RICARDO TERRA NUNES BUENO VILLELA

XSLT:

Manipulação de documentos XML

BELO HORIZONTE

2006

RICARDO TERRA NUNES BUENO VILLELA

XSLT:

Manipulação de documentos XML

Trabalho apresentado à disciplina Trabalho de Conclusão de Curso, do Curso de Ciência da Computação, da Universidade FACE - FUMEC, sob a orientação do professor Roberlei Panasiewicz e do professor Luiz Eduardo de Mello Gomes.

BELO HORIZONTE

2006

RICARDO TERRA NUNES BUENO VILLELA

XSLT:

Manipulação de documentos XML

Monografia de conclusão de curso submetida à Universidade FUMEC como requisito parcial para obtenção do título de Bacharel em Ciência da Computação e aprovada pela seguinte banca examinadora:

_____

Professor Roberlei Panasiewicz (Professor TCC)
Universidade FUMEC

_____

Professor Luiz Eduardo de Mello Gomes (Orientador)
Universidade FUMEC

Belo Horizonte, 26 de junho de 2006

Dedico esta monografia aos professores Carlos Barbieri e Marden Cicarelli. Além de ótimos amigos, são excelentes professores que em mim despertaram o interesse sobre o assunto e, nada mais justo, que a dedicação do meu estudo a estas duas pessoas maravilhosas.

AGRADECIMENTOS

Primeiramente agradeço aos meus pais por terem me dado todo o apoio em toda minha caminhada. Desde o pré-primário até os dias de hoje pude e, ainda posso, contar com todo o apoio e carinho deles.

Agradeço a Alessandra pelo amor e por todo o apoio na realização deste estudo.

Agradeço ao Luiz Eduardo por toda a atenção e dedicação e ao Roberlei Panasiewicz por toda a paciência e prontidão que fizeram possível a conclusão deste estudo.

Agradeço aos demais professores por todo o conhecimento que transmitiram a mim e que me tornou um bacharel em Ciência da Computação.

Finalmente, agradeço a Deus que sempre esteve comigo e que me deu todas as ferramentas para que eu me tornasse quem realmente sou.

Na natureza nada se perde, nada

se cria, tudo se transforma.

ANTOINE LAVOISIER

RESUMO

Com a criação do XML, acreditaram que todos os problemas de comunicação entre aplicações distintas estariam solucionados, todavia isto não ocorreu devido ao simples fato de informações serem distintamente mapeadas. Com a aplicação do XSLT consegue-se criar documentos XML extraindo informações de um outro documento XML e consegue-se também publicar as informações destes XML em formatos lidos por pessoas como o HTML e o PDF. O estudo se inicia pela explicação das tecnologias envolvidas aprofundando no conhecimento de XML e suas seus mecanismos de validação. Com os conceitos entendidos, é iniciado o estudo do XSLT para a integração entre aplicações distintas utilizando a transformação do XML em XML e, por conseguinte, faz-se o estudo sobre a apresentação de dados que, armazenados em documentos XML, transformam-se em formatos facilmente interpretados por pessoas e, finaliza-se o estudo falando sobre o filtro de informações, ou melhor, a filtragem dos dados não úteis dos XML a serem transformados ou apresentados.

PALAVRAS CHAVES: Tecnologia; XSLT; XML; transmissão e modelagem de dados.

LISTA DE FIGURAS

SUMÁRIO

INTRODUÇÃO

Realizando a interseção entre aplicações podemos encontrar um ponto em comum: arquivo texto. Todas as aplicações conseguem ler e escrever arquivos textos e, a partir desta idéia, os maiores arquitetos de sistemas de todo o mundo pensaram em criar algo comum a todas as aplicações, algo que qualquer tipo de aplicação, de qualquer linguagem de programação, de qualquer estilo de programação (seja estruturada como orientada a objetos) consiga trabalhar. Como este “algo comum” é o texto, foi criado o **XML** (Extensible Markup Language) pelo **W3C** (World Wide Web Consortion).

