Microsoft® BizTalk® Server 2010

BizTalk Server - Princípios básicos dos Mapas

Os mapas, ou transformações, são um dos componentes mais comuns nos processos de integração. Funcionam como tradutores essenciais no desacoplamento entre os diferentes sistemas a interligar. Neste artigo, à medida que exploramos o editor de mapas do BizTalk Server, exploramos os seus principais conceitos enquanto abordamos temas como a arquitectura deste servidor e alguns dos padrões mais usados na tradução de mensagens.

Sandro Pereira

Março 2012
Versão 1.0

Índice

Introdução

Os mapas, ou transformações, são um dos componentes mais comuns nos processos de integração. Funcionam como tradutores essenciais no desacoplamento entre os diferentes sistemas a interligar. Neste artigo, à medida que exploramos o editor de mapas do BizTalk Server, exploramos os seus principais conceitos enquanto abordamos temas como a arquitectura deste servidor e alguns dos padrões mais usados na tradução de mensagens.

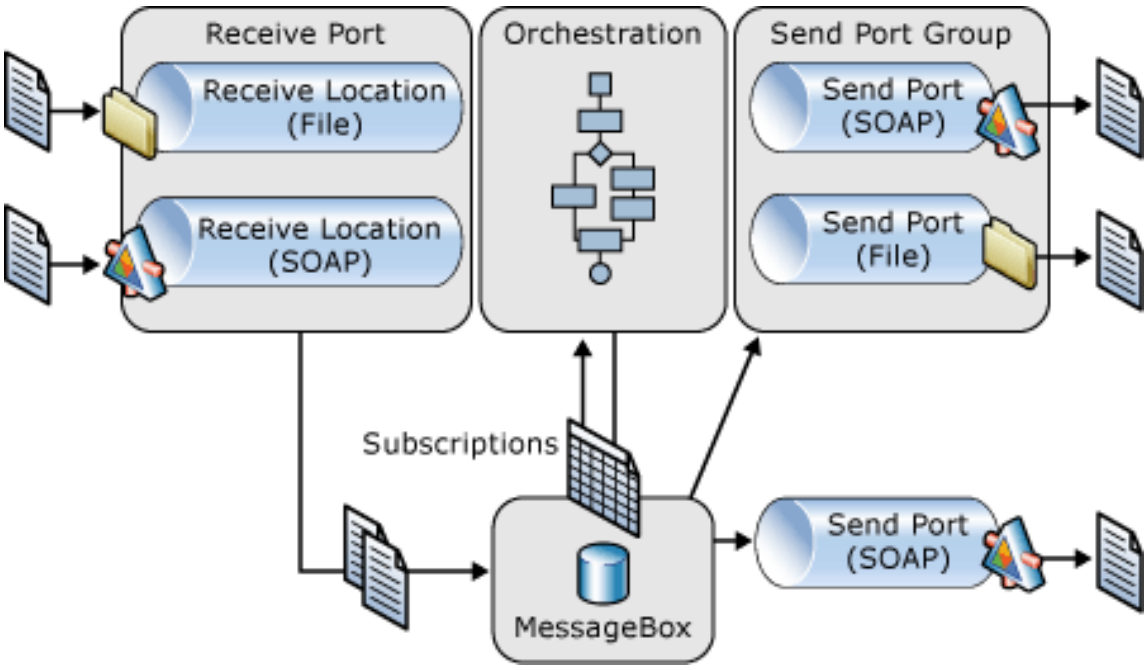
Este artigo pretende ser uma nota introdutória e destinada a quem está a dar os primeiros passos nesta tecnologia.

Podemos definir o BizTalk como um servidor de encaminhamento de mensagens, capaz de tratar, validar, transformar e controlar inúmeros processos, simplificando as necessidades de adaptação de cada sistema a interligar. Ou seja, um componente de infra-estrutura essencial nas ligações entre empresas (B2B - *Business-to-Business*) e cada vez mais, usado para ligar sistemas, também eles, cada vez mais complexos dentro das organizações (EAI - *Enterprise Application Integration*). Para além dos padrões "*Fire & Forget*", o BizTalk é também usado para cenários mais complexos onde o *workflow* depende de várias mensagens que precisam de ser correlacionadas para orquestração processos de negócio (BPM - *Business Process Management*). Neste artigo vamos focar no processo de mapeamento e transformação de mensagens apenas.

De maneira simples podemos definir que o BizTalk é um servidor de integração projectado para trabalhar com mensagens, ideal para ser usado principalmente para integração de aplicações corporativas (EAI), integração de sistemas entre parceiros de negócio (B2B) e para gestão de processos de negócio (BPM).

Arquitectura

As mensagens entram no BizTalk através de uma porta lógica (portas de recepção ou *Receive Port*) que são compostas por 1 ou varias portas físicas (locais de recepção ou *Receive Locations*). Cada local de recepção possui uma configuração específica para um adaptador, como podemos verificar no exemplo seguinte:



Um adaptador FILE poderia ser por exemplo uma pasta de rede (*fileshare.local**Encomendas*\) e um filtro (*.edifact) e, paralelamente poderíamos também estar a receber encomendas por um Web Service (SOAP/REST/XML).

Quando o servidor recebe uma mensagem num adaptador, este executa um *pipeline*. O *pipeline* é simplesmente uma composição sequencial de componentes que tem como principal objectivo:

- **Converter as mensagens** que podem estar em diferentes formatos (arquivos de texto (Flat File), arquivos compactados - ZIP), para o formato que o BizTalk usa internamente para processar as mensagens: XML (Extensible Markup Language).
- **Validar as mensagens recebidas**. No seu normal funcionamento, o BizTalk só processa mensagens reconhecidas internamente, para isso utiliza esquemas XML (XML Schema) que permitem descrever a estrutura (record, elemento, atributo, nome, tipo de dado) e define as regras de validação (se é ou não obrigatório, numero de vezes que o elemento pode aparecer, hierarquia) das mensagens XML.

De seguida as mensagens são despejadas internamente na MessageBox (base de dados) onde são os diferentes subscritores (1 ou mais interessados nessa mensagem) a vão receber. Estes subscritores podem ser outras portas de saída (*Routing*) ou entram em orquestrações lançando novos processos, ou acordando os que estavam à espera (via campos correlacionáveis).

O que são os mapas de BizTalk e onde podem ser utilizados?

Os mapas de BizTalk são representações gráficas de documentos XSLT (*Extensible Stylesheet Language Transformation*) que permitem efectuar, de forma simples e visual, transformações às mensagens XML.

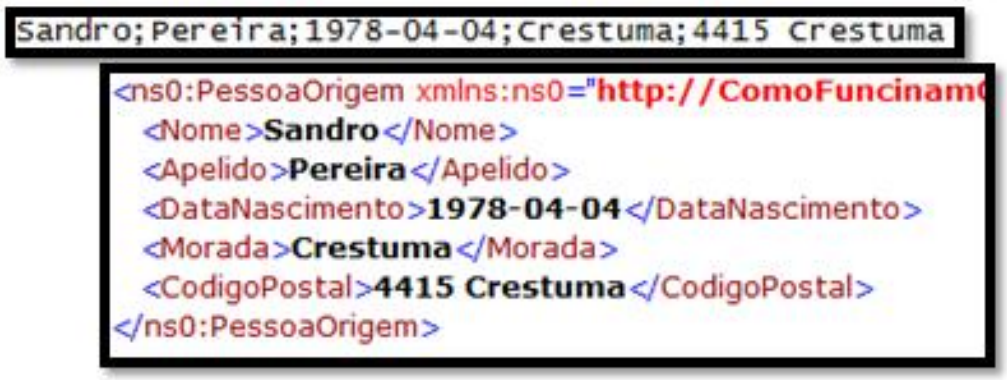
Podemos enumerar os standards usandos no BizTalk Mapper:

- **XML** (*Extensible Markup Language*) – contêm os dados das mensagens;
- **XML Schema** (*XSD - XML Schema Definition*) – define o formato das mensagens;
- E **XSLT** (*Extensible Stylesheet Language Transformation*) – define as regras de tranformação das mensagens;

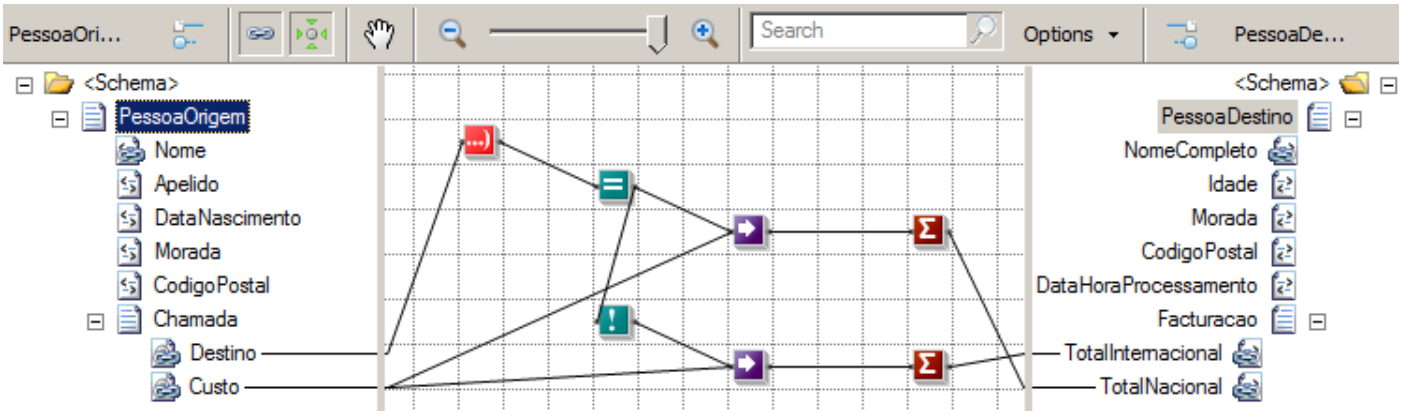
De realçar que todos eles são é uma recomendação da W3C (*Worldwide Web Consortium*) - consórcio internacional, que agrega empresas, órgãos governamentais e organizações independentes, e que visa desenvolver standards para a criação e a interpretação de conteúdos para a Web.

