

PEX - Comunicação com WebService - SOAP - XML

Olá,

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 específico, 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;

Abaixo estou disponibilizando código comentado para comunicação via SOAP - XML com software de terceiro, esta comunicação visa de forma simbólica fazer o cadastro de um novo fornecedor e obter como retorno mensagem ou código de erro, ou obter o código do novo fornecedor cadastro no caso de obter sucesso da requisição..

```
#procedure EXEMPLO(const aiAtividadeAtual : Integer; aoFormularios : TJSONObject; const
aoVariaveis : TVariaveisEventoFormulario; const aoMensagem : TJSONObject);
const
  lscaminhoURL := 'http://enderecows.com.br/ema003/consulta/'; //Endereço do Webservice
  lscabrequisicao := 'ema0003'; //Cabeçalho utilizado pelo WS, pode não ser necessário.

var
  cs_retorno,
  lsRequisicao,
  Result,
  lsCPF,
  lsCNPJ,
  lsInsEstado,
  lsEmail,
  SQL: String;
  loRESTClient : TRESTClient;
  loRESTRequest : TRESTRequest;
  loRESTResponse : TRESTResponse;
  loAuthBasica : THTTPBasicAuthenticator;
  loLeitorXML : TLibLeitorXML;

begin

  lsCPF := aoVariaveis.GetStr('/*CPF*/');
  lsCPF := StringReplace(lsCPF, '.', '',[rfReplaceAll, rfIgnoreCase]); //Removido pontos do
CPF já que WS de destino não utiliza.
  lsCPF := StringReplace(lsCPF, '-', '',[rfReplaceAll, rfIgnoreCase]); //Removido traço do
CPF já que WS de destino não utiliza.

  lsCNPJ := aoVariaveis.GetStr('/*CNPJ*/');
  lsCNPJ := StringReplace(lsCNPJ, '.', '',[rfReplaceAll, rfIgnoreCase]); //Removido pontos do
CNPJ já que WS de destino não utiliza.
  lsCNPJ := StringReplace(lsCNPJ, '/', '',[rfReplaceAll, rfIgnoreCase]); //Removido barras do
CNPJ já que WS de destino não utiliza.
  lsCNPJ := StringReplace(lsCNPJ, '-', '',[rfReplaceAll, rfIgnoreCase]); //Removido traço do
```

CNPJ já que WS de destino não utiliza.

```
lsInsEstado := aoVariaveis.GetStr('/*INSCRICAOESTADUAL*/');  
lsInsEstado := StringReplace(lsInsEstado, '.', '',[rfReplaceAll, rfIgnoreCase]); //Removido  
pontos da IE já que WS de destino não utiliza.  
  
lsEmail := aoVariaveis.GetStr('/*EMAIL_CONTATO*/');  
  
if lsEmail = '.' Then  
begin  
    lsEmail := StringReplace(lsEmail, '.', '',[rfReplaceAll, rfIgnoreCase]); // Se e-mail  
igual a vazio ou . por padrão, o ponto é removido.  
end;  
  
if ((lsCNPJ <> '') and (lsCNPJ <> '.')) then  
    lsNome := copy(aoVariaveis.GetStr('/*RAZAOSOCIAL*/'), 0,30); // LIMITANDO O CAMPO RAZAO  
SOCIAL A 30 CARACTERES (LIMITACAO DO WS)  
  
if ((lsCPF <> '') and (lsCPF <> '.')) then  
    lsNome := copy(aoVariaveis.GetStr('/*NOME*/'), 0,30);
```

Em sequencia ao código acima, esta é uma das partes mais importantes do nosso código, montar o corpo do XML para efetuar a requisição:

```
lsRequisicao:= ' <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
xmlns:urn="urn:sap-com:document:sap:soap:functions:mc-style"> ' + chr(13) +  
    ' <soapenv:Header/> ' + chr(13) +  
    ' <soapenv:Body> ' + chr(13) +  
    ' <urn:'+lscabrequisicao+'> ' + chr(13) +  
    ' <IDadosGerais> ' +chr(13) +  
    ' <CodPortal>' +InttoStr(aoVariaveis.GetInt('/*IDPROCESSO*/'))+' </CodPortal> ' + chr(13) +  
    ' <Cpf>' +lsCPF+' </Cpf> ' + chr(13) +  
    ' <Cnpj>' +lsCNPJ+' </Cnpj> ' + chr(13) +  
    ' <InscricaoEstadual>' +lsInsEstado+  
' </InscricaoEstadual> ' + chr(13) +  
    ' <Email>' +lsEmail+' </Email> ' + chr(13)  
+  
    ' </IDadosGerais> ' +chr(13) +  
    ' </urn:'+lscabrequisicao+'> ' + chr(13) +
```

```
'      </soapenv: Body> ' + chr(13) +  
' </soapenv: Envelope>' ;
```

Explicação do código acima:

Dentro da variável lsRequisição estou adicionando uma string com toda a estrutura do XML utilizando os dados obtidos do processo e armazenados variáveis também do tipo string, esses dados podem ter origem em formulários do processo, consultas no banco de dados ou até mesmo serem números e/ou textos fixos.

Perceba que foi criada toda indentação e adicionado ao final de cada linha um ENTER (chr(13)) tornando assim os testes mais fáceis para nós e/ou para terceiros na hora de fazer um "debug" e entender a requisição.

