Tratamento de exceções

Uma das tarefas mais importantes a serem desenvolvidas para que um software tenha qualidade é o tratamento de mensagens, pois com um sistema de mensagens bem elaborado o suporte a Cliente, mesmo que seja via telefone, se torne mais fácil. Pensando nisso eu desenvolvi uma forma simples de traduzir as mensagens de erro recebidas pelo programa, sejam elas vindas do banco de dados ou quaisquer outras mensagens que necessitem um tratamento ou tradução.

Vamos agora ao que interessa, primeiro vamos criar um banco de dados para exemplo, eu normalmente crio um banco de dados com os seguintes parâmetros:

```
Page size: 4096
Character Set: WIN1252
SQL Dialect: 3
Nome: DBTeste.fdb
```

A tabela a ser criada tem a estrutura:

```
CREATE TABLE TESTE(
  ID INTEGER NOT NULL PRIMARY KEY,
  DESCRICAO VARCHAR(50) COLLATE PXW_INTL850
);
```

Após criados o Banco de Dados e a nossa tabela, passemos para a ferramenta onde será desenvolvido o nosso sistema de tratamento de Exceções, para este exemplo será utilizado o Delphi 7 e como suíte de acesso o MDO (www.sourceforge.net/projects/mdo), mas a lógica pode facilmente ser aplicada a outras ferramentas como C++ Builder, Kylix, etc. e a suíte pode ser os componentes do BDE, IBO, IBX, e como banco de dados o Firebird 1.5 RC2, mas podendo ser utilizado qualquer outro BD.

Agora vamos abrir o Delphi e criar o nosso novo projeto. No formulário principal vou colocar os componentes abaixo:

- ✓ TMDODataBase
  - AllowStreamedConnected: False;
  - DataBaseName: <caminho>\DBTeste.fdb;
  - LoginPrompt: False;
  - Name: mdbPrincipal;
  - Params:
    - user_name=sysdba
    - password=masterkey
    - lc_ctype=win1252
- ✓ TMDOTransaction
  - DefaultDatabase: mdbPrincipal;
  - Name: mdtPrincipal;
  - Params:
    - read_committed
    - rec_version
    - nowait
- ✓ TMDODataSet:
  - Database: mdbPrincipal;
  - DeleteSQL: DELETE FROM TESTE WHERE ID = :OLD_ID;
  - ForcedRefresh: True;

Julio Cesar
julio_gyn@yahoo.com.br

- InsertSQL: INSERT INTO TESTE (ID, DESCRICAO) VALUES (:ID, :DESCRICAO);
- ModifySQL: UPDATE TESTE SET ID = :ID, DESCRICAO = :DESCRICAO WHERE ID = :OLD_ID
- Name: mddTeste;
- RefreshSQL: SELECT ID, DESCRICAO FROM TESTE WHERE ID = :ID;
- SelectSQL: SELECT ID, DESCRICAO FROM TESTE;
- Transaction: mdtPrincipal;

✓ TDataSource
- DataSet: mddTeste;
- Name: dsTeste;

✓ TDBEdit:
- DataField: ID;
- DataSource: dsTeste;

✓ TLabel:
- Caption: &ID;
- FocusControl: DBEdit1;

✓ TDBEdit:
- DataField: DESCRICAO
- DataSource: dsTeste;

✓ TLabel:
- Caption: &Descrição;
- FocusControl: DBEdit2;

✓ TButton:
- Caption: &Novo;

✓ TButton:
- Caption: &Gravar;

✓ TButton:
- Caption: &Cancelar;

Agora que o nosso formulário principal está completo com os nossos objetos, vamos ao que mais importa, a codificação.

Na seção **private** do nosso formulário crie a declaração da nossa procedure responsável pela tradução da mensagem:

```
procedure TraduzException(Msg: string; DataSet: TMDODataSet);
```

Agora pressione Ctrl + Shift + C para que o Delphi possa criar o esqueleto da implementação da nossa procedure.

