PEX

PEX - Acessar Outros Bancos de Dados via PEX

Neste tópico veremos como conectar em uma base diferente a que o sistema usa via PEX. Usaremos apenas um botão neste exemplo para fazer a conexão.

Código PEX aplicado :

```
var liCodigoConector : integer; // <- Conector de conexão de banco de dados criado no Modulo de Processos -> Conectores, Conector da base que irá se conectar locds : TLibCDS; lbErpOnline : boolean; // < -- Valida se estiver contactado ou não begin liCodigoConector := 16; // <-- Codigo do conectar da base que irá se conectar try try loCDS := of_CriaCDSporSQL('select * from versaodb where 1=2', liCodigoConector ); // select para validar se a conexão foi feita. lbErpOnline := true; aoMensagem.SetStr('MENSAGEM', 'Conectado'); except aoMensagem.SetStr('MENSAGEM', 'Não conectado'); lbErpOnline := false; end; finally loCDS.Free; end; end;
```

PEX - Comunicar com Webservice SOAP ou XML

Se você possui necessidade de comunicar-se com outros softwares e/ou ferramentas, há uma grande chance de que em determinado momento a única possibilidade viável seja a comunicação via WebService.

Conceito básico:

Segundo (W3C, 2004), "Um serviço web é um sistema de software desenvolvido para suportar iterações máquina-máquina interoperáveis sobre uma rede. O serviço web implementa uma interface descrita em um formato que a máquina pode processar, especificamente o WSDL (Web Service Description Language), possibilitando a iteração de outros sistemas utilizando o contrato prescrito no documento WSDL utilizando mensagens SOAP (Simple Object Access Protocol), frequentemente transportadas usando o HTTP (HiperText Transfer Protocol) com serialização XML (Extensible Markup Language). "

Ou seja, WebService permite a comunicação de diferentes softwares por simples transferência de objetos/textos e/ou XML, ou seja, sem contato com a base de dados, somente respondendo a requisições em layout especifico, no nosso caso a comunicação do DOX a arquiteturas que não temos acesso ao banco de dados.

Lembre-se sempre de consultar a documentação do webservice a ser utilizado ou consultar o desenvolvedor do webservice para entender os principais aspectos, exemplos:

- Endereço de comunicação com o WS;
- Endereço de descrição do WS para testes via ferramenta; (Exemplo WSDL)
- Layout padrão de requisição;
- Layout padrão de retorno;
- Possíveis tag e/ou códigos de retorno;
- Possíveis mensagens de retorno; (Neste caso e no anterior, verificar todas as possibilidades para tratamento, exemplo: Erros de comunicação, de layout, registro duplicados, etc).
- Tipo de Webservice; (SOAP / SOA / Outros)
- Tipo de Requisição/Retorno; (JSON / XML / Outros)
- Tipo de Comunicação; (REST / POST / Outros)
- Necessidade de cabeçalho e/ou token;

PEX - Configurar Grade com Campo de Auto Incremento

Hoje nativamente a ferramenta DOX ainda não tem o recurso de contador de linhas de uma grade, para isso foi realizado uma forma de se resolver temporariamente esta questão, pois para alguns essa informação visualmente pode fazer diferença.

Para isso utilizaremos apenas 4 formulários:

1. Grade de dados
2. Botão "Adicionar": com evento PEX ao sair para adicionar linhas na grade acima
3. Formulário dissertativo: para o usuário informar o sequencial que será incrementado
4. Botão "Remover": com evento PEX ao sair para remover as linhas adicionadas na grade.

Conforme imagem do portal:

No formulário de grade de dados desmarque a opção "Permite incluir registros" e "Permite excluir registros". Estas opções é o que permite ao usuário fazer as ações no portal, porém como os botões nativos da grade não executam PEX ao sair, iremos retirar e criarmos os nossos de forma manual.

Na aba "Campos":

Agora, depois de criado o formulário de grade, crie os outros formulários de botão e dissertativa.

