail Muniz Retamal** ([adail.retamal@borland.com](mailto:adail.retamal@borland.com)) é engenheiro eletrônico por formação e engenheiro de software por opção. Atualmente é engenheiro de sistemas na Borland Latin América.

## Glossário (para os mais novos)

*Assembly*: nome dado à linguagem de máquina do processador (não confundir com o do .NET)

*BASIC*: Beginner's All-purpose Symbolic Instruction Code (agora também na versão visual)

*BBS*: Bulletin Board System (o irmão mais velho da Internet)

*Clipper*: Compilador para a linguagem do dBase III (que depois criou vida própria)

*COBOL*: COmmon Business-Oriented Language (a linguagem comercial favorita)

*CP/M*: Control Program for Microcomputers (sistema operacional, avô do DOS)

*FORTRAN*: FORmula TRANslation (a linguagem científica favorita)

*WYSIWYG*: What You See Is What You Get (O Que Você Vê É o Que Você Recebe)

*ZX-81*: computador pessoal fabricado pela Sinclair (Inglaterra) (CP-200 e TK-85 foram clones brasileiros)



**COMPONENTES E FERRAMENTAS  
PARA DELPHI**

**Fone/Fax  
(14) 3454-7880**

### Quer imprimir em matriciais?

#### **RDPrint 3.0h**

- ✓ Imprime em matriciais como no MS-DOS
- ✓ Imprime em portas LPT/COM/USB (Local e Rede)
- ✓ Fonte 10, 12, 17, 20 cpp
- ✓ Espaçamento em Sextos/Oitavos
- ✓ Fácil impressão de Notas Fiscais, Boletos, Duplicatas, Cheques e todo o tipo de Relatório
- ✓ Compatibilidade com todos os modelos de Impressoras Matriciais, Jato de Tinta e Laser
- ✓ Compatível com Windows NT/2000/98/ME/XP

### Quer Proteger sua Aplicação?

#### **RD Acesso 2.1**

- ✓ Cadastro de usuários com privilégios
- ✓ Desabilita automaticamente os itens do Menu Principal (Controle de Acesso)
- ✓ Protege seu aplicativo contra cópias Piratas ou uso indevido
- ✓ Registro de Software com Data e Vencimento e Número de Série
- ✓ Fácil Uso: Basta arrastá-lo para o Form Principal e ligá-lo ao TMainMenu

### Outras Ferramentas & Componentes:

- ✓ RDReport - Gerador de Relatórios
- ✓ RDInstall - Instalador de BDE (1 disco) e Aplicativos
- ✓ RDExtenso - Extenso de Valores com separação de sílabas
- ✓ RDCheck - Valida IE (todos os Estados/PIS/CPF/CGC/Crédito)
- ✓ RDBarra - Código de Barras para Canvas e QuickReport
- ✓ RDFormula - Executa fórmulas e cálculos matemáticos
- ✓ RDSerial - Comunicação Serial / Automação

### em nosso site:

- Versões Demonstração
- Exemplos de Uso
- Documentos e Suporte

**[www.deltress.com.br](http://www.deltress.com.br)**

**FAÇA UM DOWNLOAD AGORA E COMPROVE!**



# Padrões de Projeto e POO

Paulo Roberto Quicoli

## Parte I - Preparando-se para mudanças

Com o advento do .NET, muitos desenvolvedores Delphi têm se deparado com um novo paradigma, um novo conceito, imposto pela Arquitetura .NET, a Orientação a Objetos, tema abordado nesta edição. Isso não deveria ser nenhuma novidade para os “Delphianos”, já que o Delphi nasceu orientado a objetos, porém essa não é a realidade que encontramos.

Tenho visto grandes esforços de grupos em divulgar conceitos e uso de ferramentas para a programação Orientada a Objetos, e o que venho apresentar não é nenhuma novidade. Os padrões de projeto (*Design Patterns*) surgiram de estudos na área de arquitetura civil e foram posteriormente aplicados na área de análise e desenvolvimento de software, sendo considerados como boa prática.

Não quero aqui promover um longo histórico sobre eles, vamos partir para seus conceitos que podem ser aplicados em qualquer análise e desenvolvimento OO, mostrando de forma prática e real seu uso, vantagens e desvantagens.



### Definindo padrões de projeto

Segundo Martin Fowler, “um padrão é uma idéia que tem sido útil em um contexto prático e que provavelmente será útil em outros”, ou seja, é uma solução já testada para problemas de modelagem que acontecem sempre. Talvez quem não utilize a Orientação a Objetos na prática está se perguntando agora que problemas são esses. Durante esta série de artigos esses problemas serão apresentados juntamente com as respectivas soluções.

### Princípios

Quando nos deparamos com um sistema que é de difícil manutenção, perguntamos onde está o problema? Talvez por nossa “herança estruturada” somos levados a pensar que a falha foi na coleta dos dados, e que melhorando essa obtenção dos usuários, o sistema se torne mais fácil de manter.

Existe uma verdade única, os requisitos sempre tendem a mudar. O usuário sempre desejará algo novo, algo que acompanhe as mudanças de sua regra de negócio, portanto, melhores técnicas de coleta de informações podem ajudar, mas não resolvem o problema.

É nisso que a visão dos padrões podem ajudar, o foco passa de “como coletar de forma melhor os dados” para “como escrever meu código de tal forma que se adapte bem à constante mudança de requisitos”.

Existem alguns princípios de modelagem e programação OO que levam os padrões de projetos a serem considerados como boas práticas, esses mesmos conceitos podem ser aplicados sem a utilização dos padrões. Portanto, a idéia é entender esses princípios e procurar aplicá-los em nosso desenvolvimento. São eles: Modelar para interfaces, favorecer composição sobre herança e encontrar o que varia, e encapsulá-lo.

### Modelando para interfaces

Existem dois problemas que impedem um sistema de ser facilmente alterado quando há uma mudança em seus requisitos: **baixa coesão** e **alto acoplamento**.

- *Baixa coesão*: Isso é detectado quando encontramos rotinas/classes que realizam várias funções e essas são dependentes entre si, ou seja, é o quanto as operações de uma rotina/classe dependem de outras;
- *Alto acoplamento*: Encontramos um alto acoplamento quando temos rotinas/classes que são muito dependentes de outras, aonde uma alteração nessa

www.clubedelphi.net



conduz a um cascadeamento de alterações, ou seja, é o quanto as rotinas/classes estão dependentes uma das outras.

Para amenizar, ou evitar esses problemas, pode-se aplicar o conceito de modelar para interfaces. Interface aqui não é a interface gráfica com o usuário do seu sistema, e sim o conjunto de operações, propriedades e atributos que compõe uma determinada classe.

É através de sua interface que uma classe se faz conhecida e interage com outras classes. Para aplicar esse conceito, é sugerido que heranças iniciem a partir de classes abstratas, que nada mais são do que interface pura, já que seus métodos são implementados por seus descendentes.

Para entender isso melhor, façamos um pequeno exemplo que mostra essa idéia que poderá ser utilizada em seus sistemas. Esse exemplo utiliza Firebird 1.5 e Delphi 7. Nele vamos fazer duas consultas diferentes utilizando um único formulário, porém esse formulário não saberá qual consulta que está em uso. Vamos abstrair o conceito de "consulta" e encapsularemos em duas classes distintas, chamadas: *TConsultaCliente* e *TConsultaFuncionario*.

Apesar de serem classes distintas, ambas possuem algo em comum: são classes do tipo *TConsulta* que por sua vez é abstrata (Figura 1).

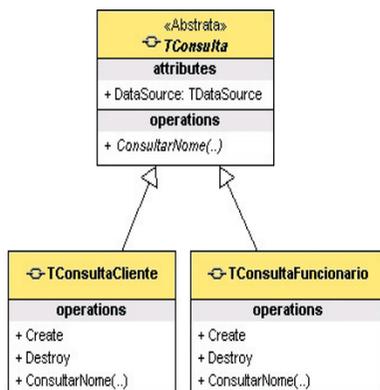


Figura 1. Diagrama de classes

### Construindo nosso exemplo

Inicie um novo projeto no Delphi e estruture o formulário principal conforme a Figura 2. Adicione um DataModule e alguns componentes de acesso a dados, conforme a Figura 3 (estou utilizando os componentes da paleta *InterBase*, fique a vontade para utilizar os de sua preferência). Faça normalmente a ligação entre os componentes de acesso a dados, relacionando conexão, transação, queries e *DataSources*.

Para esse exemplo, o *IBDatabase* aponta para o banco *Employee.fdb*, incluído no diretório de instalação do Firebird. Preencha a propriedade *SQL* dos *IBQueries* conforme a Tabela 1. Não esqueça de configurar os parâmetros, criados nas instruções SQL, através da propriedade *Params* do *IBQuery*:

*(DataType = ftString e ParamType = ptInput).*



Figura 2. Formulário principal



Figura 3. Data Module do exemplo

IBQuery	SQL
BuscaCliente	SELECT Cust_No, Customer, Address_Line1 FROM Customer WHERE Customer Starting With :NOME ORDER BY Customer
BuscaFuncionario	SELECT Emp_No, First_Name, Hire_Date FROM Employee WHERE First_Name Starting With :NOME ORDER BY First_Name

Tabela 1. Configuração dos IBQueries

Crie uma nova unit, dando-lhe o nome de "ConsultasU.pas". Nessa unit vamos criar as classes que encapsularão o conceito de busca, conforme a Listagem 1. Observe que na classe *TConsulta* o método *ConsultarNome* foi declarado como *virtual* e *abstract*.