Dê um duplo clique no botão Novo para gerar a implementação do método OnClick e coloque o código:

```
  { Coloca o DataSet em modo de Inserção }
  mddTeste.Insert;
```

Dê um duplo clique no botão Gravar e na implementação coloque as seguintes linhas:

```
  try
    { Tenta gravar o registro }
    mddTeste.Post;
    { Executa o Commit para que o banco possa }
```

```
    { gravar fisicamente os registros         }
    mdtPrincipal.CommitRetaining;
  except
    { Caso haja um erro ao tentar gravar o registro  }
    { a mensagem será passada para a nossa procedure }
    on E: Exception do
      TraduzException(E.Message, mddTeste);
  end;
```

Dê um duplo clique no botão Cancelar e coloque a linha:

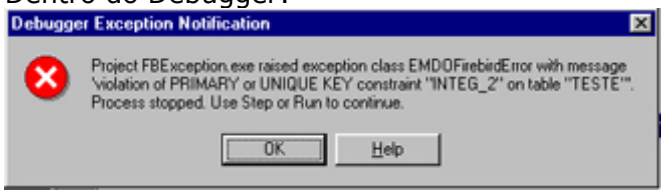
```
  { Executa o cancelamento das alterações }
  mddTeste.Cancel;
```

Por último, porém mais importante no nosso exemplo vamos implementar o nosso tratamento de erros, abaixo segue o código comentado:

Análise das mensagens de erro

As mensagens do Debugger do Delphi tem uma diferença com as mensagens de Runtime

Dentro do Debugger:



Em Runtime:

Note que a mensagem do Debug traz informações importantes para que a Exceção seja tratada da melhor forma, as informações disponíveis incluem a Classe da exceção (*EMDOFirebirdError*), a mensagem de erro (*'violation of PRIMARY or UNIQUE KEY constraint ''INTEG_2'' on table ''TESTE'''*), nos dando assim condições de executar tratamentos específicos para cada tipo de exceção ou tratá-las globalmente.

A implementação do método de tratamento de exceções é bem simples, e pode ser melhorado conforme a necessidade.

```
procedure TFPrincipal.TraduzException(Msg: string; DataSet: TMDODataSet);
var
  { Variável onde ficará a identificação do Campo que ocorreu o erro }
  Campo: string;
  { Variável para armazenar o resultado da busca pela definição do }
  { campo nas definições do DataSet                                 }
  Field: TFieldDef;

  { Procedure para mostrar uma mensagem de Erro passada como parâmetro }
  procedure Erro(Msg: string);
  begin
    MessageBox(Application.Handle, PChar(Msg), 'Erro', MB_OK +
MB_ICONERROR);
```

Julio Cesar
julio_gyn@yahoo.com.br

```
  end;

begin
  { Se o erro for de violação de chave primária }
  if (Pos('violation of PRIMARY or UNIQUE KEY', Msg) > 0) then Erro('O
registro não pode ser gravado, ocorreu um erro de duplicidade na chave
primária!')
    { Se o erro for de um campo NOT NULL com valor nulo }
  else if (Pos('validation error for column', Msg) > 0) and (Pos('***
null ***', Msg) > 0) then
    begin
      { Pega o nome do campo na mensagem de erro }
      Campo := Copy(Msg, 29, pos(',', msg) - 29);
      { Localiza o campo nas definições do DataSet }
      Field := DataSet.FieldDefs.Find(Campo);
      { Se o campo for localizado... }
      if Field <> nil then
        Erro(Format('O registro não pode ser gravado, o Campo ''%s''
precisa ter um valor', [DataSet.FieldByName(Campo).DisplayName]))
        { O campo não foi encontrado... }
      else
        Erro(Format('O registro não pode ser gravado, o Campo ''%s''
precisa ter um valor', [Campo]))
    end
      { Erro ainda não tratado pela procedure }
  else Erro(Msg);
end;
```

Conclusão

Este exemplo foi uma pequena amostra do poder do Delphi/Firebird e um pouco da criatividade que poderá ser empregada para tornar qualquer software mais intuitivo e de fácil suporte, nos sistemas que desenvolvi para a empresa onde trabalho o tratamento de exceções analisa até os valores possíveis a um campo (DEFAULT) que foram colocados na criação do banco e informa o que pode ser usado.

Julio Cesar
julio_gyn@yahoo.com.br