Partindo da idéia acima, o problema de integração de sistemas estaria resolvido. O maior problema ainda estaria por vir, como o XML é uma linguagem altamente extensível, existem milhares de modos de uma mesma informação ser mapeada. Nada impede que uma aplicação represente à mesma informação de um modo diferente de uma outra aplicação, isto é, a aplicação X represente os dados de um cliente com marcações *nome* e *telefone* enquanto que a aplicação Y represente utilizando marcações *nome-cliente* e *telefone-fixo-cliente*.

. E como faremos com dois ou mais XML que possuem a mesma informação ou informações bem similares possam ser entendidos por distintas aplicações? E será que a partir de um XML bem formado e válido que possui varias informações importantes poderemos gerar arquivos HTML ou PDF com as informações bem dispostas utilizando tabelas, cores, imagens e até aplicar filtros, funções ou conversões sobre estes dados?

Para a solução destes tipos de problema, o W3C realizou a criação de um padrão conhecido como **XSLT** (Extensible Stylesheet Language Transform) que é uma linguagem que realiza a manipulação e/ou formatação de um arquivo XML.

A importância do estudo é devido ao fato do XML ser eficaz e muito poderoso e, em conjunto com o XSLT, torna-se uma ferramenta de integração de sistemas ágil e de fácil entendimento. Empresas de qualquer porte poderão ver com XSLT, uma forma mais ágil e eficaz de realizar a comunicação entre sistemas e, com isto, duas tendências são lançadas: melhoria no processo de integração de sistemas e um novo promissor mercado de profissionais em integração de dados.

O objetivo geral desta monografia é demonstrar a utilidade e a importância de uma linguagem de transformação de XML, tal como demonstrar diversas situações em conversões e apresentações de dados que são os cenários em que o XSLT é altamente recomendado devido a sua agilidade e eficácia. Isto será realizado através de pesquisas teóricas a diversos estudiosos no assunto como Bob DuCharme, Doug Tidwell e Michael Kay e, através de práticas para fundamentar e comprovar as funcionalidades propostas por estes autores.

No primeiro capítulo serão expostos os conceitos básicos e introdutórios do XML e do XSLT e será apresentado de forma detalhada como funciona o maior órgão de padronização da internet, o W3C.

No segundo capítulo será exposto como uma informação é estruturada em documentos XML, o porquê de ser mapeada de modo distinto em diferentes sistemas e a diferença entre documentos XML bem formados e válidos, aprofundando nos métodos de validação dos mesmos. Na última seção deste

capítulo, será demonstrada uma solução para que estas informações sejam capazes de serem integradas em outros sistemas.

No terceiro capítulo serão expostos os modos das informações contidas em documentos XML serem convertidas em formatos de mais fácil leitura como HTML, PDF obtendo maior estética para o usuário.

No quarto e último capítulo será exposto como aplicaremos filtro de dados, isto é, filtramos de um documento XML somente os dados relevantes a nossa conversão ou apresentação e, a partir desta idéia, será feita uma analogia do XPath à linguagem SQL por funcionalmente fazerem a mesma coisa.

CAPÍTULO I – XML E XSLT

Baseado no conceito de que tudo que existe nos dias de hoje é fruto de estudos e melhorias sobre aquilo que um dia existiu, este capítulo tem por objetivo demonstrar o porquê da criação do XSLT.

Não há modos de se aprofundar em uma pesquisa tecnológica como esta sem que haja o entendimento de outros conceitos abordados durante esta pesquisa. O que quero dizer é que não existe como falar, por exemplo, sobre mercado de ações sem entender conceitos básicos de economia.

Nos tópicos seguintes falaremos sobre o W3C que é o maior órgão de padronização mundial e introduções ao XML e ao XSLT que são padrões desenvolvidos e especificados por este órgão.