Botão "Adicionar". PEX ao sair:

```
const cs_grade = '1'; var liIdRegistro: Integer; loRegistro: TJSONObject; loNovoRegistro : TJSONObject; liSequencia : Integer; begin //incrementa sequencial liSequencia := 1; for liIdRegistro := 0 to Pred(aoFormularios.GetJSON('1').GetArrayJSON('DADOS').Count) do begin loRegistro := aoFormularios.GetJSON('1').GetArrayJSON('DADOS').GetItemAsJson(liIdRegistro); if loRegistro.GetInt('REGISTROS') >= liSequencia then begin liSequencia := loRegistro.GetInt('REGISTROS')+1; end; end; //insere informações na grade loNovoRegistro := TJSONObject.Create; loNovoRegistro.AddPair('REGISTROS', IntToStr(liSequencia)); // Adicione aqui os demais campos que desejar aoFormularios.GetJSON(cs_grade).GetArrayJSON('DADOS').Add(loNovoRegistro); end;
```

PEX - Consultar CEP via Webservice

Neste tópicos veremos como fazer uma requisição do webservice de CEP via PEX. O Webservice que será usado é o da ViaCep, webservice gratuito que busca todos os CEP's do Brasil.

Conceito:

CEP - Código postal ou Código de Endereçamento Postal é um código desenvolvido pelas administrações postais e criado com o intuito de facilitar a organização logística e localização espacial de um endereço postal.

Acessando o site da ViaCep você encontra toda a documentação necessária para realizar as requisições JSON, XML e etc.

Retorno JSON:

https://viacep.com.br/ws/88811578/json <-- Exemplo do retorno


```
{ "cep": "88811-578", "logradouro": "Rua Leandro Martignago", "complemento": "", "bairro": "Pio Corrêa", "localidade": "Criciúma", "uf": "SC", "unidade": "", "ibge": "4204608", "gia": "" }
```

PEX - Consultar CNPJ via Webservice

Neste tópico veremos como fazer uma requisição do webservice de consulta de CNPJ via PEX. O Webservice que será usado é o da ReceitaWS, webservice gratuito que busca informações de CNPJ.

Conceito:

CNPJ - Cadastro Nacional da Pessoa Jurídica, é um número único que identifica uma pessoa jurídica e outros tipos de entidade junto à Receita Federal do Brasil.

Acessando o site da ReceitaWS você encontra toda a documentação necessária para realizar as requisições JSON e etc.

Retorno JSON:

https://www.receitaws.com.br/v1/cnpj/27865757000102 <-- Exemplo do retorno


```
{ "status": "OK", "cnpj": "27865757000102", "tipo": "MATRIZ", "abertura": "03/11/1997",
"nome": "UNIMED DO BRASIL", "fantasia": "", "atividade_principal": [ { "code": "86.50-0-01",
"text": "Atividades de atendimento hospitalar" } ], "atividades_secundarias": [ { "code":
"86.90-9-99", "text": "Outras atividades de atenção à saúde humana não especificadas
anteriormente" } ], "natureza_juridica": "3999", "logradouro": "AVENIDA PAULISTA", "numero":
"1294", "complemento": "14 ANDAR", "bairro": "BELA VISTA", "municipio": "SAO PAULO", "uf":
"SP", "cep": "01310-100", "email": "", "telefone": "(11) 3268-7020", "efr": "", "situacao":
"ATIVA", "data_situacao": "03/11/2005", "motivo_situacao": "", "situacao_especial": "",
"data_situacao_especial": "" }
```

PEX - Criar Arquivo Excel

Criando arquivo Excel:

Iremos focar mais no PEX neste exemplo. O processo contém apenas dois formulários

Grade de dados:

Coluna 1	**"Cliente"**
Coluna 2	"Valor"