Definir um método como *virtual* significa que ele poderá ser sobrescrito por seus descendentes e *abstract* indica que ele não será implementado em sua classe base, e sim em seus descendentes, que no nosso caso são *TConsultaCliente* e *TConsultaFuncionario*. Vejamos agora a implementação do método *ConsultarNome* na Listagem 2. Os construtores e destrutores das classes podem ser vistos na Listagem 3.

Com nossas classes prontas, podemos agora criar nossa classe cliente, que será o formulário de consulta. Podemos chamá-la de classe cliente porque ela é que utilizará as classes definidas anteriormente.

Para tanto, crie um novo formulário e o configure conforme a Figura 4, chamado "ConsultaGeralF". Na unit desse formulário vamos inserir duas propriedades públicas que são acessadas por métodos privados conforme a Listagem 4.

Agora dê um duplo clique sobre o botão e no evento *OnClick* e digite o seguinte código:

```
Consulta.ConsultarNome(Edit1.Text);
```

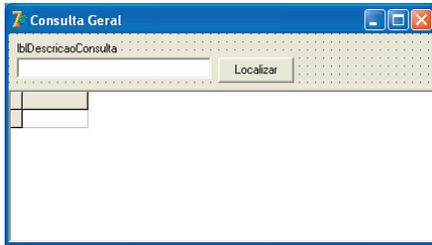


Figura 4. Formulário de consulta

### Listagem 1. Classes de consulta

```

unit ConsultasU;

interface

uses DB;

type
  TConsulta = class(TObject)
  private
    FDataSource: TDataSource;
  public
    procedure ConsultarNome(Nome: string);
    virtual; abstract;
    property DataSource: TDataSource read FDataSource
    write FDataSource;
  end;

  TConsultaFuncionario = class(TConsulta)
  public
    constructor Create;
    destructor Destroy; override;
    procedure ConsultarNome(nome: string); override;
  end;

  TConsultaCliente = class(TConsulta)
  public
    constructor Create;
    destructor Destroy; override;
    procedure ConsultarNome(Nome: string); override;
  end;

implementation

```

### Listagem 2. Implementação do método ConsultarNome

```

procedure TConsultaFuncionario.ConsultarNome(
  Nome: string);
begin
  { declare em uses o DM do projeto }
  with DM.BuscaFuncionario do
  begin
    Close;
    ParamByName('NOME').AsString := Nome;
    Open;
  end;
end;

procedure TConsultaCliente.ConsultarNome(
  Nome: string);
begin
  with DM.BuscaCliente do
  begin
    Close;
    ParamByName('NOME').AsString := Nome;
    Open;
  end;
end;

...

```

### Listagem 3. Construtores e Destrutores

```

constructor TConsultaFuncionario.Create;
begin
  DataSource := DM.dsBuscaFuncionario;

```

```

end;

destructor TConsultaFuncionario.Destroy;
begin
  DM.BuscaFuncionario.Close;
  inherited;
end;

constructor TConsultaCliente.Create;
begin
  DataSource := DM.dsBuscaCliente;
end;

destructor TConsultaCliente.Destroy;
begin
  DM.BuscaCliente.Close;
  inherited;
end;

```

### Listagem 4. Interface do formulário de consulta

```

{ Declare ConsultasU no uses }
private
  FConsulta: TConsulta;
  procedure SetConsulta(const Value: TConsulta);
  procedure SetTextoEdit(const Value: string);
public
  property Consulta: TConsulta read FConsulta
  write SetConsulta;
  property TextoEdit: string write SetTextoEdit;
end;

{ implementação dos métodos }
procedure TConsultaGeraIF.SetConsulta(
  const Value: TConsulta);
begin
  FConsulta := Value;
  DbGrid1.DataSource := Value.DataSource;
end;

procedure TConsultaGeraIF.SetTextoEdit(
  const Value: string);
begin
  lblDescricaoConsulta.Caption := Value;
end;

```

Veja que a propriedade *Consulta*, é do tipo *TConsulta*, que é abstrata e base das classes que criamos. É através dela que serão realizadas as consultas. Você deve estar se perguntando agora como que esse formulário vai servir para ambas as consultas já que declaramos que nossa propriedade como *TConsulta* e *TConsulta.ConsultarNome* não possui implementação.

Agora vem uma das vantagens da Orientação a Objetos. Através da herança utilizada (**Figura 1**), podemos passar um objeto do tipo *TConsultaFuncionario* no lugar de um *TConsulta*, porque ele herda de (é do tipo) *TConsulta* e como o método *ConsultarNome* é abstrato, sua implementação está nos descendentes de *TConsulta*. É o polimorfismo em ação.

Para vermos a aplicação funcionando, configure o evento *OnClick* dos botões do formulário principal e adicione os códigos respectivos conforme a **Listagem 5**. Observe as seguintes linhas:

```

lConsultaClientes := TConsultaCliente.Create;
ConsultaGeraIF.Consulta := lConsultaClientes;

```



### Listagem 5. Implementação dos eventos OnClick

```

{ adicione no uses a unit das classes
  e do formulário de pesquisa }
{ botão Clientes }
var
  TConsultaClientes: TConsultaCliente;
begin
  TConsultaClientes := TConsultaCliente.Create;
  ConsultaGera1F.Consulta := TConsultaClientes;
  ConsultaGera1F.Caption := 'Consulta de Clientes';
  ConsultaGera1F.TextoEdit :=
    'Informe o nome do cliente';
  ConsultaGera1F.ShowModal;
  TConsultaClientes.Free;
end;
{ botão Funcionários }
var
  TConsultaFuncionario: TConsultaFuncionario;
begin
  TConsultaFuncionario := TConsultaFuncionario.Create;
  ConsultaGera1F.Consulta := TConsultaFuncionario;
  ConsultaGera1F.Caption := 'Consulta de Funcionários';
  ConsultaGera1F.TextoEdit :=
    'Informe o primeiro nome do funcionário';
  ConsultaGera1F.ShowModal;
  TConsultaFuncionario.Free;
end;

```

Estamos criando uma instância de *TConsultaCliente* e passando ao formulário como um *TConsulta*. Execute a aplicação e veja o resultado. O formulário de pesquisa se adapta em tempo de execução

para refletir o respectivo objeto, de acordo com o seu tipo. Com isso, estamos aplicando em um exemplo real importantes conceitos da programação orientada a objetos: herança e polimorfismo.

### Conclusões

Nesta primeira parte, tivemos uma pequena introdução aos padrões de projeto e iniciamos um estudo sobre os conceitos que os formam. Aplicando o primeiro conceito conseguimos diminuir o acoplamento, isso é, o fato do formulário de consulta ter conhecimento apenas da classe *TConsulta*. Com isso, podemos criar vários descendentes de *TConsulta* sem ter a necessidade de alterar o formulário.

Aguardem o próximo artigo, onde iremos demonstrar os próximos conceitos para então aprofundarmos o tema propriamente dito, Padrões de Projeto. Um abraço é até lá!



**Paulo Roberto Quicoli** ([pauloquicoli@gmail.com](mailto:pauloquicoli@gmail.com)) é analista e programador da Control-M Informática. Trabalha com Delphi, desde de sua primeira versão, e Firebird desenvolvendo aplicações cliente/servidor. Formado em Tecnologia de Processamento de dados pela FATEC, na cidade de Taquaritinga/SP

**Mais do que uma simples ferramenta de banco de dados  
Alta Performance • Grande Portabilidade • Flexibilidade Sem Limites**



# V8

**c-treeSQL™**  
suporte multi-plataforma

**Triggers a nível ISAM via**  
notificação de arquivos

**Suporte a arquivos**  
em memória

**Interface**  
Nativa .NET

**Monitoramento de**  
performance

**E muito mais!!**



**Faça hoje mesmo o  
DOWNLOAD GRÁTIS da  
edição de avaliação!**

[www.faircom.com/go/br/1way](http://www.faircom.com/go/br/1way)



**FairCom®**  
do Brasil

**Oriente o seu desenvolvimento pelo melhor caminho: conheça o c-tree!**

SGBD desde 1979 • 11-3872-9802 • [www.faircom.com.br](http://www.faircom.com.br) • [brasil@faircom.com.br](mailto:brasil@faircom.com.br)





# Novidades da Delphi Language

Laércio Queiroz

## Parte IV - Os novos especificadores de acesso / visibilidade

Seguindo a série de artigos que tratam das alterações na Delphi Language para suporte ao .NET, examinaremos os novos especificadores de acesso do Delphi e o seu comportamento em nossas aplicações.



**Nota:** Muitos recursos que foram incluídos no compilador do Delphi 8 for .NET estão agora disponíveis para o Win32, de forma que também serão apresentados nesta série.

### Especificadores de acesso no Delphi

Nas versões do Delphi dedicadas ao Win32, os programadores podem especificar níveis de acesso a alguns elementos da classe. Essa técnica de encapsular os elementos de uma classe é uma característica marcante da POO (Programação Orientada a Objetos). Sendo assim, várias outras linguagens de programação que trabalhem sobre esse modelo disponibilizam esse recurso de encapsulamento dos elementos. No Delphi, as classes poderiam ser construídas sobre quatro especificadores, que são:

**Private (Privado):** Os elementos declarados nessa seção da classe podem ser acessados somente pelos métodos da própria classe. Todavia, no Delphi, as classes situadas na mesma unit têm livre acesso aos campos privados de outra classe;

**Protected (Protegido):** Os elementos declarados podem ser acessados pelos métodos da própria classe, pelas suas classes descendentes e por outros códigos implementados na mesma unit;

**Public (Público):** Os elementos declarados podem ser livremente acessados por outras classes;

**Published (Publicado):** Os elementos são compilados com informação de RTTI (*Runtime Type Information*) e podem ser listados no *Object Inspector*.