Podemos caracterizar dois tipos de transformações existentes:

- **Transformações de Sintaxe**: Este tipo de transformações ocorrem nas *pipelines* de recepção ou envio e têm como objectivo transformar um documento noutra representação, por exemplo de CSV para XML. Aqui o documento mantem os mesmos dados (semântica), mas muda a sintaxe com que é representado. Ou seja traduzimos o documento mas, normalmente, não o modificamos em termos de estrutura. Por norma este tipo de transformação é bidireccional, uma vez que continuamos a ter o mesmo conteúdo semântico. Podemos aplicar a mesma lógica de transformação e voltar a obter um documento no seu formato original.



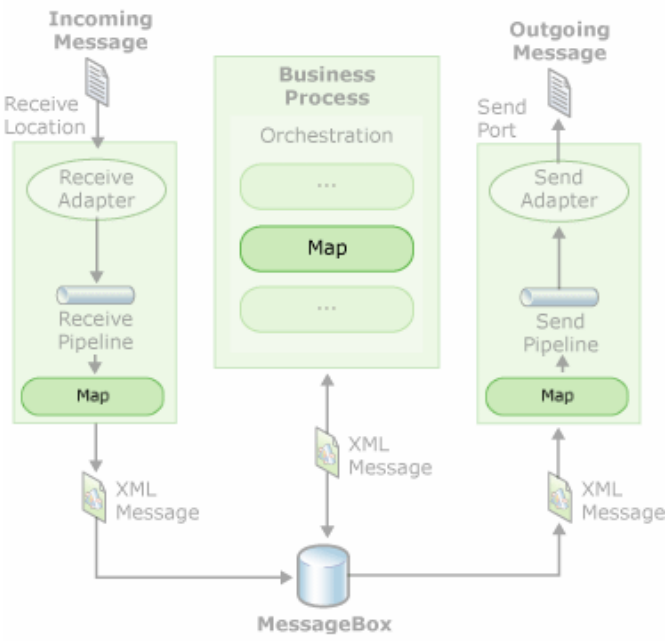
- **Transformações de Semântica**: Este tipo de transformações ocorre por norma apenas nos mapas de BizTalk. Aqui o documento mantem a mesma sintaxe com que é representado (XML), mas muda a sua semântica. Tipicamente são operações *One-way*, uma vez que quando extraímos e agregamos partes de informação de um documento e compomos um outro documento diferente, podendo perde detalhes importantes para a sua reconstrução.



Nota: Neste artigo vamos falar apenas nas transformações de semântica, ou seja, nos mapas de BizTalk.

Onde podem ser utilizados os mapas?

Conforme a imagem a baixo demonstra, os mapas de BizTalk podem ser utilizados à entrada, nas orquestrações, ou nas portas de saída.

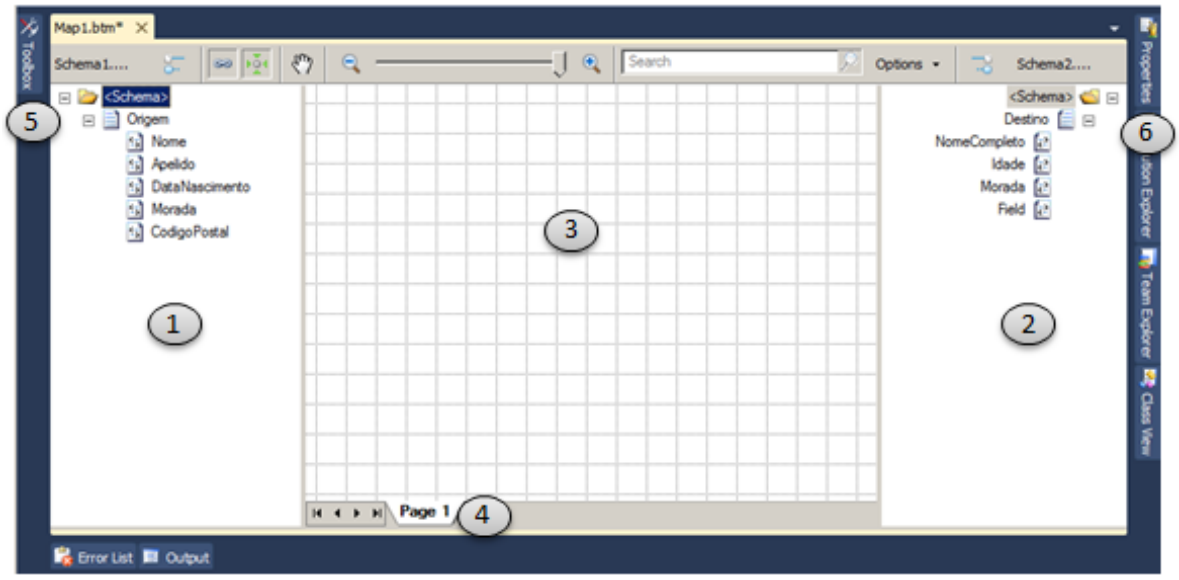


A grande diferença entre utilizar nas portas ou em orquestrações, é que a utilização na última pode ter múltiplos *inputs* de mensagens (transformações de vários documentos para um documento final – transformações N←→1) e nas portas apenas permite uma única mensagem de *input* (transformações 1←→1).

Introdução ao editor de mapas - BizTalk Mapper Designer

O editor de mapas, *BizTalk Mapper Designer*, possibilita efectuar transformações de mensagens XML complexas de forma visual e extremamente simples, expressas em associações gráficas de ligações (*links*) que definem as relações entre os vários elementos das mensagens.

Estas relações entre elementos são internamente implementadas como transformações XSLT (*Extensible Stylesheet Language Transformation*) que é o *standard* recomendado pela *Worldwide Web Consortium* (W3C) para efectuar transformações entre esquemas XML.

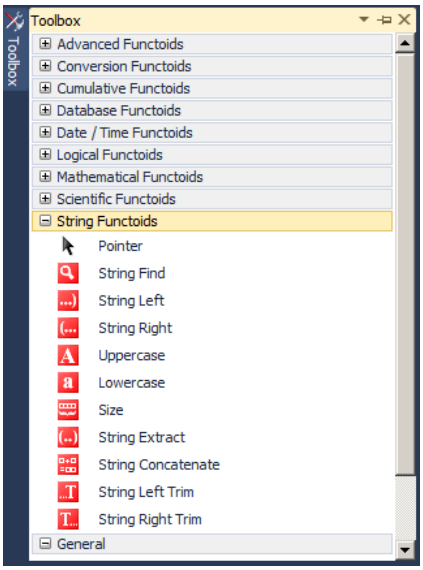


Esta ferramenta encontra-se integrada no Visual Studio e é composta essencialmente por 3 módulos:

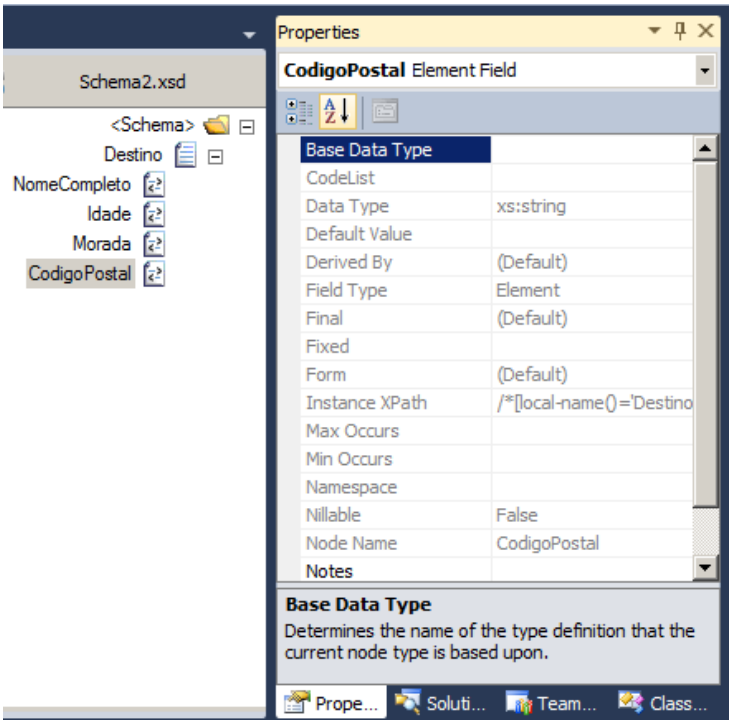
- **Esquema de Origem** (*source schema*): trata-se da estrutura de dados da mensagem de origem e encontra-se na parte esquerda da janela principal – **ponto 1**;
- **Esquema de Destino** (*destination schema*): trata-se da estrutura de dados da mensagem final após ser efectuada a transformação e encontra-se na parte direita da janela principal – **ponto 2**;
- **Grelha de mapeamento** (*mapper grid*): encontra-se no meio da janela principal, entre as duas estruturas de dados (origem e destino) - **ponto 3**;
  Esta zona desempenha um papel crítico na definição de mapas, contendo as ligações e as *functoids* que iram controlar a forma como os dados de origem da mensagem são transformados, de acordo com o esquema de destino, para a mensagem final.
  Cada mapa pode ter até 20 páginas (*mapper grids*), acedíveis através dos separadores (*tabs*) que se encontram no fundo da grelha de mapeamento – **ponto 4**.