1.1. W3C

World Wide Web Consortium (W3C) é um consórcio de várias empresas mundiais de tecnologia (aproximadamente 500 empresas) fundada por Tim Berners Lee no ano de 1994 com o objetivo de levar a Internet ao seu potencial máximo, através de desenvolvimento de protocolos comuns e fóruns abertos que promovem a sua evolução e garantem a sua interoperabilidade.

O W3C desenvolve tecnologias, ou melhor, padrões para a criação e a interpretação dos conteúdos para a Internet. Sites desenvolvidos segundo esses padrões, podem ser acessados e visualizados por qualquer pessoa ou tecnologia independente do hardware ou do software utilizados, por exemplo, PC com Linux, PDA com Windows CE, entre outros exemplos.

Apesar do W3C não ser muito conhecido no Brasil, seus padrões como Hyper Text Markup Language (HTML), Extensible Hypertext Markup Language (XHTML), Extensible Markup Language (XML) e Cascading Style Sheets (CSS), são muito populares, contudo em muitos casos são usados de forma errônea devido ao não conhecimento da especificação.

É da responsabilidade do desenvolvedor Web respeitar e seguir os padrões deste órgão, pois se assim não fizer, estará impondo barreiras tecnológicas a diversas pessoas e, com isto, desestimulando e até mesmo impedindo o acesso a suas páginas.

1.2. XML

1.2.1. Conceitos e Objetivos

Extensible Markup Language (XML) se tornou uma recomendação da W3C no dia 10 de fevereiro de 1998 cujo objetivo seria a geração de linguagens de marcação para necessidades especiais.

O W3C começou a trabalhar em um projeto de uma linguagem de marcação que combinasse a flexibilidade do SGML (Standard Generalized Markup Language) com a simplicidade do HTML. O principio do projeto era criar uma linguagem que pudesse ser lida por software, e se integrar com as demais linguagens. Sua filosofia seria incorporada por vários princípios importantes:

- **separação do conteúdo e da formatação:** *é uma linguagem cujo aspecto importante é o conteúdo. A formatação poderia ser realizada por uma outra linguagem.*
- **possibilidade de criação de tags[1] sem limitação:** *o uso de tags pode ser altamente utilizado, sem nenhuma limitação, contudo a estrutura deve ser respeitada e seguida.*
- **legibilidade tanto pelas pessoas quanto por máquinas:** a *estrutura do XML é bem completa e ilimitada e, ao mesmo tempo, simples tornando de fácil leitura por pessoas.*
- **criação de arquivos para validação de estrutura:** *arquivos de validação, a serem visto posteriormente neste estudo, permitem verificar se um XML além de bem formado, é também válido de acordo com uma estrutura.*
- **interligação:** *sua utilização permite interligar SGBDs (Sistema de Gerenciamento de Banco de Dados) distintos.*
- **simplicidade**

[1] Tags são etiquetas ou marcações em um texto, ou melhor, um texto importante é inserido dentro de uma tag para assim identificá-lo.

Existem linguagens baseadas em XML como RDF, SMIL, MathML, NCL, XSIL e SVG. O SVG, por exemplo, é um formato gráfico vetorial (graphics scalar vector) e o MathML é uma linguagem de marcação desenvolvida para o aspecto matemático.

1.2.2. HTML e XML

Tanto o XML quanto o HTML são linguagens derivadas do SGML. O que alguns pensam é que o XML foi criado com o propósito de substituir o HTML. Isto não é verdade e, antes de comentarmos o porquê, falaremos um pouco sobre o HTML.

O HTML é uma linguagem de marcação cujo objetivo é produzir páginas na internet. Suas tags consistem em uma marca de início e outra de término, suas marca de início é simbolizada por **<** e sua marca de término por **/>.**

O XML funciona como o HTML, possui tags de início e término, utiliza os mesmos símbolos, todavia não possui a maior característica do HTML: tags pré-estabelecidas. O XML é amplo, as tags são ilimitadas e qualquer tag pode ser criada desde que a estrutura seja mantida.