Botão: Evento PEX que irá gerar o Excel

A grade será carregada com um SELECT utilizando o evento de Estrutura de repetição - Carregar ao entrar, com o seguintes SQL:

```
SELECT CLIFOREMP.RAZAO CLIENTE, SUM(RECEBER.VALORSALDO) VALOR FROM RECEBER JOIN CLIFOREMP ON (CLIFOREMP.IDCLIFOREMP = RECEBER.IDCLIFOREMP) WHERE ROWNUM <= 15 AND RECEBER.VALORSALDO > 0 GROUP BY CLIFOREMP.RAZAO HAVING SUM(RECEBER.VALORSALDO) > 0 ORDER BY 2 DESC
```

Será necessário o formulário do tipo botão pois é necessário acessar algumas funcionalidades da grade, e na própria grade ao sair não é possível.

Além do formulário do tipo botão é necessário criar um arquivo modelo que o sistema irá se basear para criar o novo arquivo com as informações que a grade estará preenchida, crie uma pasta Temp no C: com esse arquivo modelo.

PEX - Criar Mensagens

Neste tópico veremos como criar as mensagem ou abortar uma atividade de um processo em execução. As mensagens são muito importante para informar operações que estão sendo feitas ou erros que podem ter acontecido.

Assim que escolher a opção que atenda sua necessidade, se for mensagem irá mostrar os seguintes itens que podem ser escolhidos.

- Adicionar mensagem de informação: Pode ser usado quando quer mostrar alguma informação para o usuário:
- Adicionar mensagem de alerta: Usando quando é necessário informar alguma mensagem de alerta para o usuário. Geralmente sua cor muda para dar ênfase em sua mensagem.
- Adicionar mensagem de erro: Usado quando o usuário deve ser alertado de algum erro na execução do processo. Geralmente é acompanhado da função "Abordar operação".
- Definir timeout da mensagem: Com esta opção sendo utilizada nas mensagens, elas irão aparecer e sumir automaticamente de acordo com o tempo definido na configuração.

E a Abortar operação: Geralmente essa opção é utilizada juntamente com a mensagem de erro.

PEX - Definir Variável de Sistema em Formulário

Neste tópico vamos demonstrar como setar o valor de uma variável de sistema em um formulário de processo utilizando PEX.

Exemplo de código

```
const aiAtividadeAtual : Integer; aoFormularios : TJSONObject; const aoVariaveis : TVariaveisEventoFormulario; const aoMensagem : TJSONObject); begin // Seta o valor da variável de sistema USUARIO_LOGADO no campo do formulário aoFormularios.SetStr('NOME_DO_CAMPO', TVariaveisSistema.USUARIO_LOGADO); end;
```

No exemplo acima, o valor da variável de sistema `USUARIO_LOGADO` será atribuído ao campo especificado do formulário.

PEX - Filtrar Dados por Data

Neste tópico iremos mostrar como criar uma grade de dados filtrando os dados pela data inserida em dois formulários.

Iremos retorna os dados via SQL no PEX filtrando a data utilizando dois campos com variáveis do tipo data, e ao clicar o botão a grade será carregada com os dados já filtrados.

Código PEX:

```
const cs_Dtinicio = '1'; cs_Dtfim = '2'; cs_grade = '3'; var locds : TLibCDS; lsSQL : string; liIdRegistro: Integer; loNovoRegistro: TJSONObject; begin lsSQL := 'select usuario,observacao,data,valortotalitens from pedido where data between '+TSTR.Aspa(FormatDateTime('dd.mm.yyyy',aoFormularios.GetJSON('1').GetDt('TEXTO')))+' and '+TSTR.Aspa(FormatDateTime('dd.mm.yyyy',aoFormularios.GetJSON('2').GetDt('TEXTO')))+' and observacao <> ''.'''; loCDS := of_CriaCDSporSQL(lsSQL); try if locds.of_TemDados() then begin loCDS.of_IniciaWhile; try loCDS.of_Primeiro; while not loCDS.of_FimDS do begin loNovoRegistro := TJSONObject.Create; loNovoRegistro // (continuação do código conforme necessidade...) loCDS.of_Proximo; end; finally loCDS.of_FinalizaWhile; end; end; finally loCDS.Free; end; end;
```

PEX - Formatar Número de Telefone

Neste tópico será abordado o tema referente ao evento PEX, com o intuito de formatar o campo TELEFONE, verificando se é um TELEFONE FIXO ou CELULAR, dependendo da quantidade de caracteres digitado.