Se você é um programador Delphi experiente, já deve ter visto como acessar (e com uma certa facilidade) as informações protegidas de uma classe. Isso se deve ao fato de que o Delphi "abre a caixa preta" com os elementos protegidos de uma classe, para uma outra classe descendente e todas as outras implementadas na mesma unit. Observe o exemplo a seguir:

```
...
type
  TClasse = class
    protected
```

```
    FCampo: Integer;
  end;
  ...
```

Em uma outra unit é possível manipular o valor do campo protegido *FCampo*, simplesmente estendendo uma classe a partir da anterior, como no exemplo:

```
type
  THackClasse = class (TClasse)
  end;
  ...
var
  obj: TClasse;
begin
  obj := TClasse.Create;
  obj.FCampo := 10; { erro, não compila! }
  THackClasse(obj).FCampo := 10; { compila!! }
end;
```

Por essa "deficiência" da Delphi Language no acesso a métodos protegidos e privados, foram adicionados na linguagem os novos especificadores *strict private* e *strict protected*. Esses novos especificadores seguem a mesma lógica dos seus antecessores *private* e *protected* (respectivamente), porém, mantendo um nível de visibilidade mais seguro exigido pelo CLS, evitando que o "Hack protected" aconteça. Com isso, se você declarar um membro usando o especificador *strict private*, ele não pode ser acessado por outras classes mesmos que estejam na mesma unit.

A **Listagem 1** mostra um exemplo que deixa clara a diferença entre o *Private* e *Strict Private*.

### Listagem 1. Strict Private vs. Private

```
type
  TClasseA = class
    private
      FTeste1: string;
    strict private
      FTeste2: string;
    end;

  TClasseB = class
    public
```





```

procedure Teste;
end;
...
procedure TClasseB.Teste;
var
  obj: TClasseA;
begin
  obj := TClasseA.Create;
  // Pode ser acessado pois é Private
  obj.FTeste1 := 'ola';
  // Não pode ser acessado pois é Strict Private
  obj.FTeste2 := 'ola'; // erro
end;

```

## Conclusões

Vimos neste artigo uma das novas especificações de linguagem imposta pelo CLS para suporte ao .NET, notória apenas aos programadores Delphi. Para maiores informações, examine o Help do Delphi e fique ligado na ClubeDelphi!

**Nota:** Vale a pena ressaltar que os outros especificadores de acesso (reputados como falhos pelo CLS) não foram removidos da linguagem e as operações citadas anteriormente ainda podem ser efetuadas no ambiente .NET.



**Laércio Santos de Queiroz** ([laercio\\_queiroz@hotmail.com](mailto:laercio_queiroz@hotmail.com)) atua a vários anos como desenvolvedor, especializado em tecnologias relacionadas a Web e desenvolvimento multi-camadas. Atualmente presta consultoria e treinamentos avançados, sobre desenvolvimento de sistemas e bancos de dados pelo Centro Avançado de Informática, em Salvador. Além disso, possui sólidos conhecimentos em várias linguagens e metodologias de programação. Quando não está programando, gosta de tocar bateria, fotografar e de ler bons livros.

# NOVO Pervasive PSQL v9™

- O Paradox continua corrompendo e inutilizando seus arquivos?
- Você ainda tem perda de dados e índices em seu banco de dados?
- A velocidade e a integridade de sua base de dados está te deixando louco?
- Seu banco de dados não é tão robusto quanto parece?

## Migre para o novo Pervasive PSQL v9™

Totalmente compatível com o modelo orientado a registros do Paradox, além de ser 100% ANSI SQL. Com os componentes nativos 100% compatíveis com o TTable/TQuery, você não precisa reescrever a sua aplicação. Além de solucionar diversos problemas, você terá os seguintes benefícios:

- Não haverá mais perda de dados ou corrupção de índices/tabelas, instale e esqueça!
- Alto desempenho na rede que não degrada com o aumento do tamanho das tabelas ou número de usuários
- Acessa o banco de dados transacional/relacional sem a necessidade de mapear um diretório
- Não vai mais precisar instalar e configurar o BDE/dbExpress nas estações
- Reduz drasticamente o suporte ao cliente, pois o banco de dados é totalmente isento de manutenção

## Vantagens

- Projetado e desenvolvido baseado nas necessidades e restrições das pequenas e médias empresas
- Maior economia para o desenvolvedor e para o usuário, devido ao seu design integrado, ser auto-configurável e auto-ajustável, o Pervasive PSQL v9™ dispensa a administração minimizando os custos de suporte e maximizando a satisfação dos clientes
- Compatível com padrões do mercado. Inclui drivers ODBC, OLE/DB, JDBC, .NET e componentes/interfaces de acesso para Delphi, C++ Builder, C/C++, VB e COBOL
- Rápido e confiável em sistemas operacionais Linux, Netware e Windows
- Escalável desde um único usuário até ambiente cliente/servidor sem alterações no aplicativo

## Novidades

- Mecanismo de busca de texto livre (FULL TEXT SEARCH)
- Performance relacional e transacional incomparável
- Novas funcionalidades e sintaxes de SQL
- Componentes nativos para Delphi (Table e Query)
- Suporte nativo a COBOL (Occurs e Redefines)
- Suporte a arquivos de 128 GB sem extensões
- Novos utilitários gráficos e de linha de comando

PERVASIVE® • [www.stern.com.br](http://www.stern.com.br) • (11) 3078-1690 • [info@stern.com.br](mailto:info@stern.com.br)

# Criando controles ASP.NET

Laércio Queiroz

## User Controls e Web Controls

Muitas vezes, no desenvolvimento de aplicações Web, os controles existentes não suprem totalmente as nossas necessidades, seja por motivos de layout, reutilização de código ou qualquer outro, impulsionando-nos a escrever os próprios controles.

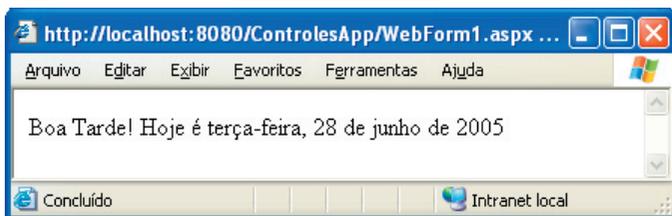
O ASP.NET, assim como a VCL do Delphi, possui classes básicas que facilitam a criação de novos controles, permitindo que o desenvolvedor estenda os controles do framework de acordo com suas necessidades. Esse será o assunto discutido neste artigo.

O ASP.NET possibilita a criação de dois tipos de controles: *User Controls* e *Web Custom Controls*. O Delphi dá suporte a esses dois modelos de desenvolvimento de controles, que podem ser utilizados em várias páginas e projetos.

### Criando um User Control

Um *User Control* (ou controle de usuário) é a forma mais simples de criação de controles em ASP.NET. Os *User Controls* possibilitam a criação de novos controles da mesma forma como são criadas as páginas ASP.NET comuns. Os *User Controls* têm a extensão *ascx* e são formados basicamente por *Pagelets* ("trechos" de página), que contêm a maioria dos conceitos e recursos das páginas *aspx*, como outros controles, eventos de página etc.

Para ilustrarmos a criação de um *User Control*, mostraremos o desenvolvimento de um controle que gera um texto de saudação na página, dependendo da hora do sistema, que no final será semelhante à **Figura 1**.



**Figura 1.** Exibição de uma página ASP.NET com o User Control

No Delphi 2005, crie uma nova aplicação do tipo *ASP.NET Web Application*, com o nome de "ControlesApp". Após ter sido criada a aplicação, vá até o menu *File|New>Other* e na seção *Delphi for .NET Projects>New Delphi ASP Files*, selecione a opção *ASP.NET User Control*.

No *Project Manager*, clique com o botão direito sobre o *User Control*

e selecione a opção *Rename*, dê o nome de "saudacao.ascx". Vamos criar a interface do *User Control*, para isso, coloque na página um *Label* e defina a propriedade ID como "Time". No evento *Load* do controle digite o código da **Listagem 1**.

### Listagem 1. Código do evento Load

```
{ Declare em uses System.Globalization }
var
  ci: CultureInfo;
  oData: DateTime;
begin
  oData := DateTime.Now;
  { Formato Brasileiro }
  ci := CultureInfo.Create('pt-BR');
  if oData.Hour < 12 then
    Time.Text := 'Bom Dia! Hoje é '+
      DateTime.Now.ToString(
        'dddd, d'' de ''MMMM'' de ''yyyy'', ci)
  else if (oData.Hour >= 12) and
    (oData.Hour < 18) then
    Time.Text := 'Boa Tarde! Hoje é '+
      DateTime.Now.ToString(
        'dddd, d'' de ''MMMM'' de ''yyyy'', ci)
  else
    Time.Text := 'Boa Noite! Hoje é '+
      DateTime.Now.ToString(
        'dddd, d'' de ''MMMM'' de ''yyyy'', ci);
end;
```

### Utilizando o User Control

Nosso controle já pode ser utilizado em outras páginas. Para inserir o controle em uma página, vá até o menu *Insert>Insert User Control* e na opção *User control file*, digite o caminho do arquivo *saudacao.ascx* ou selecione-o clicando no botão *Browse*.

Assim, o Delphi adiciona no layout o componente (**Figura 2**) e coloca na página a diretiva *@Register*, que importa o *User Control* indicado em *Src*, como no código a seguir:

```
<%@ Register TagPrefix="uc1" TagName="saudacao"
  Src="saudacao.ascx" %>
```



**Figura 2.** User Control no formulário Web



Os atributos *TagPrefix* e *TagName* servem para definir um *namespace* e um nome para o *User Control*, respectivamente.