Enquanto o HTML é mais flexível, permitindo que não se feche algumas das tags, o XML é altamente estruturado. Isto indica que no XML, ao criar uma tag deve-se obrigatoriamente fechá-la, caso contrário, sua estrutura não será validada e o arquivo estará inválido; problema que não ocorre com o HTML no quais algumas de

suas tags geralmente não são fechadas ou possuem alguma flexibilidade de estruturação.

1.3. XSLT

Extensible Stylesheet Language Transformations (XSLT) é um dos padrões criados pelo W3C cuja finalidade seria a de transformar documentos XML em outros documentos como pode se ver na figura 1.

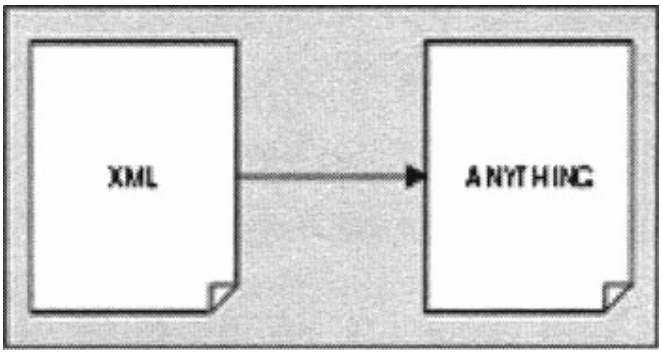


Figura 1: Funcionalidade do XSLT
Fonte: KAY, 2004, p. 18.

A partir desta idéia, podemos pensar no XSLT como uma linguagem de transformação que, a partir de um documento XML gera como saída qualquer outro documento.

> XSLT (que se entende como Extensible Stylesheet Language: Transformations) é uma linguagem que, de acordo com o primeiro parágrafo da especificação [...], é primariamente desenvolvida para transformar um

> documento XML em outro. Entretanto, XSLT é também capaz de transformar XML em HTML e qualquer outro formato baseado em texto. Então a melhor definição geral deveria ser a seguinte:
> *XSLT* é uma linguagem para transformar a estrutura e o conteúdo de um documento XML. (KAY, 2004, p. 01. Tradução nossa).

Segundo DuCharme (2001), o XSLT é um padrão, todavia este fato não significa que seja uma boa linguagem de transformação, mas sim que muitos fabricantes trabalharam juntos e cada um contribuiu para seu projeto e comprometeu a usá-lo em seus produtos. Existem outras maneiras de se transformar documentos XML, e ainda na visão de DuCharme, são divididas em duas categorias:

- XML relacionadas a bibliotecas adicionadas a linguagens de programação de finalidade geral como Java, Perl, Visual Basic, Python e C++;
- linguagens como OMNImark e Balise, projetadas especificamente para manipular documentos XML (e tipicamente SGML).

> O XSLT era originalmente parte da Extensible Stylesheet Language (XSL). De fato o XSLT continua tecnicamente parte da XSL. A especificação descreve XSL como uma linguagem com duas partes: uma linguagem para transformar documentos XML e um vocabulário XML para descrever como formatar conteúdo de documentos. Esse vocábulo é uma coleção de elementos especializados chamados “objetos de formatação”, que especificam o layout de páginas e outros detalhes relativos à apresentação de textos marcados com essas tags de elementos: família de cores, tamanho de fonte, margens, espaço entre linhas e outras configurações. (DUCHARME, 2001, p. 04-05).

Existe a confusão entre o Cascading Style Sheets (CSS) com o XSLT. Não existe nenhum conflito entre eles, pois foram criados para propósitos distintos. Serão vistos, no tópico seguinte, mais detalhes sobre o que é cada um deles e no que se diferenciam.

1.3.1. XSLT e CSS

Enquanto o Cascading Style Sheets (CSS) é “um mecanismo utilizado para definir várias propriedades de elementos de marcação” (TIDWELL, 2001, p. 02. Tradução nossa), o XSLT é um mecanismo “para definir regras para transformação de um documento XML”. (TIDWELL, 2001, p. 02. Tradução nossa).