Para além destes 3 módulos, existem 2 janelas de extrema importância para o programador:

- **Janela de Ferramentas** (toolbox window): normalmente encontra-se no lado esquerdo do esquema de origem – ponto 5;
  Providencia acesso a todas as *functoids* que podemos utilizar nos mapas.

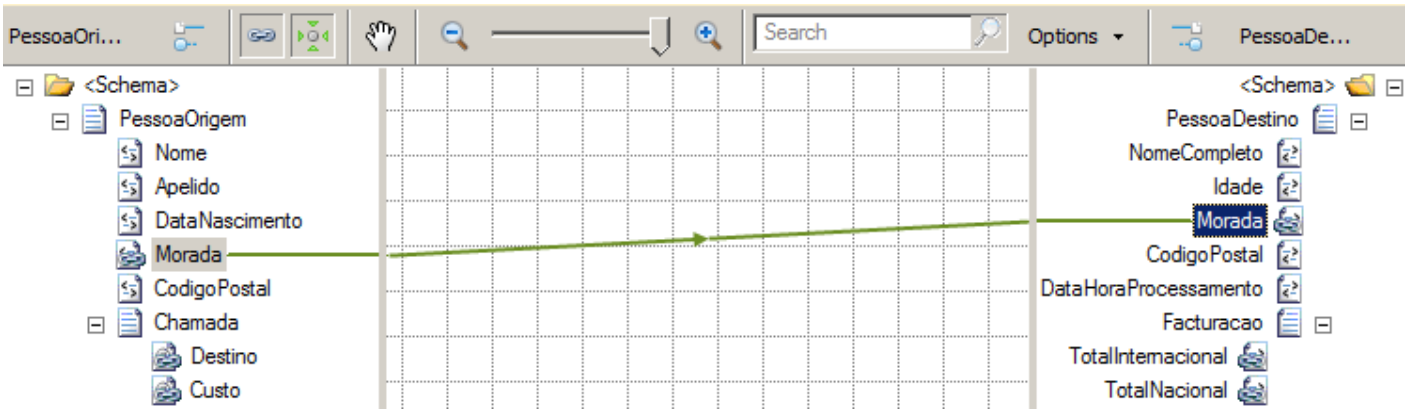


- **Janela de Propriedades** (*properties window*): nesta janela podemos ver e modificar as propriedades de um objecto seleccionado na grelha ou nos esquemas, normalmente encontra-se disponível à direita do esquema de destino – ponto 6.



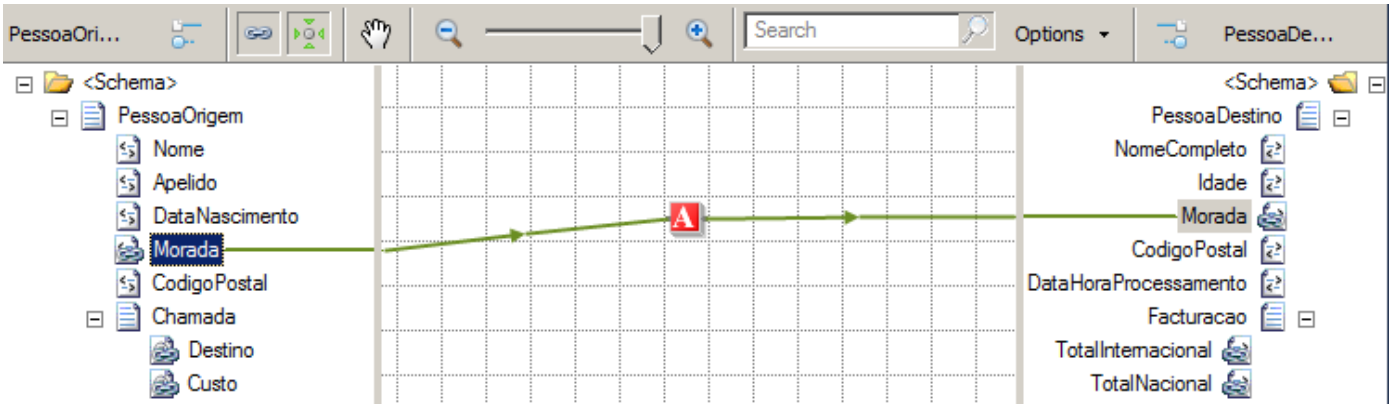
Ligações e Functoids

As transformações num mapa podem ser definidas na forma de relações simples, como copiar um nome ou um endereço de um documento para outro. Podemos expressar uma cópia directa dos dados usando uma ligação, que é representada no *BizTalk Mapper Designer* como uma linha que liga os elementos da origem para os elementos de destino.



O utilizador também pode especificar transformações mais complexas usando *functoids*. Podemos considerar uma *functoid* como funções pré-definidas que podemos utilizar para efectuar mapeamentos ou transformações complexas.

Tipicamente num mapa, os dados são copiados da origem para o destino, arrastando ligações entre os elementos dos dois esquemas. As *functoids* ficam no meio destas operações e aplicam uma operação sobre os dados de entrada, de modo a transformá-los às exigências do destino. *BizTalk Mapper Designer* representa uma *functoid* como uma caixa no meio da ligação ou ligações entre os elementos de transformação.



BizTalk fornece um extenso conjunto de *functoids* que podem ser usadas nos mapas para executar uma variedade de operações nos dados que estão a ser transformados a partir de uma mensagem de origem para uma mensagem de destino.

Por defeito, as *functoids* estão organizadas em 9 categorias com base nas suas funções:

- ***Advanced Functoids***: Usadas para criar vários tipos de manipulação de dados, como a implementação de script personalizado (C#, Visual Basic .NET, XSLT), mapear valores ou gerir e extrair dados de elementos recursivos.
- ***Conversion Functoids***: Funções típicas de conversão de valores como: conversão de caracteres para ASCII ou de números de uma base para outra (Hexadecimal, decimal).
- ***Cumulative Functoids***: Usadas para realizar vários tipos de operações de acumulação de valores que ocorrem várias vezes numa mensagem.
- ***Database Functoids***: Utilizadas principalmente para pesquisar dados existentes em base de dados.
- ***Date and Time Functoids***: Trata-se de um conjunto de operações sobre datas como: adicionar data, hora, data e hora, ou adicionar dias a uma data específica.
- ***Logical Functoids***: Estas funções permitem controlar de forma condicional o mapeamento dos valores de origem ou o comportamento de outras *functoids*, determinando assim se os dados de saída são criados ou não.
- ***Mathematical Functoids***: Utilize as *functoids* matemáticas para executar cálculos numéricos específicos, como adição, multiplicação ou divisão.
- ***Scientific Functoids***: Usadas para realizar cálculos científicos específicos, como funções logarítmicas, exponenciais ou trigonométricas.
- ***String Functoids***: Usadas para manipular dados alfanuméricos (texto) através de funções bem conhecidas tais como: calcular comprimento, concatenação de elementos, extrair bloco de texto, converter para maiúsculas ou minúsculas.

No entanto, a plataforma permite que sejam criadas novas *functoids* pelos programadores assim como organizar e criar novas categorias.

Projecto de referência para criação e instalação de novas *functoids*: "BizTalk Mapper Extensions UtilityPack".

Grelha de Mapeamento

A grelha de mapeamento desempenha um papel crítico na definição de mapas, ela irá conter as ligações e *functoids* que irão controlar a forma como os dados de origem de uma mensagem são transformados numa mensagem de destino de acordo com o seu esquema.

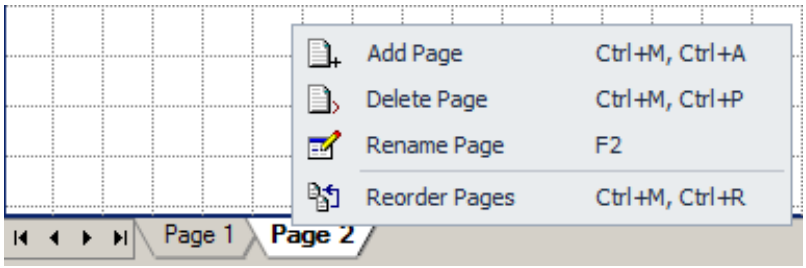
A grelha poderá ter múltiplas camadas, num máximo de 20, chamadas de páginas. Particionar os mapas em diferentes páginas, além de ser uma boa prática, pode-se tornar extremamente útil por forma a organizar-nos e desta forma torná-los mais legíveis. Apesar de em pequenos mapas, uma página ser suficiente, quando lidamos com esquemas complexos como EDI, "infestar" uma página com inúmeras ligações e *functoids* torna-os ilegíveis, chegando ao ponto de não conseguirmos distinguir um elemento do outro.

Podemos definir as páginas como contentores de ligações e *functoids*, que servem apenas para organizar os mapas, isto porque ao nível do compilador não têm qualquer impacto uma vez que são invisíveis.