Ex.: `<uc1:saudacao></uc1:saudacao>`.

## Web Custom Controls

Apesar de podermos criar a maioria das soluções com *User Controls*, há situações em que é necessário que o código do controle fique “oculto” ou que o controle seja mais complexo e robusto. Para esses e outros casos, criaremos *Web Custom Controls*, que são encapsulados em *assemblies* (DLLs compiladas). *WebControl*, classe derivada de *System.Web.UI.Control* e que serve de base para todos os *Custom Controls* que você criar, define as características básicas de um controle. Para ilustrarmos a criação de um controle desse tipo, vamos portar o controle *saudacao.ascx* para *saudacao.dll*.

No Delphi, vá até o menu *File>New>Other* e na seção *Delphi for .NET Projects*, escolha a opção *Web Control Library*. Salve a unit do controle como “ClubeDelphi.pas” e o projeto como “ClubeDelphi-Controls.bdsproj”.

Nosso controle poderá gerar a data abreviada e por extenso. Vamos então definir um tipo enumerado, que expressa a escolha do desenvolvedor, adicionando o seguinte código:

```
type
  TipoData = (tdExtenso, tdAbreviada);
  ...
```

Agora observe o seguinte bloco de código:

```
[DefaultProperty('Text'),
  ToolboxData('<{0}:MyWebControl1
runat=server></{0}:MyWebControl1>')]
MyWebControl1 =
  class(System.Web.UI.WebControls.WebControl)
```

*DefaultProperty* é a propriedade padrão do *Web Custom Control*, *ToolboxData* especifica como será a construção da *tag* do controle, onde *{0}* representa o nome do *namespace* e *MyWebControl1* é o nome da classe do controle que deriva da superclasse *System.Web.WebControls.WebControl*.

Altere os dados do código anterior como demonstrado a seguir:

```
[DefaultProperty('Formato'), ToolboxData(
  '<{0}:Saudacao runat=server></{0}:Saudacao>')]
Saudacao =
  class(System.Web.UI.WebControls.WebControl)
```

Agora, iremos declarar os campos privados do controle, com o seguinte código:

```
strict private
  _ManhaText: string;
  _TardeText: string;
  _NoiteText: string;
  _Formato: TipoData;
```

Após termos declarado os campos privados, iremos declarar os

seus métodos de escrita e leitura, no especificador de acesso *strict protected*, conforme a **Listagem 2**. Altere dois métodos criados pelo controle (*Constructor* e *RenderContents*) com o nome correto da nova classe: *Saudacao*, no lugar de *MyWebControl1*.

## Listagem 2. Métodos do Web Custom Control

```
strict protected
  procedure SetManha(Texto: string);
  procedure SetTarde(Texto: string);
  procedure SetNoite(Texto: string);
  procedure SetFormato(Tipo: TipoData);
  function GetManha: string;
  function GetTarde: string;
  function GetNoite: string;
  function GetFormato: TipoData;
```

## Adicionando propriedades

Na seção *published*, vamos declarar as propriedades do controle, conforme o código da **Listagem 3**. Veremos as características de cada propriedade:

- *Manha*: Texto que o componente vai gerar se a hora do sistema for pela manhã;
- *Tarde*: Texto se a hora do sistema for à tarde;
- *Noite*: Texto se a hora do sistema for à noite;
- *FormatoData*: Formato da data a ser *renderizada* pelo controle.

## Listagem 3. Propriedades publicadas do Web Custom Control

```
published
  [Bindable(Ttrue), Category('Saudacao'),
  DefaultValue('')]
  property Manha: string read GetManha
  write SetManha;

  [Bindable(True), Category('Saudacao'),
  DefaultValue('')]
  property Tarde: string read GetTarde
  write SetTarde;

  [Bindable(True), Category('Saudacao'),
  DefaultValue('')]
  property Noite: string read GetNoite
  write Setnoite;

  [Bindable(True), Category('Saudacao'),
  DefaultValue('')]
  property FormatoData: TipoData read GetFormato
  write SetFormato;
```

As declarações entre colchetes antes de cada propriedade são chamadas de atributos, discutidos a seguir:

- *Bindable(True)*: Especifica se a propriedade terá a opção de *binding* de dados;
- *Category(Saudacao)*: Especifica o nome da categoria a qual a propriedade pertencerá no *Object Inspector* do Delphi;
- *DefaultValue*: Define um valor padrão para a propriedade.

Aperte **Ctrl+Shift+C** para que o Delphi crie o cabeçalho dos métodos. Todas as *functions* retornam os dados dos respectivos campos privados. Iremos implementar essas *functions* como no código da **Listagem 4** (utilize o modelo para configurar as demais propriedades). Assim como as *functions*, as *procedures* também possuem a mesma lógica, sendo assim, o código da **Listagem 5** servirá de modelo para as demais.

#### Listagem 4. Implementando as functions do controle

```
function Saudacao.GetFormato: TipoData;
begin
    Result := Formato;
end;

function Saudacao.GetNoite: string;
begin
    Result := NoiteText;
end;
```

#### Listagem 5. Implementando as procedures do controle

```
procedure Saudacao.SetFormato(Tipo: TipoData);
begin
    if Formato <> Tipo then
        _Formato := Tipo;
end;

procedure Saudacao.SetManha(Texto: string);
begin
    if ManhaText <> Texto then
        ManhaText := Texto;
end;
```

Os *ASP.NET Controls* geram código HTML de saída, para serem apresentados em um browser. Sendo assim, cada controle é responsável por *renderizar* o seu próprio HTML. Esse processo de *renderização* é efetuado no método *Render*, que é abstrato e definido na classe *Control* e todos os controles derivados dessa classe implementam esse método.

Na classe do nosso controle, vamos declarar o método *Render* na seção *strict protected*.

```
strict protected
procedure Render(Output: HtmlTextWriter); override;
```

Pressione novamente **Ctrl+Shift+C** e implemente o método com o código da **Listagem 6**.

#### Listagem 6. Implementando o método Render do Web Control

```
{ Declare em uses System.Globalization }
procedure Saudacao.Render(Output: HtmlTextWriter);
var
    ci: CultureInfo;
    oData: DateTime;
begin
    oData := DateTime.Now;
    ci := CultureInfo.Create('pt-BR');
    if oData.Hour < 12 then
    begin
        if FormatoData = tdExtenso then
            Output.Write('<tr><td>' + ManhaText +
                oData.ToString('dddd, d'' de ''MMMM'' de '+
                    ''yyyy', ci) + '</td></tr>');
        else
            Output.Write('<tr><td>' + ManhaText +
                oData.ToString('dd/MM/yyyy', ci) +
                    '</td></tr>');
        end
    else if (oData.Hour >= 12) and
```

```
(oData.Hour < 18) then
begin
    if FormatoData = tdExtenso then
        Output.Write('<tr><td>' + TardeText +
            oData.ToString('dddd, d'' de ''MMMM'' de '+
                ''yyyy', ci) + '</td></tr>');
    else
        Output.Write('<tr><td>' + TardeText +
            oData.ToString('dd/MM/yyyy', ci) + '</td></tr>');
    end
else if oData.Hour >= 18 then
begin
    if FormatoData = tdExtenso then
        Output.Write('<tr><td>' + NoiteText +
            oData.ToString('dddd, d'' de ''MMMM'' de '+
                ''yyyy', ci) + '</td></tr>');
    else
        Output.Write('<tr><td>' + NoiteText +
            oData.ToString('dd/MM/yyyy', ci) + '</td></tr>');
    end;
end;
```

### Entendendo o método Render

No método *Render* do controle, criamos uma série de testes condicionais. Primeiro testamos a hora do sistema, em seguida a opção de formato de data selecionado pelo o usuário e então geramos o HTML contendo o texto de saudação e a data, através do método *Write* do objeto *Output* do tipo *HtmlTextWriter*, passado como parâmetro no método *Render*.

As tags `<tr></tr>` e `<td></td>` são inseridas no código para criar uma linha e uma célula, respectivamente. Instale a DLL através do menu *Component\Installed .NET Components*. Adicione o componente em um *WebForm* e teste suas propriedades (**Figura 3**).



**Figura 3.** Visualização de uma página ASPX utilizando o controle "Saudacao"

### Conclusões

Observe que as técnicas de orientação a objeto utilizadas neste exemplo em .NET não se diferenciam muito do Win32. Criar seus próprios controles é provavelmente a parte mais fácil deste artigo. O grande macete é compreender o que "reutilizar" em suas aplicações Web, criando assim controles realmente úteis. Espero que o exemplo tenha ajudado a explicar pelo menos alguns dos problemas envolvidos no desenvolvimento de controles ASP.NET no Delphi. Bom proveito e sucesso nos projetos com essa tecnologia que se tornou o padrão para desenvolvimento de aplicações para Web com o Delphi 2005. Um abraço a todos e até a próxima!



**Laércio Santos de Queiroz** ([laercio\\_queiroz@hotmail.com](mailto:laercio_queiroz@hotmail.com)) é Web Designer e desenvolvedor especializado em tecnologias Web e multicamadas, além de possuir conhecimentos em ASP, ASP.NET, Java e Delphi.