O CSS pode ser utilizado com XML, mas o que é observado na prática e o uso do estilo CSS com documentos HTML. Você pode utilizar o CSS para dizer que um elemento deve ser apresentado (renderizado) em cor azul, em negrito, entre outras formatações visuais. Segundo TIDWELL (2001) existem muitas coisas que o CSS não faz e não foi desenvolvido para fazer:

- CSS não altera a ordem que os elementos aparecem no documento. Se desejar ordenar elementos segundo alguma propriedade ou filtrar alguns elementos, o CSS não realiza esta tarefa.
- CSS não realiza cálculos. Se você desejar calcular uma saída, por exemplo, um somatório dos salários de um departamento específico e depois exibí-lo, o CSS não realiza esta tarefa.
- CSS não combina múltiplos documentos. Se você quiser combinar nove documentos de nota fiscal e imprimir o sumário de todos os itens destas notas fiscais, o CSS não realiza esta tarefa.

Estas observações acima não podem ser vistas como uma crítica ao CSS. Como já foi dito, o XSLT e o CSS foram desenvolvidos para propósitos distintos.

“Um uso razoavelmente comum de XSLT é para gerar um documento HTML que contém um elemento CSS”. (TIDWELL, 2001, p. 02. Tradução nossa).

1.3.2. Características, objetivos e cenários aplicáveis

Segundo Tidwell (2001), o XSLT foi criado para ser a mais poderosa e flexível linguagem para transformar documentos.

Ainda na visão deste mesmo autor citaremos alguns objetivos e características específicas do XSLT:

- O estilo XSLT deve ser um documento XML. Isto significa que você pode escrever um estilo que transforma um segundo estilo em outro estilo. Este tipo de pensamento recursivo é comum no XSLT.
- A linguagem XSLT deve ser baseada em combinações de padrões. A maioria de seus estilos consiste em regras[2] usadas para transformar um documento. “Cada regra diz, “Quando você observar uma parte do teste que pareça com isto, aqui você deve converter isto, naquilo.” Isto é provavelmente diferente de qualquer programação que você já tenha feito.” (TIDWELL, 2001, p. 02. Tradução nossa).
- XSLT deve ser desenvolvida para ser livre de efeitos laterais. Em outras palavras, XSLT é desenvolvida para que várias diferentes regras de estilo possam ser aplicadas simultaneamente. O maior impacto

[2] Regras são conhecidas como **templates** em XLST.

disto é que as variáveis não podem ser modificadas. Uma vez iniciada a variável, você não poderá alterar seu valor; se as variáveis pudessem ser alteradas, então o processamento de uma regra de estilo poderia gerar efeitos laterais, que impactaria outras regras de estilo.

> XSLT é altamente influenciada pelo projeto de *linguagens de programação funcional*, tal como Lisp, Scheme e Haskell. Estas linguagens também possuem variáveis imutáveis como característica. Ao invés de definir regras como o XSLT, linguagens de programação funcional definem programas como uma série de funções, cada uma delas gera uma saída bem definida (livre de efeitos laterais, com certeza) em resposta a uma entrada bem definida. O objetivo é executar as instruções de uma dada regra XSLT sem afetar a execução de qualquer outra regra. (TIDWELL, 2001, p. 03. Tradução nossa).

- Ao invés de looping[3], XSLT utiliza interações e recursões. Como as variáveis não podem ser modificadas, não teria como realizar um for ou um do-while que são estruturas de repetição. XSLT utiliza duas técnicas equivalentes: iteração e recursão.

> Iteração significa que você pode escrever uma regra XSLT que diga, “pegue todas as coisas que pareça com isto, e é aqui que eu quero fazer com cada uma delas”. Embora isto seja diferente de um laço do-while, geralmente o que você faz em uma linguagem procedural é algo como, “faça isto enquanto existir itens para processar”. Neste caso, iteração funciona exatamente como você deseja.
> Recursão exige muito prática para utilizá-la. Se você implementa algo como uma estrutura for (for i=1 to 10 do, por exemplo), recursão é o caminho a seguir. (TIDWELL, 2001, p. 03. Tradução nossa).