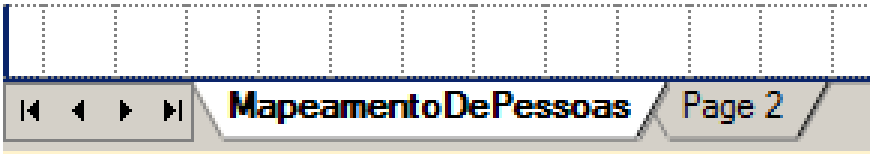
Operações possíveis nas páginas

Por defeito os mapas são criados com apenas uma página, com o nome "*Page 1*", apesar das operação mais frequentes serem a de criação e renomeação, são 4 operações as operações que podemos efectuar sobre as páginas:

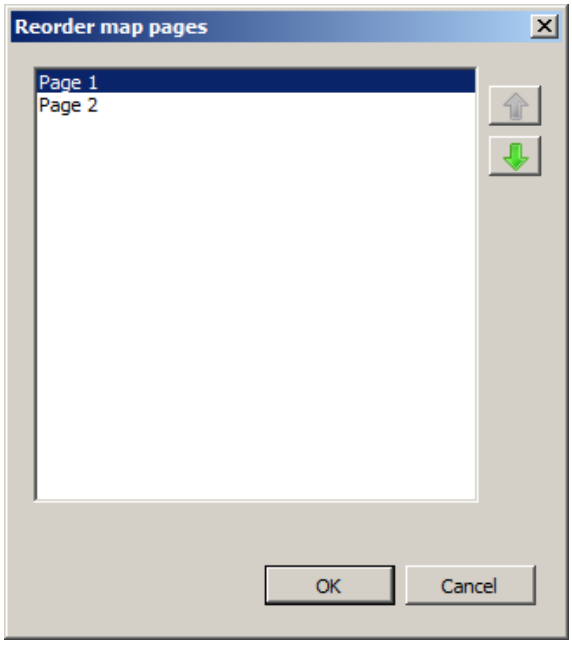
- **Adicionar nova página**: esta é a funcionalidade mais comum, adicionar novas páginas permite-nos organizar diferentes zonas do mapa em contentores.
    - Pressionar botão direito sofre o separador da página, e seleccionar a opção "*Add Page*"



- **Renomear uma página existente**: muita das vezes esquecida, esta opção permite-nos renomear os separadores de forma a tornar as diferentes páginas mais legíveis visualmente.
    - Pressionar botão direito sofre o separador da página, e seleccionar a opção "*Rename Page*"



- **Eliminar uma página existente**: eliminação de páginas desnecessárias ou obsoletas.
    - Pressionar botão direito sofre o separador da página, e seleccionar a opção "*Delete Page*"
- **Reorganizar as páginas existentes**: muita das vezes temos a necessidade de organizar a disposição das páginas numa sequência diferente, para isso basta:
    - Pressionar botão direito sofre o separador da página, e seleccionar a opção "*Reorder Pages*"



Transformações - Funcionalidades básicas dos mapas

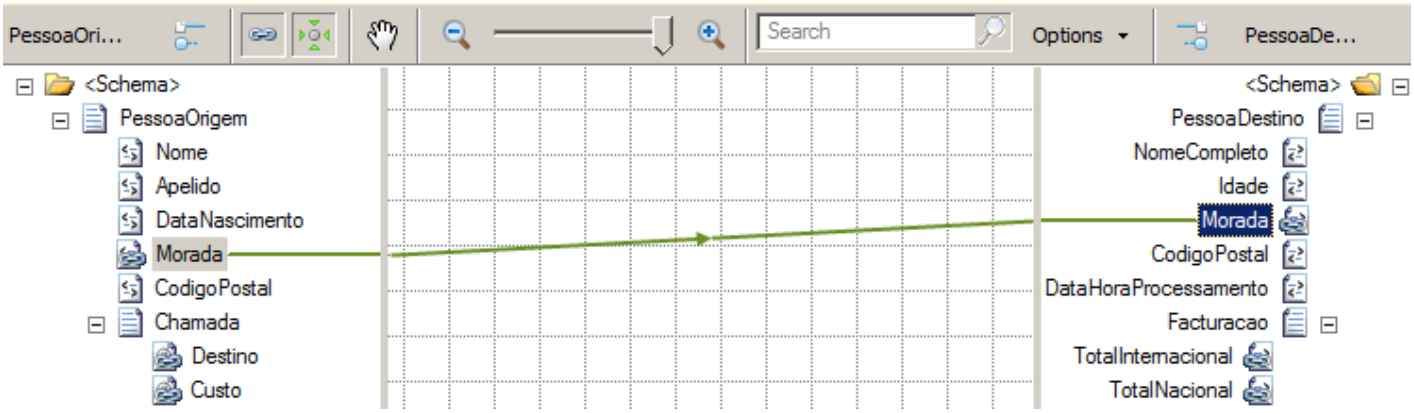
Quando estamos a efectuar uma transformação de mensagens são 5 as funcionalidades básicas que normalmente nos surgem:

- Mapeamento simples de um determinado valor (cópia directa)
- Concatenação de valores
- Selecções condicionadas
- Scripts customizados
- Adicionar novos dados

Mapeamento simples de um determinado valor (cópia directa)

Esta é a operação mais básica de todas as operações, onde pretendemos mover um valor da origem para um determinado destino, sem efectuarmos qualquer tipo de operação sobre os valores (cópia directa).

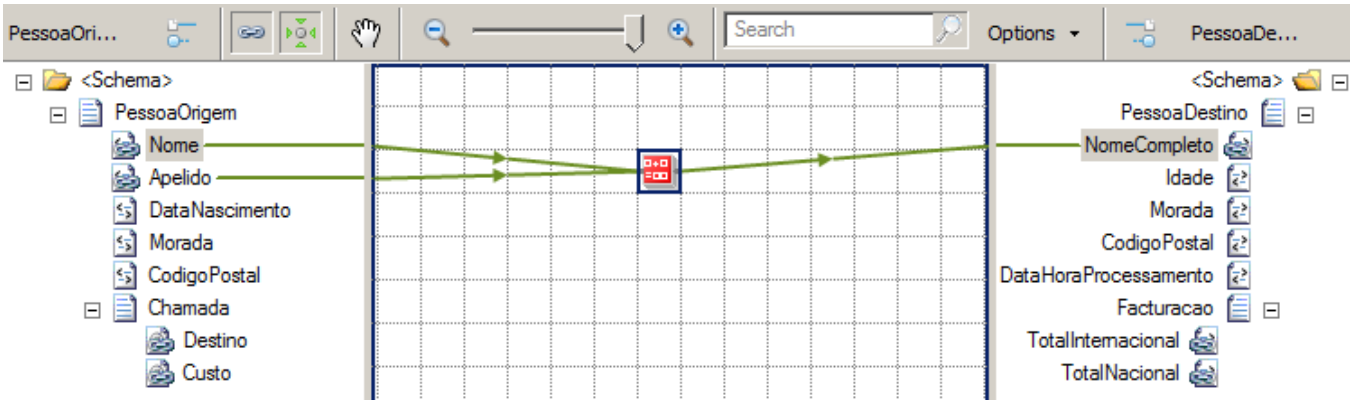
Para isso apenas de necessitamos de efectuar drag-and-drop do elemento de origem para o esquema destino. Na imagem seguinte está exemplificado o elemento Morada.



Concatenação de valores

Concatenar diferentes valores da origem para um determinado elemento no esquema de destino é outra das operações quotidianas em transformações, para isso apenas necessitamos de:

- Abrir a janela de ferramentas e arrastar para a grelha a *functoid* "*String Concatenate*";
- Conforme foi explicado no ponto anterior, arrastar uma ligação entre os elementos pretendidos e a *functoid* "*String Concatenate*", por exemplo os elementos "Nome" e "Apelido";
- Arrastar a ligação da *functoid* "*String Concatenate*" para o elemento no esquema de destino, neste caso o "NomeCompleto";



Nota: a ordem de entrada dos elementos na *functoid* é importante, uma vez que a concatenação é efectuada pela ordem de entrada dos elementos (iremos aprofundar este tema mais à frente).

Selecções condicionadas

Muitas das vezes não queremos simplesmente mover valores da origem para o destino, às vezes necessitamos de gerar a saída dos dados de acordo com certas condições. Podemos efectuar estas condições de muitas formas distintas através de *functoids* ou scripts customizados, aqui fica um exemplo: Testar se o valor na origem é uma *string* válida, se sim mapeá-la para o esquema de origem, para isso necessitamos:

- Arrastar a *functoid* "*Logical String*" para a grelha, esta *functoid* permite validar se um determinado valor é uma *string* válida, semelhante a função C# *String.IsNullOrEmpty*
  - Retorna "*False*" se a *string* for vazia ou nula;
  - Retorna "*True*" caso seja uma *string* válida;
- Arrastar uma ligação do elemento pretendido da origem para a *functoid* "*Logical String*", neste caso o elemento "CodigoPostal";
- Arrastar a *functoid* "*Value Mapping*" para a grelha, esta *functoid* providencia, através de um valor booleano, controlar a saída dos valores da origem para o destino, ou seja, recebe dois valores:
  - O primeiro terá de ser um booleano (True/False);
  - O segundo será o valor a ser mapeado;
  - Caso o primeiro seja verdadeiro, o valor é mapeado para o esquema de destino, caso seja falso, o mesmo não é mapeado.
- Arrastar uma ligação da *functoid* "*Logical String*" para a *functoid* "*Value Mapping*";
- Arrastar uma ligação do elemento "CodigoPostal" do esquema de origem para a *functoid* "*Value Mapping*";
- Arrastar uma ligação da *functoid* "*Value Mapping*" para o elemento "CodigoPostal" do esquema de destino;

Scripts customizados

Scripts customizados são utilizados normalmente em transformações mais complexas ou por forma a facilitar algumas condições. Basicamente existem dois cenários para utilizarmos esta *functoid*:

- Quando nenhuma das *functoids* existentes permite efectuar o que pretendemos, o exemplo que vamos ver é converter uma data de nascimento em idade.
- Ou quando a utilização das *functoids* existentes tornam-se extremamente complexas para resolver um problema do mapeamento.