Para acessar o PEX de um formulário basta marcar a opção Ao Sair e clicar no botão com três pontos.

Segue o código que será aplicado Ao sair - PEX:

//PEX PARA TRATATIVA DE CAMPO FORMATADO COMO TELEFONE FIXO OU CELULAR CONST CS_FONE = '4'; VAR liTamanho : Integer; lsMascara : String; begin liTamanho := length(aoFormularios.GetJSON(CS_FONE).GetStr('TEXTO')); //Conta a quantidade de caracteres digitada. lsMascara := '('+ copy(aoFormularios.GetJSON(CS_FONE).GetStr('TEXTO'),0,2)+')' +copy(aoFormularios.GetJSON(CS_FONE).GetStr('TEXTO'),3,liTamanho-6)+'-' +copy(aoFormularios.GetJSON(CS_FONE).GetStr('TEXTO'),liTamanho-3,liTamanho); //Insere a máscara (parênteses, espaçamento e traço) conforme a quantidade de caracteres digitados. aoFormularios.GetJSON(CS_FONE).SetStr('TEXTO', lsMascara); //Preenche o campo digitado ao sair dele, já com a máscara atribuída. // Esta parte abaixo, formata a mensagem de erro caso a quantidade de caracteres digitados for maior que o limite máximo. if length(aoFormularios.GetJSON(CS_FONE).GetStr('TEXTO')) < 14 then

PEX - Função AbreSite

Neste tópico veremos como utilizar a função AbreSite via PEX para abrir um site externo a partir de um evento no sistema.

Exemplo de código

```
begin AbreSite('https://www.google.com'); end;
```

No exemplo acima, ao executar o evento PEX, será aberto o site informado no navegador padrão do usuário.

PEX - Função LaserBase64

Essa função é utilizada para gerar arquivos físicos de conteúdos base64. Ela recebe dois parâmetros, o conteúdo base64 e o caminho a qual irá ser criado. O caminho deve conter o nome do arquivo e extensão do mesmo como por exemplo: C:\Temp\ArquivoAssinado_*/IDPROCESSO/*.pdf, neste caso ele irá criar o arquivo na pasta Temp com o nome ArquivoAssinado_1.pdf.

Exemplo de código

```
procedure pex_LerBase64(const Conteudo, Caminho : String); var loStream: TBytesStream; begin loStream := TSO.Decode64(Conteudo); try loStream.SaveToFile(Caminho); finally loStream.Free; end; end;
```

Ela pode ser criada pelo desenvolvimento, caso necessário, e utilizar em eventos como PEX - Executar função para criar dinamicamente um ou mais arquivos.

PEX - Incluir e Remover Linhas da Grade

Olá, neste tópico iremos apresentar como gerar e modificar uma grade dinamicamente utilizando PEX. Serão abordados a inserção de novas linhas na grade, remoção de um índice e reorganização dos índices.

Vamos criar um processo de exemplo, contendo uma atividade onde iremos adicionar uma grade de dados, dois botões (Adicionar e Remover) e um campo para informarmos a quantidade (este campo será usado para inserir "x" linhas na grade ou remover a linha de índice "x").

Com o processo e a atividade devidamente criados, vamos criar os formulários, para este exemplo serão necessários 4 formulários:

1. Grade de Dados
2. Botão (Adicionar)
3. Dissertativa (N°/Índice)
4. Botão (Remover)

Com os formulários devidamente criados, iremos configurar seus eventos.