Acima observamos várias características e objetivos do XSLT. Estas informações são altamente relevantes, pois nos faz pensar na seguinte questão: Onde poderemos utilizar esta tecnologia? E, ainda a partir das idéias do Tidwell, cito:

---

[3] Looping é uma estrutura que realiza laços de repetição de um código

- Seu website necessita entregar informações para uma variedade de dispositivos, isto é, computadores, celulares, notebooks, palmtops, entre outros dispositivos. Seria excelente disponibilizar estas informações em documentos estruturados e, então transformá-los em qualquer formato que necessite.
- Você necessita trocar informações de uma de suas empresas com outras, mas você utiliza diferentes sistemas de banco de dados. Seria excelente se pudesse definir um XML comum de formato de dado e, então transformá-lo no arquivo de importação que necessita (SQLs, comma-separated values[4], entre outros formatos de importação de dados existentes).
- Informação XML de um outro sistema deve ter sua estrutura modificada para que seja armazenada no sistema local.
- Sistemas como J2EE, .NET, Natural Web pode responder a uma requisição de um usuário com uma página XML e, este XML ser convertido em HTML na máquina-cliente.

Como pôde ser visto, existem vários cenários no qual a aplicação do XSLT é altamente apropriada. Estes vários cenários, na visão de Kay (2004) podem ser descritos simplesmente em dois cenários principais, que serão trabalhados nos capítulos seguintes: Conversão e apresentação (publicação) de dados.

Em uma visão altamente superficial cujo propósito é expor o que será visto nos capítulos seguintes podemos definir como conversão de dados, toda situação

---

[4] Comma-separated values é um formato de importação de registros para SGBD no qual a separação dos valores é realizado através de vírgulas (,).

em que você necessita de converter um XML em um outro XML, com as mesmas ou a grande maioria das informações em uma estrutura igual ou diferente para uma outra aplicação. Em contrapartida, podemos definir como apresentação de dados, toda situação em que você, a partir de um documento XML, necessite publicar seu conteúdo de uma forma mais apresentável, desde HTML, PDF até gráficos, músicas (MIDI), etc.

CAPÍTULO II – TRANSMITINDO DADOS ENTRE APLICAÇÕES

A transmissão de dados entre aplicações distintas é um problema atual e grave. Diferentes aplicações podem realizar a mesma função, isto é, fazer as mesmas coisas como, por exemplo, gerar folha de pagamento, cadastrar funcionários, gerenciar contas a pagar e receber, entre outras funções. O XML foi criado com o intuito de criar um “protocolo” comum de comunicação, mas iremos destacar neste capítulo o porquê do XML não solucionar completamente o problema da comunicação de dados entre aplicações e onde o XSLT pode ser inserido a fim de resolver tais problemas.

2.1. Modelagem de dados

As aplicações podem realizar a mesma funcionalidade embora, conforme as idéias de Braganholo e Heuser, seja completamente provável que os dados armazenados estejam modelados de forma diferente. Ao citar que a modelagem está diferente não se refere aos campos com nomes distintos, mas sim, a todos os seguintes casos:

- Nomenclatura distinta de campos;
- Tipos distintos;
- Gravação em formatos distintos;

- Modos de relacionamento entre os dados feitos de diferentes formas.

Exemplificando os três primeiros casos acima, posso citar o armazenamento de um CPF (cadastro de pessoa física) de cliente. Ele pode ser armazenado com o nome de NUM_CPF, ser do tipo numérico ou pode ser armazenado com o nome de CPF_CLIENTE, ser do tipo alfanumérico e ser armazenado com máscara. Neste caso, a mesma informação pode ser extraída de ambas as bases de dados com o entendimento de forma intuitiva, mas, internamente, foram modeladas de forma distinta.