Existe uma "regra" que normalmente usa-mos para definir se devemos utilizar *functoids* ou utilizar scripts customizados que nos diz: se para um determinado mapeamento utilizarmos mais de 6 *functoids*, então deveremos utilizar scripts customizados.

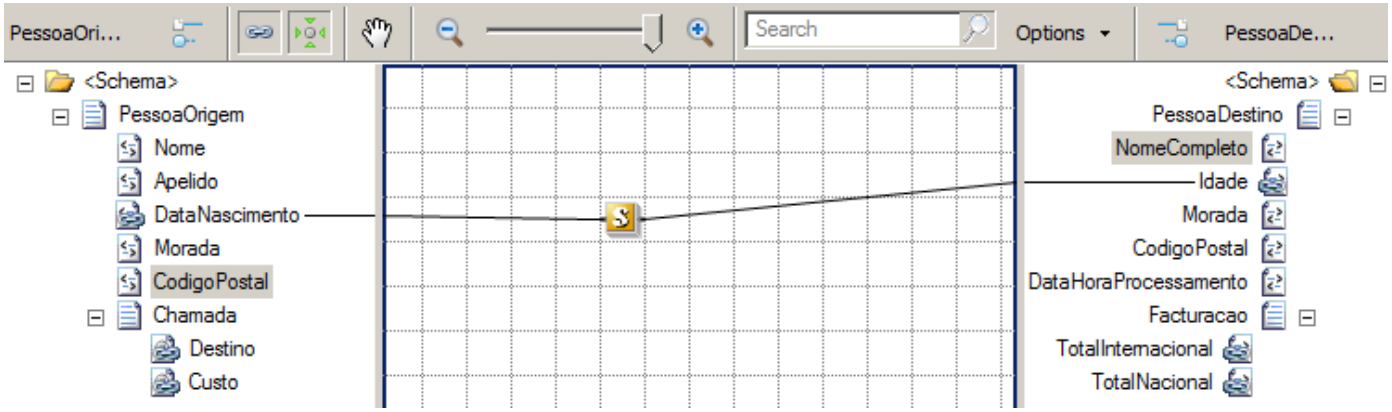
Eu gosto de utilizar esta regra de forma ponderada e não à letra, ou seja, se as *functoids* existentes me ajudarem de forma fácil a resolver o problema, eu utilizo as *functoids* existentes, se a utilização das mesmas, se torna extremamente complexa, então opto por utilizar scripts customizados.

Estão ao dispor do programador 6 tipos de scripts:

- ***External Assembly***: Permite-nos associar e invocar esta *functoid* com uma função existente numa *assembly* existente na *Global Assembly Cache* (GAC).
- ***Inline C#***: Esta opção permite-nos associar e invocar código C# directamente na *functoid* (*Inline Script Buffer property*).
- ***Inline JScript .NET***: Igual à anterior, mas com uso de código JScript .NET
- ***Inline Visual Basic .NET***: Igual às anteriores, mas com uso de código Visual Basic .NET
- ***Inline XSLT***: Esta opção permite-nos associar e invocar scripts XSLT directamente na *functoid* (*Inline Script Buffer property*).
- ***Inline XSLT Call Template***: idêntico à anterior, no entanto esta permite-nos invocar templates XSLT directamente da *functoid*.

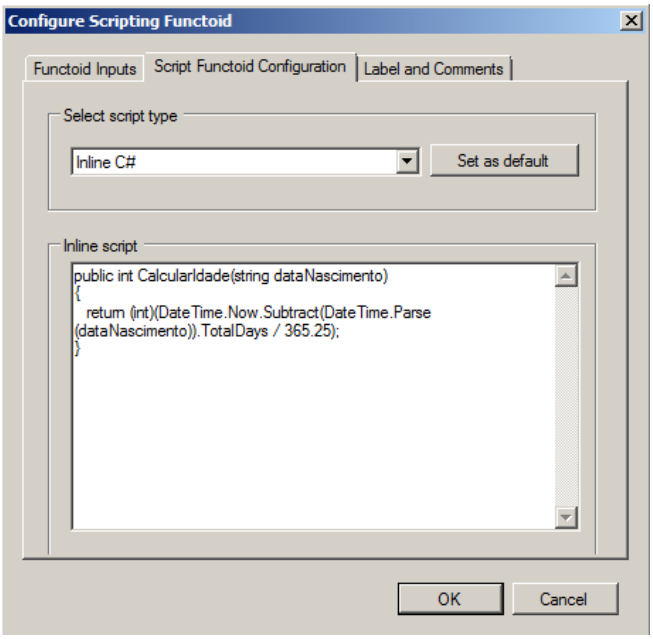
Neste exemplo, queremos transformar a data de nascimento na idade da pessoa, neste cenário devemos:

- Arrastar a *functoid* "*Scripting*" para a grelha;
- Arrastar uma ligação do elemento "DataNascimento" do esquema de origem para a *functoid* "*Scripting*";
- Arrastar uma ligação da *functoid* "*Scripting*" para o elemento "Idade" do esquema de destino;



O mapeamento encontra-se efectuado, faltando-nos apenas configurar a script customizado. Para este cenário vamos utilizar um script do tipo "*Inline C#*", e para isso devemos:

- Efectuar duplo click na *functoid* "*Scripting*" e seleccionar o separador "*Script Functoid Configuration*";
- Na opção "*Select script type*", seleccionar da lista a opção "*Inline C#*";
- Na opção "*Inline script*" colocar o seguinte script:

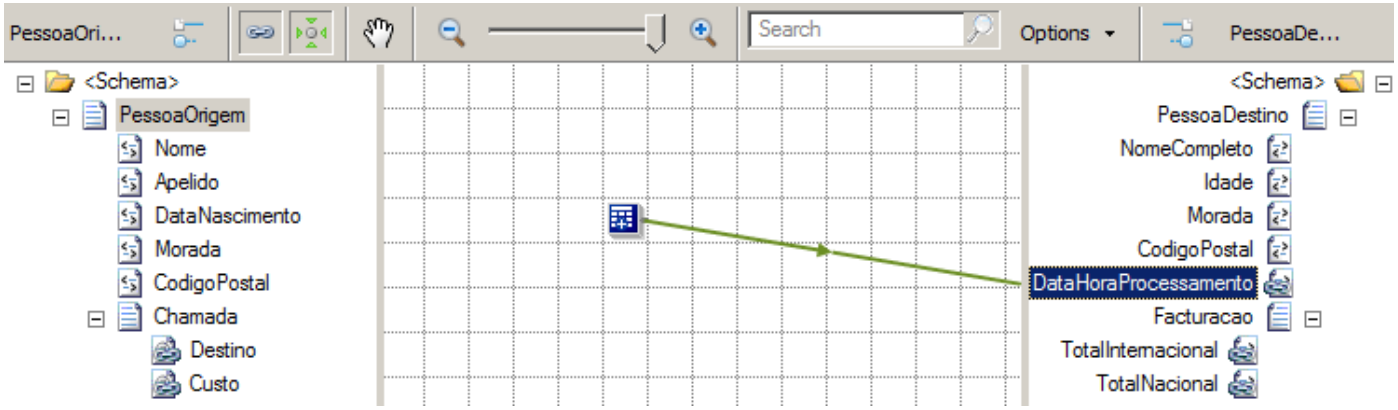


Adicionar novos dados

Em muitos cenários temos a necessidade de adicionar novos dados à mensagem final que não existem na mensagem de origem. É normal encontrarem situações em que, com base em alguns dados existentes na mensagem de origem, necessitaremos de consultar um sistema externo, por exemplo base de dados, por forma a adquirimos mais informações para preencher os dados necessários na mensagem final.