Abaixo sequencia ao código:

```
ExecuteSQL('insert into WS_XML (sequencial,retorno,idprocesso) values ((select  
coalesce((select max(sequencial) from teste),0)+1 from versaodb),' + TSTR.Aspa(lsrequisicao)  
+ ', ' + aoVariaveis.GetStr('/*IDPROCESSO*/')+'))');  
// Utilizado tabela personalizada no banco para guardar as requisições ao WS, utilizado para  
debug, após pleno funcionamento deve ser desabilitado.  
  
Result := '';  
loRESTClient := nil;  
loRESTRequest := nil;  
loRESTResponse := nil;  
  
try  
    loRESTClient := TRESTClient.Create(nil); //Cria Objeto cliet para comunicação  
    loRESTRequest := TRESTRequest.Create(nil); //Cria Objeto para envio de  
Requisição  
    loRESTResponse := TRESTResponse.Create(nil); //Cria Objeto para Obter o Retorno  
    loAuthBasica := THTTPBasicAuthenticator.Create('portal_ema','passEma'); //Autenticação  
- THTTPBasicAuthenticator.Create('USUÁRIO','SENHA');  
  
    loRESTClient.RaiseExceptionOn500 := False; //Torna o status das exceções falso para  
verificar se teremos exceções  
    loRESTClient.BaseURL := lscaminhoURL; //Define caminho(URL) para o Client enviar a  
requisição  
    loRESTClient.ContentType := 'text/xml'; //Define o formato do conteudo (ou seja XML  
neste caso)  
    loRESTClient.Authenticator := loAuthBasica; //Define o tipo de autenticação da nossa
```

```

consulta;

    loRESTRequest.Method := rmPost; //Metodo utilizado na consulta (Ex: rmPost / rmREST
)
    loRestRequest.Resource := ''; //Resource ficará vazio, normalmente não será
necessário;
    loRestRequest.AddBody(lsrequisicao,ctText_xml); //Definição da estrutura da requisição
(Variavel com conteudo XML que alimentamos anteriormente.)
    loRESTRequest.Client := loRESTClient; //Define o cliente da requisição (selecionamos o
client que criamos inicialmente)
    loRESTRequest.Response := loRESTResponse; //Define metodo de retorno

try
    loRESTRequest.Execute; // Executa a requisição
except
    on e : exception do // Trata possiveis exeções
        ShowMessage( e.message ); // Exibe msg em caso de exeção
end;

if TSTR.of_IsNotNullEma(loRESTResponse.JSONText) then // Verifica se retorno é
nulo
    Result := loRESTResponse.JSONText // Se possuir conteudo, atribui a uma
variavel
else
    Result := loRESTResponse.Content;

    ExecuteSQL( ' UPDATE CRM_PROCESSO_VARIAVEL SET VALORATUAL = ' + TSTR.ASPA(Result) + '
WHERE IDVARIAVEL = 97 AND IDPROCESSO = ' + aoVariaveis.GetStr('/*IDPROCESSO*/') );
// Utilizado uma variavel do processo para guardar todo o resultado da consulta ao WS, para
analises, poderá ser desativado quando estiver operando normalmente)

loLeitorXML := TLibLeitorXML.create; // Cria o objeto leitor de XML
try
    loLeitorXML.of_CarregarTextoArquivo(loRESTResponse.Content); //Carrega o arquivo

    lsCodFornecedorSAP := loLeitorXML.of_GetCampoRepeticao('ECodFornecedor', tcrTexto, '',
i); //Extrai a tag ECodFornecedor do XML
    aoVariaveis.SetStr('/*NOVO_COD_SAP_FORN*/', lsCodFornecedorSAP); //Adicionar o

```

conteudo a uma variavel do processo

```
        aoVariaveis.SetStr('/*ERRO_WS*/', loLeitorXML.of_GetCampoRepeticao('ECodRetorno',  
tcrTexto, '', i)); //Atribui código de erro a uma variavel;  
        aoVariaveis.SetStr('/*MSG_WS*/', loLeitorXML.of_GetCampoRepeticao('EMsgRetorno',  
tcrTexto, '', i)); //Atribui possivel mensagem de erro a uma variavel;
```

```
finally  
    TSO.Fan(loLeitorXML); // Finaliza o leitor XML;  
end;
```

```
finally  
//### ATENCAO, PASSO IMPORTANTE PARA GARANTIR DESEMPENHO, CONFIABILIDADE E BOM FUNCIONAMENTO  
DAS CONSULTAS AO WS, SEMPRE LIMPAR/LIBERAR MEMORIA;  
    loAuthBasica.Free; //Limpar componente de autenticação;  
    loRESTClient.Free; //Limpar client  
    loRESTRequest.Free; //Limpar requisição  
    loRESTResponse.Free; //Limpar retorno  
end;  
  
end;
```

**** Lembre-se, o exemplo serve para aprendizado, não deve ser copiado na íntegra para dentro do seu código PEX. Copiar o nome da procedure/PEX por exemplo pode causar problemas difíceis de diagnosticar.

Como citado no tópico é muito importante consultar a documentação do Webservice ou o seu desenvolvedor, e somente após entender o cenário, limitações e recursos disponíveis neste WS iniciar o desenvolvimento, haja vista que cada um deles possui características específicas de acordo com suas tecnologias e recursos;

Revisão #1

Criado 28 March 2022 16:33:06 por Nicolly Andrielly

Atualizado 6 April 2022 09:07:47 por Nicolly Andrielly