Botão Adicionar, este botão será o responsável por inserir novas linhas na nossa grade de dados, para isso é validados a quantidade de linhas atuais da grade e os índices que serão atribuídos a ela.

```
const cs_grade = '1'; cs_Qtd = '3' ; var liIdRegistro: Integer; loRegistro: TJSONObject;
loNovoRegistro : TJSONObject; liSequencia : Integer; liQtd : Integer; liQtdAtual : Integer;
```

PEX - Preencher Grade com Informações de Formulários

No DOX existe a possibilidade de inserir registros em uma grade de dados por meio de outros campos da mesma atividade. Para fazer isso, utilizamos o recurso PEX ao sair de um botão, que insere os dados digitados em um campo texto na grade de dados.

Clique aqui para baixar o processo deste exemplo

Neste exemplo, há um campo para digitar a chave de acesso de uma nota fiscal eletrônica, um botão para executar a inserção e a grade, onde será gravado as informações de cada nota informada.

Crie os seguintes formulários:

Dissertativa "Chave de acesso" - */CHAVEACESSO/* (Texto)

- Botão "inserir"
- Grade de dados "Notas informadas" - */GRADENOTAS/* (Grade de dados)

No formulário de grade crie as colunas de acordo com a imagem abaixo:

A grade foi configurada com os campos de informações que contém na chave de acesso, incluindo um campo invisível para a chave.

O código PEX foi criado no formulário do tipo Botão. Foi utilizado recurso para percorrer o registro de grade para evitar duplicidade de registros, algumas validações das informações da chave de acesso e o recurso de inserir novo registro em grade.

PEX - Preencher Grade com Informações de um Select

Neste tópico veremos como preencher uma grade com informações vindas de um SELECT, ou seja, várias linhas, utilizando PEX.

Exemplo de código

```
const cs_grade = '1'; // Código do formulário grade var loCDS: TLibCDS; loNovoRegistro: TJSONObject; begin loCDS := of_CriaCDSporSQL('SELECT campo1, campo2 FROM sua_tabela'); try loCDS.of_IniciaWhile; try loCDS.of_Primeiro; while not loCDS.of_FimDS do begin loNovoRegistro := TJSONObject.Create; loNovoRegistro.AddPair('CAMPO1', loCDS.FieldByName('campo1').AsString); loNovoRegistro.AddPair('CAMPO2', loCDS.FieldByName('campo2').AsString); aoFormularios.GetJSON(cs_grade).GetArray('DADOS').Add(loNovoRegistro); loCDS.of_Proximo; end; finally loCDS.of_FinalizaWhile; end; finally loCDS.Free; end; end;
```

No exemplo acima, o método `of_CriaCDSporSQL` executa o comando SELECT e retorna um dataset (`TLibCDS`) com os resultados. Cada linha do resultado é adicionada à grade do formulário.

PEX - Realizar Select

Neste tópico veremos como utilizar o SELECT via PEX para buscar informações em tabelas do banco de dados.

Exemplo de código

```
var loCDS : TLibCDS; begin loCDS := of_CriaCDSporSQL('SELECT * FROM sua_tabela WHERE campo = valor'); try while not loCDS.Eof do begin // Acesse os campos retornados pelo SELECT // Exemplo: loCDS.FieldByName('nome_do_campo').AsString loCDS.Next; end; finally loCDS.Free; end; end;
```

No exemplo acima, o método `of_CriaCDSporSQL` executa o comando SELECT e retorna um dataset (`TLibCDS`) com os resultados, que podem ser percorridos utilizando um loop.

PEX - Remover Caracteres e Manter Somente Números

Abaixo veremos comandos de evento PEX para extrair de qualquer string somente os números, ou seja, poderá atender a diversos casos como exemplo:

- Remoção de caracteres especiais de telefones, mantendo somente os números.
- Remoção de caracteres especiais de CPF/CPNJ, mantendo somente os números.
- Seleção do numero do endereço, em caso de campos onde o número é digitado junto ao nome da rua.
- Tratamento para CEP, Placas e outros mantendo somente os números.