Baseado no problema acima, pode-se entender o real significado da palavra intuição embora o computador não possua este entendimento. O sistema que trabalha com o CPF na forma numérica não conseguiria abstrair uma informação de um XML bem formado e válido cujo campo de CPF seja alfanumérico; É neste ponto que a utilização do XSLT é funcionalmente adequada e, antes de entrarmos no tópico de como fazer esta conversão utilizando XSLT falaremos um pouco sobre o que é um documento XML bem formado e válido, destacando os tipos de validações existentes.

2.2. XML bem formado e válido

Segundo Myer (2005), existem dois tipos de “legalidade” de documentos XML: documento bem formado e documento válido.

> Um documento XML bem formado segue as seguintes regras [...]:
> - Um documento XML deve conter um único elemento raiz que contenham outros elementos.
> - Todos os elementos devem ser corretamente aninhados.
> - Todos os elementos devem ser fechados com uma tag ou com um "auto-fechamento" tag elemento vazia. (por exemplo: <tag/>).
> - Todos os valores de atributos devem ser citados.
>
> Um documento XML válido é tanto bem formado quanto segue todas as regras explicitadas em um documento DTD (document type definition). Um documento válido, então, é nada mais que um documento bem formado vinculado a seu DTD. (MYER, 2005. Tradução nossa).

Em síntese, podemos definir um documento XML bem formado como aquele documento XML que segue todas as definições do W3C para seu desenvolvimento e, documento XML válido aquele documento XML bem formado que está vinculado a um documento DTD ou a uma XML Schema (Esquema XML) de validação.

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Figura 2: Estrutura de um XML bem formado.
Fonte: REFSNES, 2006.

Na figura 2, é demonstrada a estrutura de um XML bem formado e na seção seguinte veremos como vinculá-lo a uma validação DTD e a um XML Schema.

2.2.1. Validação de XML

A validação de um XML pode ser realizada tanto por um documento DTD como por um XML Schema.

O DTD foi criado antes do XML Schema. “O propósito de um documento DTD é definir os blocos de construção legais de um documento XML. Ele define a estrutura do documento com uma lista de elementos legais. Um DTD pode ser declarado dentro do seu documento XML, ou como uma referência externa”. (REFSNES, 2006. Tradução nossa). Mas porquê utilizar o DTD?

> Com o DTD, cada um de seus documentos XML podem carregar, neles próprios, uma descrição com seu próprio formato.
> Com o DTD, grupos independentes de pessoas podem aceitar utilizar um DTD comum para compartilhar dados.
> Sua aplicação pode utilizar um DTD padrão para verificar se o dado que está recebendo do meio externo é válido.
> Você também pode utilizar o DTD para verificar seus próprios dados.
> (INTRODUCTION TO DTD, W3C. Tradução nossa).

Conforme W3C, o XML Schema tem exatamente o mesmo propósito que o DTD, porém com o intuito de ser a sucessora do DTD. O XML Schema como toda nova tecnologia que possui a intenção de substituir uma tecnologia legada, apresenta-se como mais poderosa e eficaz.

> Nós pensamos que tão logo XML Schemas serão utilizados na maioria das aplicações Web como uma substituição aos DTDs. Estas são algumas razões:
> - XML Schemas são extensíveis para adições futuras.
> - XML Schemas são mais ricos e mais úteis que os DTDs.
> - XML Schemas são escritos em XML

- XML Schemas suportam tipos de dados
- XML Schemas suportam namespaces[5]

(INTRODUCTION TO XML SCHEMA, W3C, 2006. Tradução nossa).

O mais interessante do XML Schema é o suporte aos tipos de dados. O DTD somente consegue validar se é um tipo binário (PCDATA) ou tipo alfanumérico (CDATA) enquanto o XML Schema possui vários tipos de dados o que o torna mais ricos e com validações mais úteis. Por ser escrito em formato XML pode ser transformado em outro XML Schema utilizando o XSLT conferindo a ele a extensibilidade citada pelo W3C. Observando a quantidade de novos recursos que o XML Schema oferece, conclui-se que o XML Schema sucederá o DTD em pouco tempo. Veremos na Figura 3, um exemplo demonstrando um XML vinculado a uma validação DTD e uma validação pelo XML Schema.