[www.clubedelphi.net](http://www.clubedelphi.net)

# Delphi 2005 e .NET Compact Framework

Andreano Lanusse

## Desenvolvendo aplicações Delphi para Celulares e Pocket PCs

Há alguns anos vemos os dispositivos móveis tomando conta do nosso dia a dia, de *palms* a celulares. Essa evolução foi absorvida rapidamente pelas empresas, que para se tornarem competitivas, começaram a utilizar essa tecnologia. No entanto, o custo dos aparelhos no mercado brasileiro ainda é muito alto, mas com a expansão, isso tende a diminuir.

As empresas de telefonia celular no mercado de voz atingiram todos os níveis sociais e agora buscam outras formas de incrementar seu faturamento. O tráfego de dados é a “bola da vez”, onde existem inúmeras oportunidades no mercado de dados via telefonia celular.

Sabendo disso, acabamos de disponibilizar suporte ao *Compact Framework Preview* para Delphi 2005. Através deste novo compilador, poderemos desenvolver nossas aplicações Delphi para dispositivos móveis, como Pocket PC's e Smartphones! A comunidade solicitou e a Borland, como sempre, atendeu ao pedido dos desenvolvedores.

Os usuários registrados de Delphi 2005 poderão fazer o download do CF Preview através do link [www.borland.com/downloads/registered/download\\_delphi.html](http://www.borland.com/downloads/registered/download_delphi.html). Neste artigo conheceremos o .NET Compact Framework e veremos como desenvolver nossa primeira aplicação mobile com Delphi 2005.

### .NET Compact Framework (CF)

O .NET CF é o framework para desenvolvimento de aplicações mobile. Tem como base o .NET Framework, porém com recursos bem mais restritos, visto que os *devices* (equipamentos) têm diversas restrições, como veremos muitas delas aqui. O .NET CF também inclui a implementação do CLR (*Common Language Runtime*).

O .NET CF oferece suporte para Pocket PC's, SmartPhone 2003 e Windows Mobile 2003. Os Pocket PC's geralmente têm mais recursos que os SmartPhone. Na **Tabela 1**, temos um comparativo de recursos, onde ambos suportam:

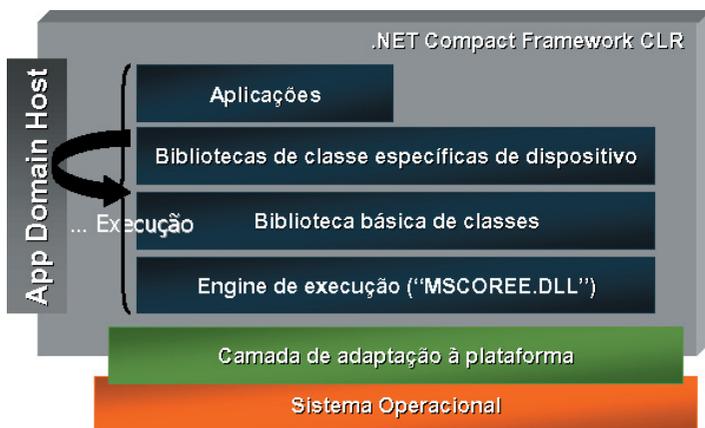
- Recursos Multimídia;
- Pocket Outlook, Pocket Internet Explorer;
- GSM/GPRS ou CDMA Radio;
- SD Card e outros mais.

O CLR do .NET CF é a base para execução das aplicações, assim como o .NET Framework, ele é o “tradutor” entre o sistema operacional do *device* e as aplicações .NET. Na **Figura 1** temos a arquitetura do CLR.

Pocket PC Phone Edition	SmartPhone
Touch Screen	Não há Touch Screen
PDA	Telefone
Gravação em RAM	Gravação em Flash
32 – 128MB RAM, ~200 – 400MHz CPU	16MB RAM, ~125 – 200MHz CPU

Tabela 1. Recursos do Pocket PC e SmartPhone





**Figura 1.** Arquitetura do .NET Compact Framework

O CLR do .NET CF corresponde a cerca de 12% do .NET Framework, impondo com isso limites no desenvolvimento de aplicações para móvel. Nem todos os recursos de uma aplicação desktop podem ser utilizados em mobile, obviamente. Até então as principais linguagens suportadas em .NET CF eram o VB.NET e C#. Agora nossos problemas foram resolvidos: o Delphi é a mais nova linguagem para .NET CF!

O suporte a formulários é feito através do namespace *System.Windows.Forms*, mas temos uma namespace adicional para aumentar os recursos *System.WindowsCE.Forms*. A interoperabilidade COM e *.NET Remoting* não são suportados em .NET CF, mas algumas classes para transmissão de dados via Infrared e outras específicas de cada *device*, como Bluetooth, estão disponíveis.

O desenvolvimento para banco de dados é feito através de ADO.NET. O BDP ainda não está disponível, pois faz acesso nativo. Os *Data Providers* OleDb e ODBC não são suportados pelo .NET CF. Com isso, a utilização de arquivos XML será muito comum em .NET CF, como veremos isso mais adiante.

A **Tabela 2** compara o .NET com o .NET CF, nos ajudando a entender um pouco das limitações que teremos. Veja que essas são limitações impostas pelo .NET CF e *devices*, e não pelo Delphi. Mesmo o .NET CF sendo limitado em relação ao .NET Framework, existem ainda limitações do .NET CF de acordo com o SO, como por exemplo:

- Windows Mobile 2003 - Baseado no SO Windows CE 4.2, inclui o .NET CF na ROM;
- Pocket PC 2003 - .NET CF completo;
- SmartPhone 2003 - Maioria dos recursos do .NET CF são suportados: controles adaptados ao design do

SmartPhone, suporte ao modelo de navegação do SmartPhone e suporte a navegação através de teclas e eventos.

Por exemplo, em um SmartPhone 2003 não é possível utilizar a classe *DataGrid*, responsável por apresentar um Grid na tela.

Os componentes suportados pelo .NET CF sofrem suas variações de acordo com o SO, para diminuir essas limitações você pode utilizar o OpenNetCF ([www.opennetcf.org](http://www.opennetcf.org)) que é um conjunto de componentes que estendem o suporte de componentes para o .NET CF.

## Desenvolvendo para .NET CF com Delphi 2005

Com essa pequena introdução podemos começar a desenvolver nossas aplicações com um mínimo de conhecimento sobre a plataforma. Vamos a partir de agora ver o que o *Delphi 2005 Compact Framework Technology Preview* oferece de recursos.

Apesar dessa ser a primeira versão do nosso compilador para .NET CF, ela não traz restrições do acesso às classes do .NET CF. Vale lembrar que VCL não é suportada e os tipos *TDateTime*, *TDate*, *TTime* e *Currency* ainda não são suportados, mas você pode utilizar os tipos nativos do Framework.

Como os desenvolvedores não podem esperar, já disponibilizamos esse *release*. Ainda não está disponível o design para desenvolvimento de aplicações mobile no Delphi (até o fechamento dessa edição), mas estamos preparando e em breve teremos um design para você “arrastar e soltar” seus componentes em um formulário.

Infelizmente o designer do .NET CF não é aberto (já o design padrão para Windows Forms sim). Portanto, teremos que desenvolver nosso design, nada difícil para quem fez o Delphi, aguardem para ver.

## Instalação e configuração do .NET CF Preview no Delphi 2005

O processo de instalação do .NET CF Preview é muito simples. Será criado a partir do diretório de instalação do seu Delphi o diretório *CFPreview*, que contém os subdiretórios:

- *bin* (compilador);
- *lib* (Bibliotecas);
- *source* (fonte do *Borland.Delphi.System.pas*).

Após a instalação será necessário integrar a compilação ao IDE do Delphi. Isso pode ser feito através do menu *Tools|Configure Tools*. Adicione um *Tool* (botão *Add* na janela *Tool Options*) e configure de acordo com os parâmetros mostrados a seguir e na **Figura 2**:

	File Size			Classes			Methods		
	NETCF	Desktop	%	NETCF	Desktop	%	NETCF	Desktop	%
“MSCorEE”	400K*	2.2 M	18	N/A	N/A	N/A	N/A	N/A	N/A
MSCorLib	200K	2M	10	364	1286	28	3989	13817	29
System	100K	1.2M	8	140	765	18	1090	6953	16
System.Drawing	20K	458K	5	41	254	16	385	3509	11
System.Web.Services	67K	503K	13	54	274	20	302	2083	14
System.Windows.Forms	56+55K	2M	5	43	823	5	393	11337	3
System.XML	138K	1.2M	12	100	724	14	927	7227	13

**Tabela 2.** Restrições do .NET Compact Framework

Title: "Delphi .NET CF Preview Compiler";  
 Program: "..\BDS\3.0\CFPreview\bin\buildCF.bat" (verifique o diretório de instalação do Delphi);  
 Parameters: "\$SAVEALL \$NAMEONLY(\$PROJECT).dpr \$PATH (\$PROJECT)";

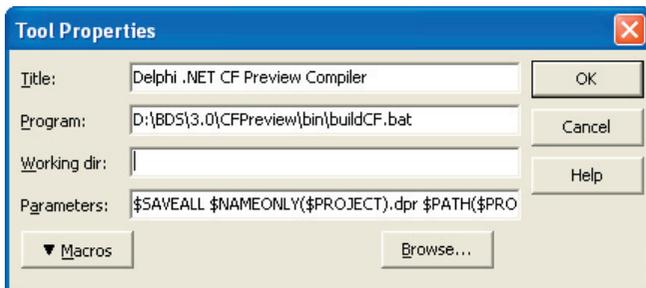


Figura 2. Tela de configuração para compilação do projeto

O arquivo *buildCF* (Listagem 1) é um BAT, que você deve criar, para compilar a aplicação .NET CF.