Exemplo de procedure:

```
const aiAtividadeAtual : Integer; aoFormularios : TJSONObject; const aoVariaveis :
TVariaveisEventoFormulario; const aoMensagem : TJSONObject); var Ind : Integer; Result :
String; begin Result := ''; FOR Ind := 1 to Length('TESTE123teste - evento 1 - PeX') do begin
if TSTR.of_TemNumero(Copy('TESTE123teste - evento 1 - PeX',Ind,1)) then begin Result := Result
+ Copy('TESTE123teste - evento 1 - PeX', Ind, 1); end; end; end;
String = 'TESTE123teste - evento 1 - PeX'
Result = '1231'
```

Explicação do código:

Vamos utilizar apenas 2 variáveis, uma para índice do tipo inteiro e outra para o nosso resultado do tipo String.

No primeiro momento, limpamos por garantia a variável Result para começar a receber os dados, após iniciamos nosso FOR estabelecendo que o índice (Ind) começará de 1 ou seja, do primeiro caractere, e percorrerá até o Length da nossa string ou seja, até seu comprimento total (no nosso exemplo = 13).

PEX - Usar Coluna Verdadeiro ou Falso na Grade

Neste tópico veremos como utilizar uma coluna do tipo Verdadeiro/Falso (checkbox) em uma grade de dados via PEX.

Exemplo de código

```
const cs_grade = '1'; // Código do formulário grade var liIdRegistro: Integer; loRegistro: TJSONObject; begin for liIdRegistro := 0 to Pred(aoFormularios.GetJSON(cs_grade).GetArrayJSON('DADOS').Count) do begin loRegistro := aoFormularios.GetJSON(cs_grade).GetArrayJSON('DADOS').GetItemAsJson(liIdRegistro); if loRegistro.GetBool('NOME_DA_COLUNA') then begin // Ação para quando o checkbox estiver marcado (verdadeiro) end else begin // Ação para quando o checkbox estiver desmarcado (falso) end; end; end;
```

No exemplo acima, o código percorre todas as linhas da grade e verifica o valor da coluna do tipo Verdadeiro/Falso (checkbox) para cada linha.

PEX - Utilizar Campo Adicional da FK

Neste tópico iremos adicionar, por meio de uma evento ao sair do campo (PEX), uma informação do campo adicional de uma FK em um formulário dissertativo.

Campo adicional são os demais campos da tabela (ou view) fora o código ou descrição que o modelador deseja utilizar no procedimento. Atualmente o sistema mostrar apenas o ID + DESCRIÇÃO para o usuário na FK, com os campos adicionais as demais informações que a tabela puxa podem ser utilizados no procedimento.

Pode-se utilizar uma tabela sem problemas, mas neste exemplo vamos criar a view colaboradores para a FK.

CREATE OR REPLACE VIEW PUBLIC.VW_COLABORADORES AS SELECT C.IDCLIFOREMP AS ID, C.RAZAO AS NOME, U.IDUSUARIO AS USERID, U.USERNAME, U.IDPAPELFUNCAOPRINCIPAL AS IDPAPELPRINCIPAL, F.DESCRICAO AS BPM_DESCRICAO FROM USUARIO U JOIN CLIFOREMP C ON C.IDCLIFOREMP = U.IDCLIFOREMP JOIN BPM_FUNCAO F ON F.IDBPMFUNCAO = U.IDPAPELFUNCAOPRINCIPAL WHERE U.INATIVO = 'N'::BPCHAR;

Este exemplo foi feito em um banco de dados PostgreSQL. Se necessário, personalize o SQL para o seu respectivo banco.

Crie um conector do tipo FK onde puxa as informações dessa view, de acordo com a imagem abaixo:

Processo de Criação de Formulários

Agora no processo, crie quatro formulários para serem preenchidos de acordo com o campo adicional.