Um exemplo mais básico e simples é adicionar à mensagem final um carimbo de data/hora actual em que a mensagem é processada, para isso devemos:

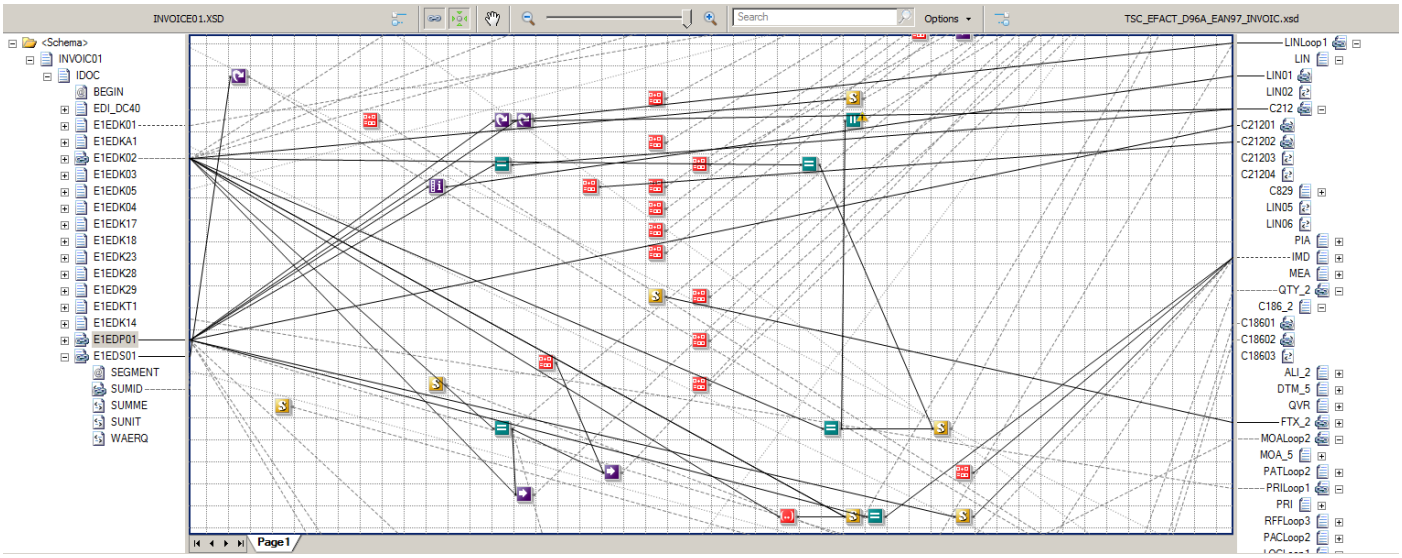
- Arrastar a *functoid* "*Date and Time*" para a grelha;
- Arrastar uma ligação da *functoid* "*Date and Time*" para o elemento no esquema de destino "DataHoraProcessamento";



Nota: Como podem verificar, esta *functoid* não requer nenhum dado de entrada, retornando a data e hora actual do sistema.

Como organizar os mapas BizTalk

Os mapas podem tornar-se extremamente complexos, dificultando assim a sua leitura e manutenção.

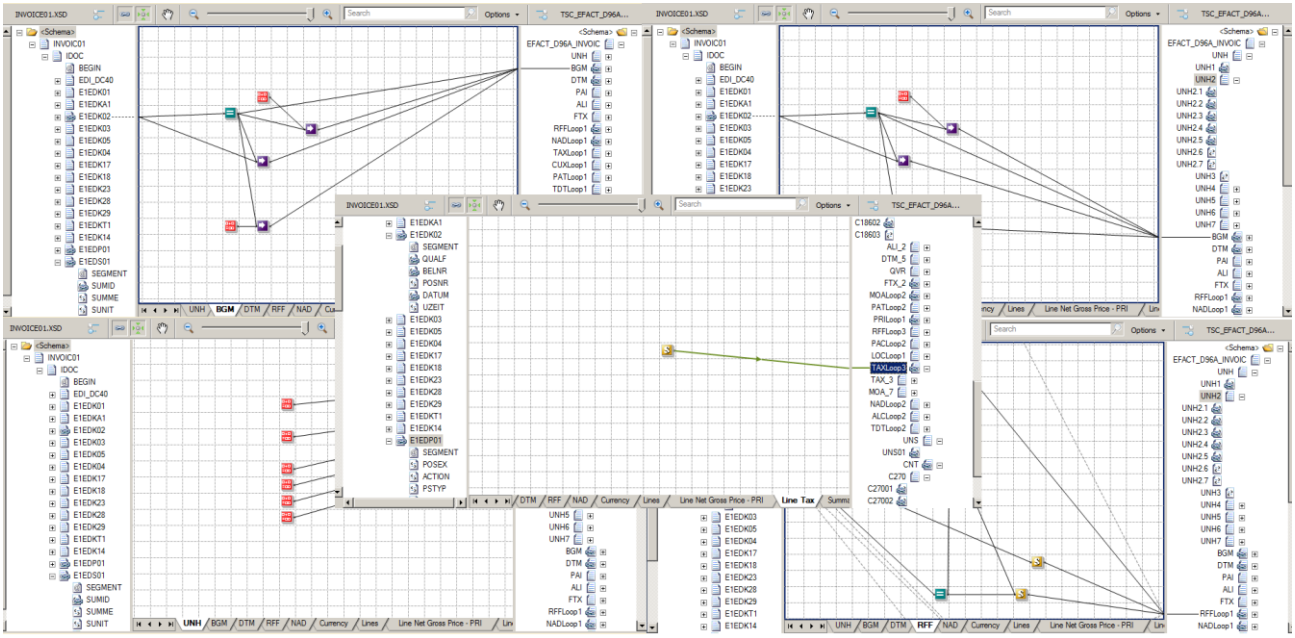


O editor de mapas do BizTalk providencia duas importantes funcionalidades que nos permitem minimizar este problema:

- Páginas
- Etiquetas para as ligações

Páginas

As páginas permitindo organizar os mapas, especialmente os mais complexos, em subdivisões lógicas de mapeamentos. Esta funcionalidade já foi descrita anteriormente.



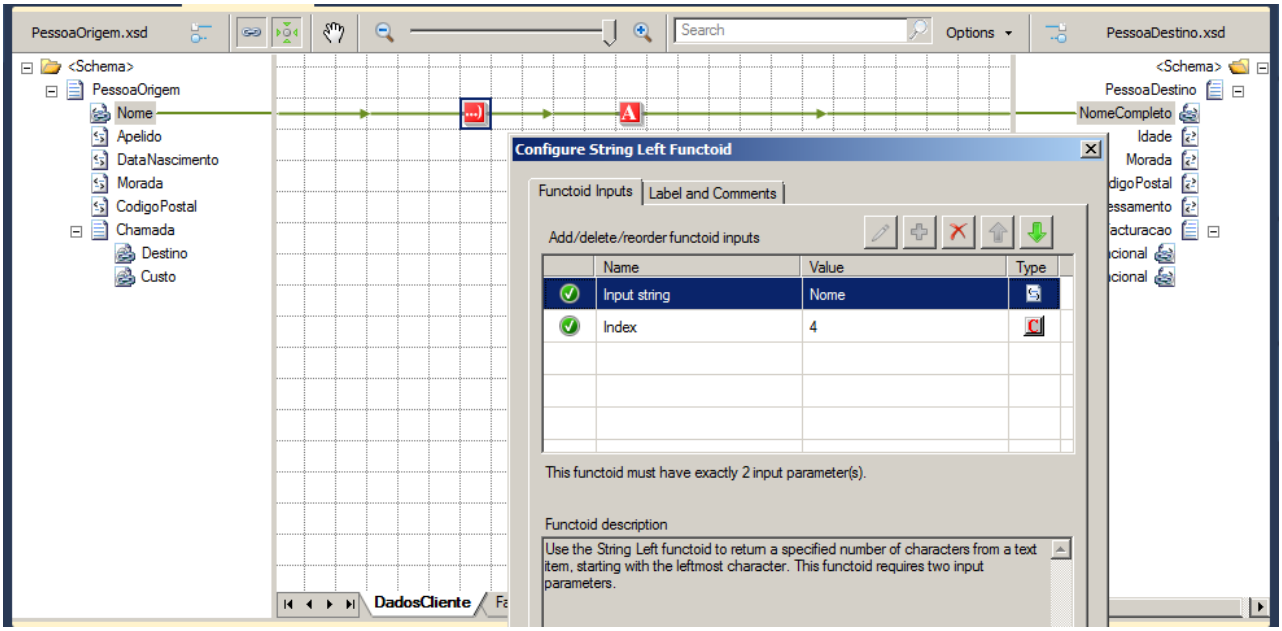
Etiquetas para as ligações

Nas versões anteriores do produto, por defeito, era apresentado a *query* XPATH se uma ligação fosse estabelecida a uma *functoid*:

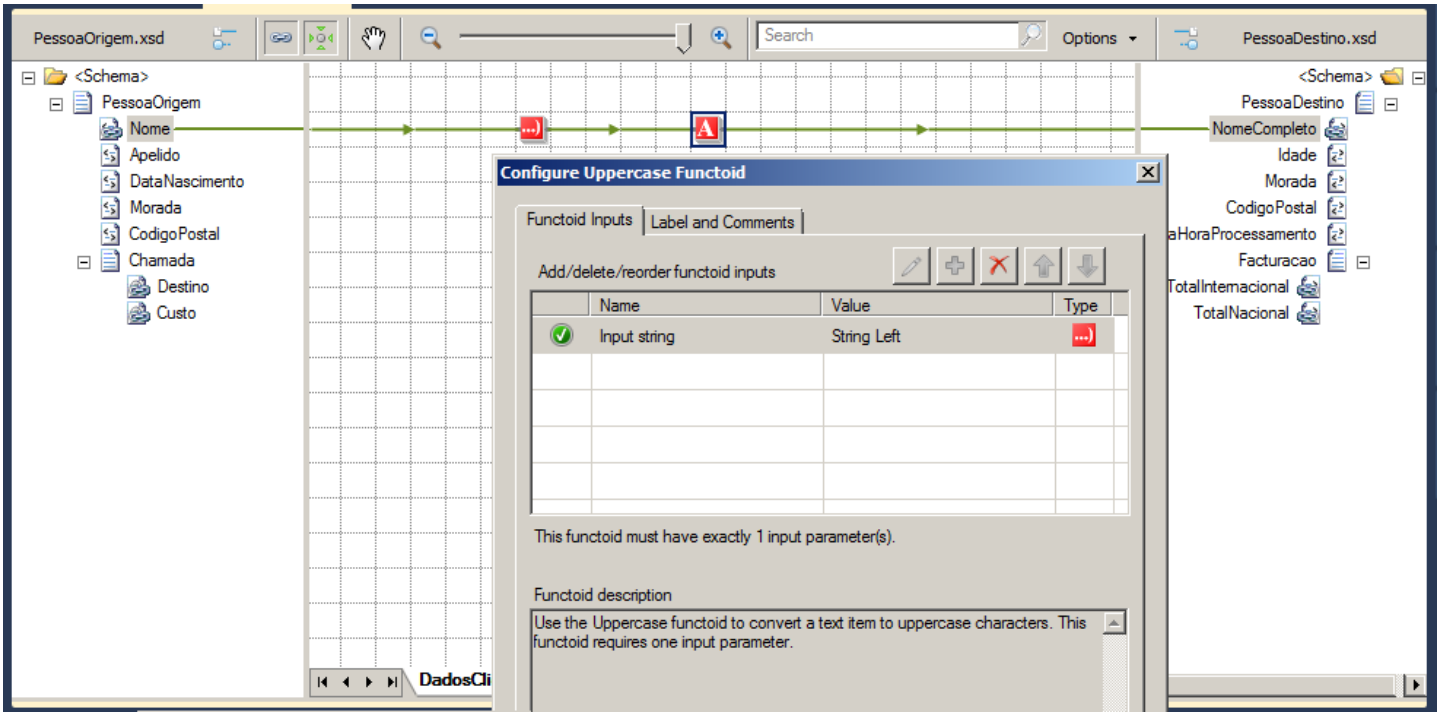
- /*[LOCAL-NAME()='PESSOAORIGEM' AND NAMESPACE-URI()='HTTP://COMOFUNCINAMOSMAPAS.PESSOAORIGEM']/*[LOCAL-NAME()='NOME' AND NAMESPACE-URI()='']

Ou o nome da *functoid* caso a ligação *proviesse* de outra *functoid*, o que poderá dificultar a leitura dos mapas.

Nesta versão do produto este comportamento foi ligeiramente melhorado, sendo que actualmente o valor defeito é o **nome do elemento do esquema de origem** de onde provem a ligação:

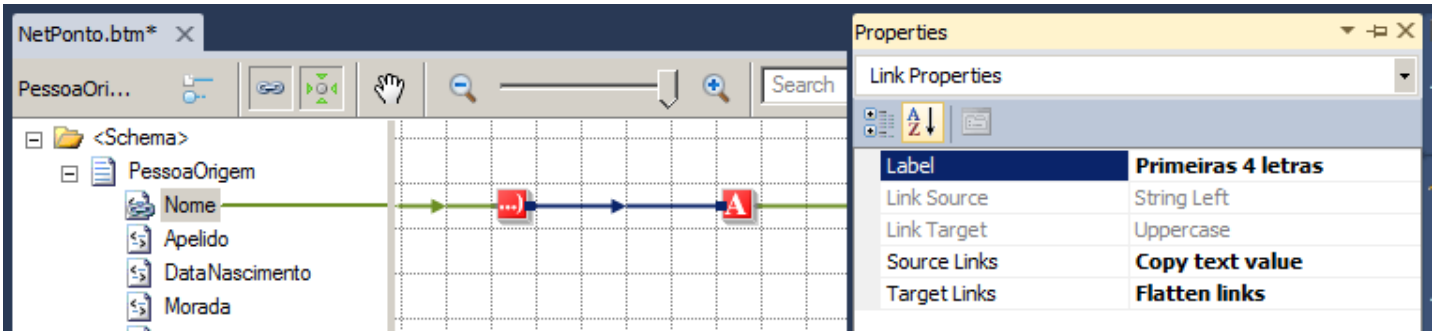


Ou o **nome da *functoid*** anterior:



No entanto, o editor de mapas permite-nos colocar etiquetas nas ligações, originando a que a mesma substitua a *query* XPATH (nas versões antigas) ou o nome do elemento do esquema de origem, para isso devemos:

- Seleccionar a ligação e com o botão direito do rato seleccionar a opção "*Properties*";
- Preencher a propriedade "*Label*" com a etiqueta pretendida.



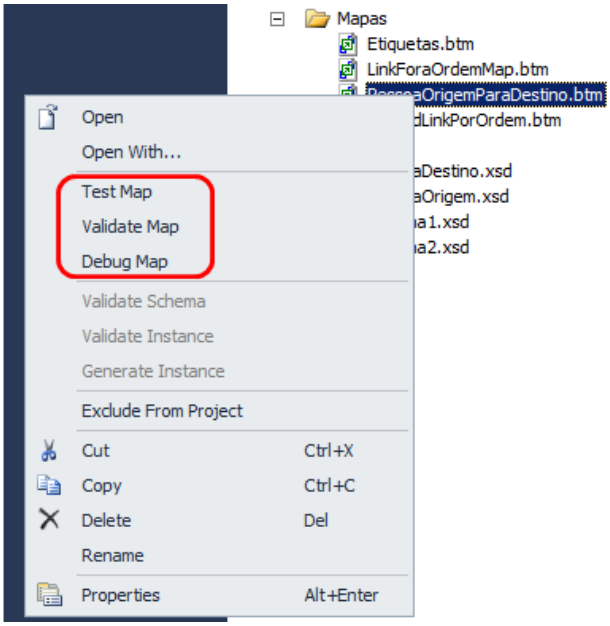
Normalmente esta funcionalidade é esquecida pelos programadores.

Apesar de parecer uma funcionalidade banal e sem impacto significativo, a meu ver, trata-se de uma funcionalidade importante a longo prazo. Enquanto as ideias estão frescas sabemos o que estamos a fazer, mas se for necessário intervir passado algum tempo e rever os mapeamentos, esta funcionalidade irá tornar a nossa tarefa mais facilitada.

Operações possíveis de efectuar sobre os mapas

Na fase de desenvolvimento, temos ao dispor 3 funcionalidades, incluídas no Visual Studio, que nos permite testar e validar os mapas:

- Testar mapas
- Validar mapas
- Depurar mapas



Estas funcionalidades estão ao dispor dos programadores de uma forma fácil e directamente da ferramenta de desenvolvimento, Visual Studio, não necessitando assim de executar (*build*) e publicar (*deploy*) os mapas ou até mesmo criar e configurar portas.

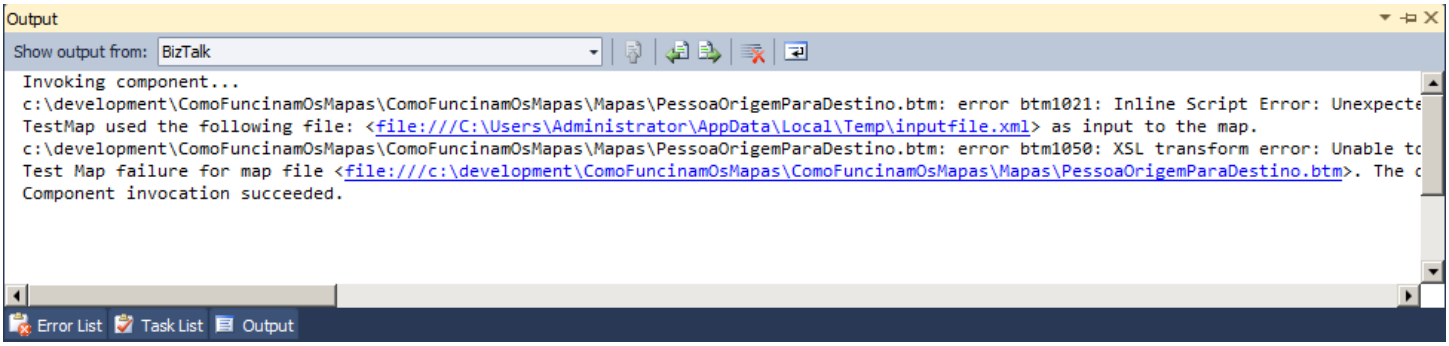
Testar mapas (Test Map)

Deveremos testar o nosso mapa de uma forma continua, não apenas no final do desenvolvimento, mas sempre que necessário ou quando um bloco de transformação importante seja concluído.