**Nota:** Verifique se os caminhos que constam no *buildCF* são os mesmos dos diretórios de instalação do Delphi e do Preview do .NET CF.

### Listagem 1. Arquivo BAT para compilação em .NET CF

```
@echo off
rem *****
rem ** Bat usado para integrar ao Delphi 2005
rem ** 2 parâmetros são necessários para compilar o projeto
rem ** %1 - Nome do arquivo de projeto.dpr
rem ** %2 - Diretório do projeto
rem *****
rem
set _DCCILPATH="..\BDS\3.0\CFPreview\bin\dccil.exe"
set _CFUNITS="..\BDS\3.0\CFPreview\lib"
set _PROJECTNAME=%1
set _PROJECTDIR=%2

rem *****
rem ** Certifique que o diretório Bin do Delphi 2005 está no Path
rem *****
set _BDSDIR="..\BDS\3.0\Bin"
path %_BDSDIR%;%PATH%

SHIFT
SHIFT
:LOOP
IF "%1" == "" GOTO END
set _PROJECTDIR=%_PROJECTDIR% %1
SHIFT
GOTO LOOP
:END

cd %_PROJECTDIR%
del *.dc?il

%_DCCILPATH% "%_PROJECTDIR%\%_PROJECTNAME%" -u%_CFUNITS% -luSystem.Windows.Forms -luSystem.Data
Pause
```

## Instalação e Configuração do Emulador

Para testar a aplicação podemos fazer uso de um emulador. Em nosso exemplo utilizaremos o *Microsoft Windows CE 5.0 Device Emulator* para testar nossa aplicação. O download está disponível em [www.microsoft.com/downloads/details.aspx?familyid=A120E012-CA31-4BE9-A3BF-B9BF4F64CE72&displaylang=en](http://www.microsoft.com/downloads/details.aspx?familyid=A120E012-CA31-4BE9-A3BF-B9BF4F64CE72&displaylang=en).

Depois de instalado o emulador, o mesmo estará disponível no menu *Iniciar>Programas>Microsoft Windows CE 5.0>CE 5.0 Emulator>Sample CE Device*. Com o ambiente configurado podemos começar a desenvolver nossa aplicação (Figura 3).

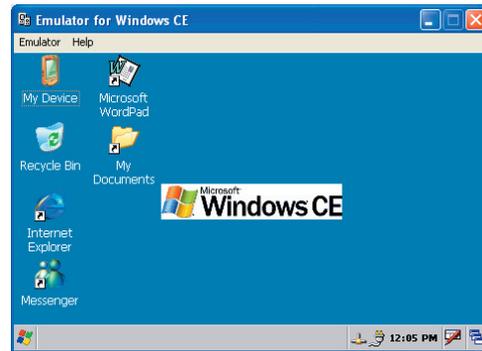


Figura 3. Emulador do Windows CE 5.0

**NC SOFTWARE**  
 TREINAMENTO EM INFORMÁTICA

**O momento de investir em sua carreira profissional é esse!**

- ✓ FORMAÇÃO CURRÍCULO JAVA - (120 hs)
- ✓ FORMAÇÃO DBA ORACLE 9i - COMPLETO - (250hs)
- ✓ FORMAÇÃO ORACLE DEVELOPER 9i - (140hs)
- ✓ FORMAÇÃO SQL SERVER - (210hs)
- ✓ FORMAÇÃO ADMINISTRADOR DE REDES - (140hs)
- ✓ FORMAÇÃO GESTÃO DE PROJETOS EM TI - (160hs)

No fechamento de qualquer formação você ganha 01 assinatura de uma dessas revistas

Para maiores informações:  
 TEL/FAX: (11) 3541-3370 / 3541-3592 / 3541-2743  
[www.ncsoftware.com.br](http://www.ncsoftware.com.br)

## Desenvolvendo aplicações em Delphi 2005

O desenvolvimento de aplicações .NET CF toma como base uma aplicação Windows Forms. Como mencionei anteriormente, não temos o design ainda, mas podemos utilizar o design de uma aplicação Windows Forms normal para adiantar nosso trabalho.

Em função de algumas diferenças no desenvolvimento .NET CF e .NET, e como estamos usando ainda um "preview", teremos que excluir algumas propriedades e métodos acessadas em nosso código. Essa aplicação é bem simples, mas mostrará como você poderá desenvolver para .NET CF preservando seu conhecimento.

No Delphi 2005, siga os seguintes passos para criar a aplicação:

1. Crie uma nova aplicação *Windows Forms - Delphi for .NET*;
2. No *Text* do formulário digite: "Delphi for .NET Compact Framework";
3. Inclua um *Label* no formulário com o texto (propriedade *Text*) "Olá ClubeDelphi";
4. Inclua um *TextBox* e limpe a propriedade *Text*;
5. Inclua um *Button* e preencha a propriedade *Text* com: "ShowMessage";
6. No evento *Click* do botão insira o seguinte código:

```
MessageBox.Show(TextBox1.Text);
```

7. Salve e compile o projeto;
8. Abra o código do formulário (menu *View>Show Code*);
9. Abra o grupo (*Region*) *Windows Form Designer generated code*;
10. Comente a linhas mostradas na **Listagem 2**.

### Listagem 2. Linhas de código que devem ser comentadas

```
Self.SuspendLayout;  
Self.Label1.Name := 'Label1';  
Self.Label1.TabIndex := 0;  
Self.TextBox1.Name := 'TextBox1';  
Self.TextBox1.TabIndex := 1;  
Self.Button1.Name := 'Button1';  
Self.Button1.TabIndex := 2;  
Self.AutoScaleBaseSize :=  
    System.Drawing.Size.Create(5, 13);  
Self.Name := 'TWinForm';  
Self.ResumeLayout(False);
```

**Nota:** É necessário comentar essas linhas porque algumas propriedades e métodos não são suportados no .NET CF. Na documentação do Microsoft .NET SDK você pode ver as propriedades e métodos suportados por cada classe do .NET CF.

11. Selecione o menu *Project|View Source*;
12. Encontre a linha [*STAThread*] e exclua;
13. Salve o projeto;
14. Exclua o arquivo CFG do projeto (que está no mesmo diretório onde você salvou a aplicação);

15. Acesse a opção de compilação criada anteriormente (*CF Preview*), para que o executável do .NET CF seja criado.

Será necessário fazer o *deploy* dessa aplicação no emulador. Execute o emulador e acesse o menu *Emulator>Folder Sharing*, informe o diretório onde está o executável. Geralmente os *devices* têm uma unidade externa de armazenamento (o *Storage Card*).

Configurado o *Folder Sharing* estamos dizendo ao emulador que nossa unidade de armazenamento é o diretório da aplicação. Clique em *My Device>Storage Card* e execute o arquivo *Project1.exe* e a sua primeira aplicação em Delphi 2005 para .NET CF está pronta! Ao preencher o *TextBox* e clicar no botão o resultado será exibido (**Figura 4**).

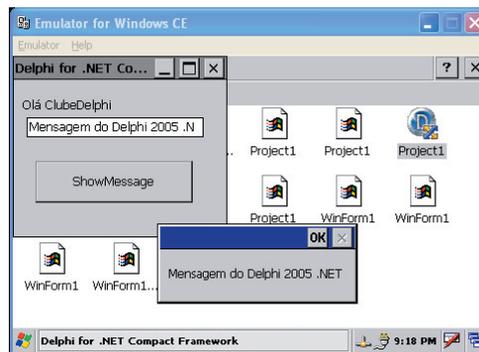


Figura 4. Aplicação Delphi 2005 rodando no emulador

## Desenvolvendo para bancos de dados

E como trabalhar com banco de dados em aplicações móveis? Essa deve ser a pergunta que está sem resposta nesse momento em seu pensamento, então vou explicar algumas formas de acesso a banco de dados. Lembre-se que todos os métodos apresentados têm suas vantagens e desvantagens, o tipo de aplicação que será desenvolvida vai nos ajudar a definir.

Podemos trabalhar com informações atualizadas a cada x período. Essas informações podem estar armazenadas em XML no celular, através do *DataSet* do .NET, onde você poderá fazer a leitura desse arquivo. Na **Listagem 3** temos um exemplo de um arquivo XML gerado por um *DataSet*. A leitura do arquivo XML pode ser realizada conforme o código da **Listagem 4**. Para gravação do XML, troque o *ReadXml* pelo método *WriteXml*.

### Listagem 3. Arquivo XML usado com DataSets

```
<?xml version="1.0" standalone="yes"?>  
<Clientes>  
  <Cliente>  
    <Status>1</Status>  
    <ClienteID>1</ClienteID>  
    <Nome>Delphi 2005</Nome>  
  </Cliente>  
  <Cliente>  
    <Status>1</Status>  
    <ClienteID>2</ClienteID>  
    <Nome>JBuilder</Nome>  
  </Cliente>  
  <Cliente>  
    <Status>1</Status>  
    <ClienteID>3</ClienteID>  
    <Nome>Borland Enterprise Server</Nome>  
  </Cliente>  
</Clientes>
```



#### Listagem 4. Usando DataSets e XML

```

procedure TForm1.TwinForm_Load(
  sender: System.Object; e: System.EventArgs);
var
  DS: DataSet;
begin
  DS := DataSet.Create();
  // Coloque o arq. XML no dir raiz
  DS.ReadXml('\Dados.xml');
end;

```

Outra forma de obter dados é acessando um Web Service e através de métodos enviar e receber *DataSets*. O cuidado com o tamanho do *DataSet* é fundamental nessa situação. Enviar apenas registros alterados pode ser uma boa estratégia para não ter problemas de performance.