- Colaborador com a variável */COLABORADOR/*
- Username com a variável */USERNAME/*
- Bpm descrição com a variável */BPM_DESCRICAO/*
- Userid com variável */USERID/*

Configure o tamanho dos campos como achar melhor

No primeiro formulário (colaborador) vincule a Fk que criamos (Fk_vw_colaboradores) e salve o processo. Volte para o formulário e vamos criar nosso evento PEX ao sair.

```
const cs_username = '2'; // Aqui nós declaramos as constantes cs_userid = '3'; cs_bpm = '4';
begin // Aqui nós modificamos os formulários do processo
aoFormularios.GetJSON(cs_username).SetStr('TEXTO', aoValoresFK.GetStr('USERNAME'));
aoFormularios.GetJSON(cs_userid).SetInt('TEXTO', aoValoresFK.GetInt('USERID'));
aoFormularios.GetJSON(cs_bpm).SetStr('TEXTO', aoValoresFK.GetStr('BPM_DESCRICAO')); end;
```

Copie esse código ou use o assistente:

Com este PEX, os valores que estão no campo adicional da FK, serão preenchidos nos demais formulários.

PEX - Validação de Campos

Validar Campo Data ao Sair do Formulário

Neste tópico veremos como validar um campo de data ao sair do formulário utilizando PEX.

Exemplo de Código

```
if aoFormularios.GetJSON('ID_DO_CAMPO').GetDt('TEXTO') = 0 then begin
aoMensagem.SetStr('MENSAGEM', 'O campo de data é obrigatório!'); AbortarOperacao; end;
```

No exemplo acima, o código verifica se o campo de data está vazio (igual a 0) e, caso esteja, exibe uma mensagem e aborta a operação.

Validar Campo Somente Leitura

Neste tópico veremos como validar um campo que está como somente leitura, mas que é obrigatório no processo.

Quando um campo é configurado como somente leitura, o usuário não consegue preenchê-lo manualmente, mas pode ser necessário garantir que ele esteja preenchido por algum evento ou processo automático.

Exemplo de código

```
1if aoFormularios.GetJSON('ID_DO_CAMPO').GetStr('TEXTO') = '' then begin
aoMensagem.SetStr('MENSAGEM', 'O campo obrigatório não foi preenchido!'); AbortarOperacao;
end;
```

No exemplo acima, o código verifica se o campo está vazio e, caso esteja, exibe uma mensagem e aborta a operação.

Validar Campos ou Dados via PEX

Este tópico tem a finalidade de abordar os principais validadores presentes no PEX, como bem sabemos, o PEX nas versões maiores que 12 possuem um assistente PEX, no PEX existem também muitas rotinas e/ou funções publicadas que podem ser acessadas via menu de contexto, conforme imagem abaixo:

As funções abordadas serão:

- TSTR.Testa_CNPJ(Codigo);
- TSTR.Testa_CPF(Codigo);
- TSTR.of_IsNotNullEMA(asText);
- TSTR.of_IsNullEMA(asText);
- TSTR.of_VerificaEmail(asTexto, abVerificaNull);
- TSTR.of_TemNumero(s);
- TSTR.of_TemTexto(s, abValida);
- TSTR.of_UFValido(asUF);
- TSTR.of_IsCodigoBarra(Codigo);

Valida CNPJ

Esta função deve ser utilizada com uma variável do PEX, variável do processo e/ou formulário do processo afim de validar um CPNJ, retornando False (Booleana) no caso de digitado um CPNJ inválido por engano ou algo do gênero.

```
var CNPJ : string; Resposta : Boolean; begin CPNJ := '07297774000175'; //CNPOJ Informado em um formulario e/ou variavel do tipo texto Resposta := TSTR.Testa_CNPJ(CNPJ); -- Retorna True ou False para varaivel boleana end;
```