Para isso apenas necessitamos de:

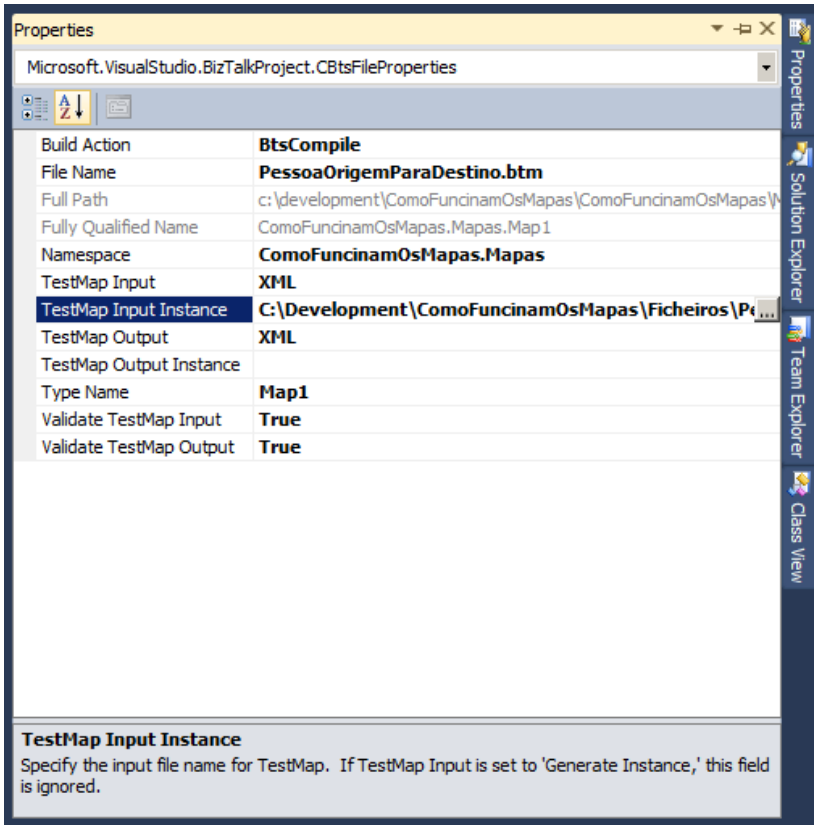
- Abrir a janela do "*Solution Explorer*"
- Seleccionar o mapa em causa
- Premir o botão direito do rato sobre o mapa e seleccionar a opção "*Test Map*"

Por defeito uma instância do esquema de entrada é gerada automaticamente com valores, também eles por omissão, de acordo com o seu tipo e testada no mapa seleccionado. No final é apresentado o resultado gerado ou os erros ocorridos na execução do mesmo na janela de saída (*Output*).



No entanto, muita das vezes, este cenário não é o ideal, e o que pretendemos é testar um documento existente com um mapa. Para isso apenas necessitamos de configurar as propriedades do mapa antes de corremos o teste:

- Efectuar o procedimento anterior, mas em vez de seleccionar a opção "*Test Map*", iremos seleccionar a opção "*Properties*";
- Na janela de propriedades configurar a propriedade "*TestMap Input Instance*" com o caminho para o ficheiro de entrada.



Nesta janela também podemos configurar diversas propriedades como: o formato de entrada e saída dos documentos, assim como o caminho final do documento de saída, mas mais importante podemos configurar se pretendemos validar os formatos de entrada e saída com os respectivos esquemas.

Esta segunda opção: **validar formato do ficheiro de saída** (*Validate Test Map Output*), é extremamente importante para os testes intermédios. Ao configurar esta opção como "*False*" permite-nos testar um mapa incompleto sem ser apresentados erros de testes ou falta de dados obrigatórios, muito deles associados a zonas por mapear, permitindo-nos assim apenas validar o trabalho efectuado até à data.

Nota: Esta opção deverá ser configurada como "*True*" para os testes finais.

Validar mapas (Validate Map)

Esta opção permite-nos validar a estrutura do mapa. Podemos desta forma inspeccionar e analisar o código XSLT gerado pelo compilador, fornecendo-nos informações adicionais de como os mapas funcionam assim como mais uma outra opção de depuração (*debug*).

Para executar esta opção basta:

- Aceder ao "*Solution Explorer*";
- Botão direito do rato sobre o mapa (arquivo *. btm)
- E seleccione a opção "*Validar Map*".

Uma ligação para o XSLT gerado será apresentada na janela de saída (*Output*).

```
C:\Users\Administrat...rigemParaDestino.xsl
URL: C:\Users\Administrator\AppData\Local\Temp\_MapData\ComoFuncinamOsMapas\PessoaOrigemParaDestino.xsl
xmlns:userCSharp="http://schemas.microsoft.com/BizTalk/2003/userCSharp">
<xsl:output omit-xml-declaration="yes" method="xml" version="1.0" />
- <xsl:template match="/">
    <xsl:apply-templates select="/s0:PessoaOrigem" />
  </xsl:template>
- <xsl:template match="/s0:PessoaOrigem">
    <xsl:variable name="var:v1" select="userCSharp:StringConcat(string(Nome/text()) , "" "" , string(Apelido/text()))" />
    <xsl:variable name="var:v3" select="userCSharp:LogicalIsString(string(CodigoPostal/text()))" />
  - <ns0:PessoaDestino>
    - <NomeCompleto>
        <xsl:value-of select="$var:v1" />
      </NomeCompleto>
      <xsl:variable name="var:v2" select="userCSharp:CalcularIdade(string(DataNascimento/text()))" />
    - <Idade>
        <xsl:value-of select="$var:v2" />
      </Idade>
    - <Morada>
        <xsl:value-of select="Morada/text()" />
      </Morada>
    - <xsl:if test="string($var:v3)='true'">
        <xsl:variable name="var:v4" select="CodigoPostal/text()" />
      - <CodigoPostal>
          <xsl:value-of select="$var:v4" />
        </CodigoPostal>
      </xsl:if>
    </ns0:PessoaDestino>
  </xsl:template>
- <msxsl:script language="C#" implements-prefix="userCSharp">
  - <![CDATA[
      public int CalcularIdade(string dataNascimento)
```

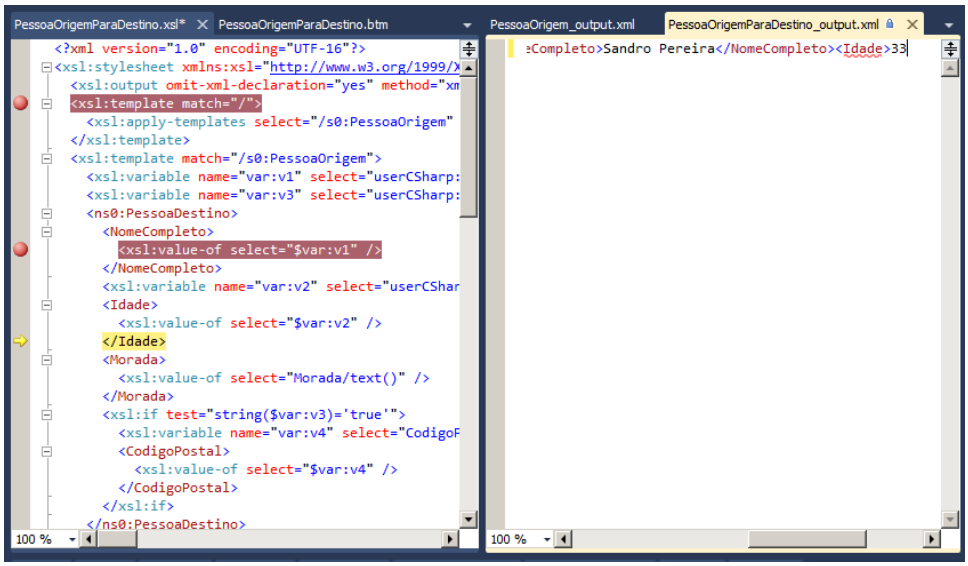
Depurar mapas (Debug Map)

Esta opção permite-nos efectuar a depuração (*debug*) de mapas e assim identificar e corrigir problemas complexos de mapeamento na fase de desenvolvimento.

Esta funcionalidade faz uso de algumas propriedades dos mapas, tais como "*TestMap Input Instance*" e "*TestMap Output Instance*". Portanto, antes de depurar o mapa, será necessário configurar as suas propriedades dos mapas, com especial foco nas duas anteriores.

Para executar esta opção, teremos de:

- Aceder ao "*Solution Explorer*"
- Botão direito do rato sobre o mapa (arquivo *. btm)
- E seleccione a opção "*Debug Map*".
- Pressione a tecla F10 ou F11 para depurar o código XSL.
    - Quando você pressiona F11 numa chamada a uma functoid, o Visual Studio irá entrar para o código C# associado à functoid.



Conclusão

Devido ao inúmero número de sistemas e aplicação distintas existentes nas organizações, é imperativo usar boas ferramentas e técnicas para produzir soluções que funcionem durante muitos anos de uma forma controlada e fácil de manter. Ao mesmo tempo novos processos são adicionados e os existentes vão sofrendo pequenas melhorias, tudo sem perder o rasto ao que está a acontecer em produtivo.

O BizTalk ajuda-nos a resolver muitos destes problemas, dispondo de inúmeras funcionalidades "*out of the box*" com o produto. Neste artigo acho que consegui explicar de uma forma intuitiva os principais conceitos básicos dos mapas de BizTalk, à medida que exploramos o seu editor.

Autor

Escrito por Sandro Pereira [MVP & MCTS BizTalk Server 2010]

Actualmente *Senior Software Developer* na empresa DevScope (www.devscope.net). É *Microsoft Most Valuable Professional* (MVP) em Microsoft BizTalk desde 2011 (https://mvp.support.microsoft.com/profile/Sandro.Pereira). O seu principal foco de interesse são as tecnologias e plataformas de Integração (EAI): BizTalk e SOAP / XML / XSLT e Net, que utiliza desde 2002.

É um participante bastante activo nos fóruns da Microsoft (*MSDN BizTalk Server Forums*), contribuidor no *MSDN Code Gallery* e na *Microsoft TechNet Wiki*, autor do **Blog**: http://sandroaspbiztalkblog.wordpress.com/, membro da comunidade BizTalk Brasil: http://www.biztalkbrasil.com.br/ e da comunidade BizTalk Administrators: http://www.biztalkadminsblogging.com.

Podes contacta-lo em: sandro-pereira@live.com.pt (Twitter: @sandro_asp).

Sandro Pereira | DevScope | MVP & MCTS BizTalk Server 2010