E para os usuários de SQL Server, utilizar o *assembly SQLServerCE* pode ser uma solução interessante. Lembre-se que a aplicação ficará presa ao SQL Server, assim trabalhar com Web Services e *DataSet* dará uma maior flexibilidade à aplicação.

#### Conclusões

Este artigo foi o primeiro de vários que virão sobre desenvolvimento mobile com Delphi 2005 e .NET Compact Framework. O exemplo mostrado foi simples, pois acredito que é assim que devemos começar. Os próximos artigos irão focar a necessidade e maior complexidade no desenvolvimento, sempre visando a necessidade dos desenvolvedores. Muitos outros exemplos estão sendo disponibilizados no *CodeCentral* ([cc.borland.com](http://cc.borland.com)) e em meu Blog na área *Compact Framework*, claro que todos em Delphi.

#### Links

##### **Blogs.borland.com/andreanolanusse**

Esse é o link para o meu blog onde disponibilizo diversas informações, principalmente Delphi.

##### **Blogs.borland.com**

Blog dos funcionários Borland, muitos artigos e idéias interessantes.

##### **www.sqlmagazine.com.br**

Esse é meu blog onde publico informações específicas sobre banco de dados, principalmente InterBase.

\* .NETCF tamanho do MSComEE baseado na versão Win32/x86



# Borland® E-Commerce

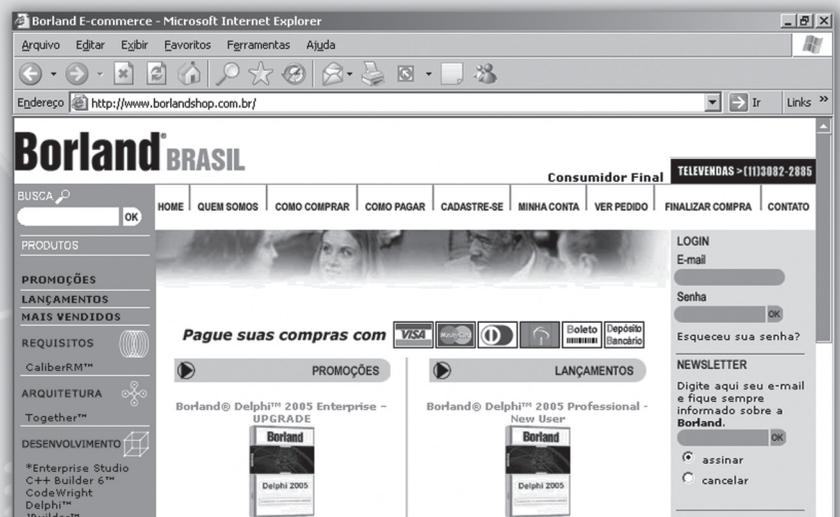
## www.borlandshop.com.br

O seu canal direto de compras para produtos Borland

Mais Fácil

Mais Seguro

Mais Econômico



## Obtenha diagramas de classes por meio de engenharia reversa

Hoje estamos vivendo uma revolução nas áreas relacionadas ao desenvolvimento de software. Essa revolução está acontecendo em virtude do amadurecimento da pesquisa e das técnicas relacionadas à Engenharia de Software.

Há muito tempo a única preocupação das empresas era concluir o software dentro dos prazos estabelecidos (ou dentro dos limites aceitáveis). Contudo, muitas empresas estão utilizando diversas técnicas oferecidas pela Engenharia de Software para auxiliar a concepção e, sobretudo, na construção de softwares com índices maiores de qualidade e produtividade.

A UML (*Unified Modeling Language*) é uma notação padronizada para a especificação e modelagem de softwares Orientados a Objetos amplamente aceita pelo mercado. Fundamentalmente, a UML oferece um conjunto de diagramas cujo objetivo é representar graficamente os diversos elementos de um software orientado a objetos. Aqui, interessa-nos particularmente um deles: o *Diagrama de Classes*.

O Diagrama de Classes tem a finalidade de especificar o “esqueleto” de um sistema orientado a objetos. Nesse diagrama, são apresentadas as classes e os seus respectivos relacionamentos: Associação, Agregação, Composição, Generalização e Especialização. No entanto, ao contrário do que possa parecer, você poderá obter facilmente diagramas de classes a partir dos arquivos-fonte do seu projeto Delphi.

### ESS-Model - Engenharia reversa

Para realizar tal tarefa, você poderá utilizar o ESS-Model, ferramenta *Open Source* escrita em Delphi, cujo objetivo principal é obter diagramas de classes por meio de engenharia reversa a partir dos fontes. Convém lembrar, aliás, que o ESS-Model faz engenharia reversa de fontes em Delphi, Kylix (\*.dpr, \*.pas) e Java (\*.class, \*.java).

Você poderá fazer o download dessa ferramenta no seguinte endereço [essmodel.sourceforge.net](http://essmodel.sourceforge.net). Por se tratar de uma ferramenta extremamente simples, o ESS-Model não precisará ser instalado. Você deverá descompactar o arquivo em qualquer pasta e pronto, a ferramenta estará disponível para uso.

Nessa condição, tendo como ponto de partida uma suposta classe chamada *TPessoa* e duas outras classes especializadas *TPessoaFisica* e *TPessoaJuridica*, vamos analisar por meio de exemplos práticos as principais funcionalidades dessa ferramenta. A **Figura 1** apresenta

um diagrama de classes com a representação do relacionamento de especialização (Herança) entre as classes citadas anteriormente. Note que você poderá selecionar quais atributos ou operações serão exibidos conforme o tipo de visibilidade (*Filter*).

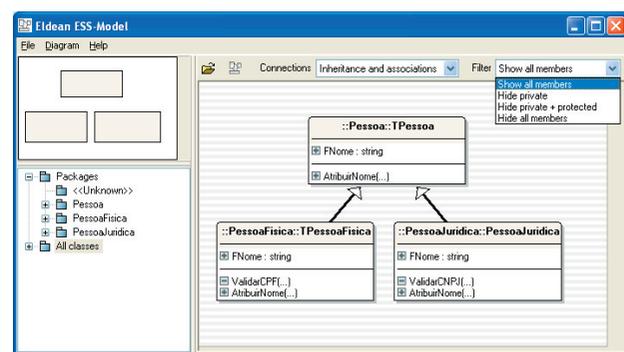


Figura 1. Diagrama de classes

### Recursos visuais

O ESS-Model utiliza alguns recursos visuais a fim de facilitar a leitura e o entendimento dos diagramas de classes, como pode ser visto na **Tabela 1**. Além disso, você poderá visualizar e navegar entre os *packages* e classes de um projeto, através de uma árvore hierárquica.

Aparência	Significado
Fonte itálica	Abstract
Fonte verde	Constructor
Fonte vermelha	Destructor
Fonte preta	Procedure
Fonte cinza	Function
Sinal (+)	Visibilidade Public
Sinal (-)	Visibilidade Private
Sinal (#)	Visibilidade Protected

Tabela 1. Significado dos recursos visuais

Você poderá obter o diagrama de classes de uma unit isoladamente ou, ainda, um projeto inteiro. Você poderá reorganizar as classes utilizando um recurso chamado *Layout*. O *Layout* realiza a reorganização das classes de um diagrama automaticamente.

## Intercâmbio com outros aplicativos

É preciso lembrar que, o propósito do ESS-Model é a obtenção de diagramas de classes por meio de engenharia reversa a partir dos fontes de um projeto, sem contudo, oferecer recursos para a manipulação desses diagramas (exceto visual).

Nessa condição, para contornar esse problema, você poderá exportar o diagrama para a área de transferência e manipulá-lo em algum documento ou relatório; se for necessário, você também poderá salvar o diagrama como um arquivo no formato PNG ou WMF. Em ambos os casos, os diagramas exportados serão imagens estáticas.

Você poderá exportar os diagramas de classes para um arquivo XML no formato XMI (*Metadata Interchange*). Basicamente, o arquivo XMI contém informações sobre as classes e os seus relacionamentos, exceto as suas representações gráficas.

Segundo orientações encontradas no manual do ESS-Model, esse arquivo deverá ser importado pelo *ArgoUml*, ferramenta *Open Source* para projetar sistemas utilizando a notação UML, maiores informações sobre o *ArgoUml* poderão ser obtidas no seguinte endereço: [argouml.tigris.org](http://argouml.tigris.org).

## Relatórios em HTML

Assim como as principais ferramentas disponíveis no mercado, o ESS-Model também realiza a exportação dos seus diagramas para um relatório padronizado no formato HTML. Os relatórios são extremamente sofisticados e profissionais; as classes são apresentadas hierarquicamente com base nos seus tipos de relacionamento.

Você poderá navegar de maneira intuitiva entre as classes, visualizando os seus atributos e operações, conforme apresentado na **Figura 2**. Entretanto, para utilizar esse recurso, torna-se necessário a instalação do parser XML da Microsoft, versão 4.0 ou superior. Normalmente, essa biblioteca é instalada junto com o Internet Explorer 5.0 ou superior, contudo, você poderá fazer o download dessa biblioteca no seguinte endereço: [msdn.microsoft.com/library/default.asp?url=/downloads/list/xmlgeneral.asp](http://msdn.microsoft.com/library/default.asp?url=/downloads/list/xmlgeneral.asp).

The screenshot shows a web interface for a UML class diagram. On the left, there is a 'Package overview' section with a table listing packages: 'Pessoa', 'PessoaFisica', and 'PessoaJuridica', each with a 'Diagram' link. The main content area is titled 'Package: PessoaJuridica' and shows details for the 'PessoaJuridica' class. It indicates it inherits from 'TPessoa' and lists its attributes: 'FNome' (string, public). Below the attributes, there is a table for 'Operations' with columns for Name, Parameters, Returns, and Visibility. One operation is listed: 'ValidarCNPJ' with parameters 'AtribuirNome' (string FNome) and visibility 'public'.

Figura 2. Exemplo de relatório em HTML

## Linha de comando

Você também poderá utilizar o ESS-Model na linha de comando ou, se for necessário, executá-lo diretamente a partir da sua aplicação. Para tal tarefa, basta observar atentamente a sintaxe apresentada a seguir:

```
essmodel.exe [opções] [@lista-arquivos] [arquivo]
```

Onde:

[*opções*]: são as opções globais aceitas pelo ESS-Model (veja a **Tabela 2**);

[*@lista-arquivos*]: arquivo contendo a listagem dos arquivos que serão lidos pelo ESS-Model;

[*arquivo*]: arquivo que será lido pelo ESS-Model;

Na **Tabela 2** temos mais algumas opções.

Opção	Significado
-help	Exibe a ajuda
-d [caminho]	Gera a documentação no caminho especificado
-a [+/-]	Exibe as associações no diagrama gerado (Sim/Não)
-v [0-3]	Filtra a visibilidade dos atributos e operações das classes no digrama gerado. 0=Exibir todas; 3=Esconder todas
-x [arquivo]	Exporta o diagrama para um arquivo no formato XML

Tabela 2. Opções globais do ESS-Model

## Integração com o Windows e o Delphi

Um outro recurso interessante fornecido pelo ESS-Model é a integração nativa com o Windows Explorer e a IDE do Delphi. Você poderá configurar o ESS-Model para inserir no menu de contexto do Windows Explorer ou do menu *Tools* da IDE do Delphi, uma nova opção com o propósito de obter o diagrama de classes da unit ou do projeto que você estiver trabalhando atualmente na IDE.

## Conclusões

O ESS-Model é, em virtude da sua proposta simples, uma ferramenta extremamente poderosa e flexível para obtenção de diagramas de classes (UML Standard). Os seus recursos são objetivos e práticos, sem, contudo, abrir mão do uso de algumas idéias sofisticadas (relatórios em HTML, integração com o Windows Explorer e a IDE do Delphi).



**Cristiano Caetano** ([caetano\\_leite@dell.com](mailto:caetano_leite@dell.com)) É autor do livro *CVS: Controle de Versões e Desenvolvimento Colaborativo de Software*, atualmente exerce a função de Test Leader do Global Development Services da DELL.

Quer saber mais sobre os 10 anos do Delphi?

Visite o hot site da Borland

[www.devmedia.com.br/clubedelphi/borland](http://www.devmedia.com.br/clubedelphi/borland)

**Borland**®

# Borland Conference 4

Amandio Aguiar

## eXtreme Performance

Caros Borlanders, este mês tenho o prazer de anunciar o MAIOR evento Borland do ano! Não é somente o maior evento em tamanho, como também o mais importante de todos os que realizamos, pois teremos a oportunidade de estar em contato com a comunidade Borland do país.

Estaremos, pela quarta vez consecutiva, reunidos com nossos amigos e companheiros de desenvolvimento, conversando sobre o futuro das tecnologias, as novidades que vão permitir criar softwares ainda melhores, em menos tempo, e vislumbrar o que vai acontecer com o mercado em 2006.

### O Evento

O tema do evento esse ano é *eXtreme Performance*, e não escolhemos esse tema à toa. O objetivo é agregar performance extrema ao desenvolvimento de aplicações. Isso se faz com planejamento, processo, metodologia, qualidade, equipe cooperativa, sincronismo, informação, um pouco de tudo que queremos mostrar para vocês.

E haja performance extrema por parte de vocês também! Serão três dias inteiros de pura adrenalina, com várias *tracks* paralelas, destinadas a desenvolvedores, arquitetos, analistas de negócio, analistas de qualidade, coordenadores e gerentes de TI, com palestras, demos e mini-cursos.

Já participou da *Borland Conference*? Prepare-se para se surpreender. O formato do evento esse ano foi totalmente alterado para que você possa aproveitar ainda mais.

Agora, independentes da feira COMDEX, pudemos montar uma estrutura melhor, com total liberdade para que nossa única preocupação fosse adequar o evento às necessidades dos participantes. Manteremos nossa área de exposição, com a presença de empresas com produtos e serviços desenvolvidos para vocês que trarão novidades e soluções para o seu dia a dia.

### O Congresso

Você terá a sua disposição as informações para agregar performance extrema aos seus projetos de ponta a ponta: dos requisitos ao *deploy* da aplicação, passando pela modelagem, testes, muito código e gerenciamento desse ciclo + metodologias, processos e certificações de qualidade.

Muitas e muitas horas técnicas, divididas em palestras, demos, casos de sucesso e várias atividades. Atendendo a pedidos, teremos ainda "mini-cursos", levando soluções para suas necessidades específicas. Com os mini-cursos preparados, vocês terão a oportunidade de aprender o passo a passo com alguns dos craques da Borland Brasil e da comunidade.

Teremos a presença de palestrantes e *keynotes* não só do Brasil, mas importados de nossa matriz! Quem vem? A verdade é que um monte de feras se ofereceu (viva à comunidade brasileira e à caipirinha!). Acompanhem as confirmações em nosso site: [borcon.borland.com.br](http://borcon.borland.com.br).

### Macro-temas Borland Conference 2005

- *Metodologias de Desenvolvimento*: XP, FDD e UP;
- *Delphi*: ASP.NET, VCL, VCL.NET, ECO II e Win32;
- *Java*: JSF, Spring, Refactoring e Segurança;
- *InterBase*: Segurança, Procedures e Eventos;
- *Together*: UML, Design Patterns, Métricas e Auditoria;
- *StarTeam*: SCM na prática;
- *CaliberRM*: Desenvolvimento e Gerência de Requisitos;
- Casos de Sucesso.

Grade completa? Também no site: [borcon.borland.com.br](http://borcon.borland.com.br).

### Quanto custa

Agora sim! Atendendo mais uma vez a pedidos conseguimos melhorar o custo, e bastante! Além da redução dos preços, conseguimos melhorar o nível do material, melhorar nossas instalações e incluir o almoço. Isso mesmo: o almoço é por nossa conta!

Por falar em *cash*, teremos nossas já tradicionais promoções Borland. Você terá a oportunidade de adquirir nossos produtos com condições comerciais especiais.

### Quero ir, e agora?

As inscrições estão abertas! Basta acessar [borcon.borland.com.br](http://borcon.borland.com.br) e vocês terão acesso à tabela de descontos progressivos, ou seja, é melhor se apressar.

"Xii, mas eu trabalho Amandio, não poderei ir." Nosso evento será realizado em uma quinta, sexta e sábado. No sábado também haverá uma super grade de palestras/mini-cursos e demos e você pode fazer a inscrição para um dia apenas.

### Mas não é só isso...

Vamos lá! Quero ver o pessoal entrar no clima RADical do evento, se divertir junto com a gente nas atividades que preparamos, se atualizar muito e ainda concorrer a super prêmios. Quero ser Tetracampeão no quesito evento de sucesso e para isso, preciso de vocês! Chegando lá, me procurem: quero conhecer e agradecer pessoalmente a cada um que comparecer! Vejo vocês no BorCon!



**Amandio Aguiar** ([amandio.aguiar@borland.com](mailto:amandio.aguiar@borland.com)) é Gerente de Produto Delphi da Borland.



A líder em hospedagem de sites  
atua em tantas áreas de tecnologia que  
tem gente que pensa que ela é multi.

A LocalWeb, líder em hospedagem de sites, mais uma vez comprova sua postura de inovação e qualidade, conquistando novos prêmios em 2005: Melhor Empresa de Hospedagem do Brasil e Marca Mais Confiável em Hosting, entre outros. E não é para menos. Além de hospedagem de sites, bancos de dados e comércio eletrônico, que todos já conhecem, somente neste ano a LocalWeb lançou:

- Revendas Linux e Windows
- Transmissão de áudio e vídeo (Windows Streaming Media e Flash Communication Server)
- Servidores HotSite para sites de promoções
- Banco de dados Oracle 10g Compartilhado
- Acesso Speedy para clientes
- Cluster, Storage e Load Balance
- Link Dedicado

E as novidades não param por aí. Afinal, a LocalWeb é mesmo multi.  
Multi em serviços de internet.

**LOCALWEB**  
SERVIÇOS DE INTERNET

[www.localweb.com.br](http://www.localweb.com.br)





# Borland Conference 4

MKT - Borland

## Sua Trilha na Criação Otimizada de Software

**EXtreme Performance - 17 a 19 de novembro - São Paulo**

A Borland Conference Brasil 4 está chegando.

Todas as novidades sobre processos, metodologias, gestão de riscos, requisitos e mudanças, modelagem, código, qualidade e certificações, reunidas em 3 dias de pura adrenalina.

**Fazendo sua inscrição agora você garante preços promocionais e mais:**

- Acesso a todas as palestras da Developer Conference, incluindo os keynotes internacionais;
- Material completo do evento (mochila, camiseta e material de apoio);
- Alimentação (almoço com experts Borland e coffee breaks);
- Descontos imperdíveis na aquisição de produtos Borland

Participe, reúna-se à comunidade Borland do país e agregue Performance Extrema aos seus projetos.

<http://borcon.borland.com.br>

**Borland®**

© 2005 Borland Software Corporation. Todos os direitos reservados. Todas as marcas e produtos da Borland são marcas, registradas ou não, da Borland Software Corporation nos EUA e em outros países.