Figura 3: DTD e XML Schema
Fonte: INTRODUCTION TO XML SCHEMA, 2006.

[5] "Um namespace XML é um padrão W3C para prover elementos e atributos com nomeação única." (NAMESPACE..., 2006).

Conclui-se que a validação por DTD possui menos recursos enquanto que pelo XML Schema é bem mais rica, mas pode-se também observar que a validação por DTD é bem mais simples e fácil, ao contrário do XML Schema que é um pouco mais difícil, ou melhor, mais avançada.

2.3. Conversão de dados em XML

Como visto nas seções anteriores percebe-se que existem diversos modos de representar uma mesma informação utilizando XML; e que um documento XML geralmente está vinculado a um DTD ou XML Schema, que a partir deste ponto trataremos que esteja vinculado a um esquema independente se este for um DTD ou um XML Schema.

Uma das vantagens do XSLT é “converter documentos XML, de acordo com um esquema em documentos de conformidade com outros, tornando muito mais fácil passar informações de cá e para lá em sistemas diferentes.” (DuCharme, 2001). O objetivo deste capítulo é demonstrar a utilização do XSLT para transformar um documento XML válido em uma estrutura em um documento XML válido em uma outra estrutura.

> Somente por todos estarem utilizando o XML não significa que a necessidade por conversão de dados desaparecerá. Sempre existirão vários padrões em uso. Por exemplo, uma indústria de jornal provavelmente utiliza formatos diferentes para trocar novos artigos para o formato utilizado na indústria de televisão. Igualmente, sempre haverá uma necessidade para fazer algo como extrair um endereço de uma ordem de compra e adicionar ele a uma fatura. Então a ligação entre as empresas para realizar o

> comércio eletrônico está, cada vez mais, tornando um caso para definir como extrair e combinar dados de um grupo de documentos XML para gerar outro grupo de documentos XML: o XSLT é a ferramenta ideal para fazer este trabalho. (KAY, 2000, p.16. Tradução nossa).

Segundo Kay (2000), o uso de XSLT para transformar documentos XML em outros documentos XML ocorre freqüentemente no comércio eletrônico devido à necessidade de transferir informações usando documentos XML. Na visão do mesmo autor, muitas das necessidades de transformações ocorrem para a adequação estrutural do documento.

Analisando um caso bem simples em que possuímos um documento XML cujas informações do cliente (*customer*) estão inseridas em atributos da própria marcação (conforme pode ser observado na figura 4) e que seja necessária a conversão deste XML em um outro que seja válido segundo o esquema de uma outra aplicação.

```
<?xml version="1.0" encoding="utf-8" ?>
- <customers>
    <customer CustomerID="ALFKI" CompanyName="Alfreds Futterkiste"
      ContactName="Maria Anders" ContactTitle="Sales
      Representative" Address="Obere Str. 57" City="Berlin"
      PostalCode="12209" Country="Germany" Phone="030-0074321"
      Fax="030-0076545" />
    <customer CustomerID="ANATR" CompanyName="Ana Trujillo
      Emparedados y helados" ContactName="Ana Trujillo"
      ContactTitle="Owner" Address="Avda. de la Constitución 2222"
      City="México D.F." PostalCode="05021" Country="Mexico"
      Phone="(5) 555-4729" Fax="(5) 555-3745" />
  </customers>
```

Figura 4: XML de clientes
Fonte: XML TO XML WITH XSLT, 2006.

Nesta outra aplicação todas as informações do cliente estão em marcações dentro da marcação cliente (*customer)*. Para tal deve-se criar uma folha de estilos XSLT que iria entrar no cliente e para cada um de seus atributos criará uma

marcação com o mesmo nome e com o mesmo valor original como pode ser visto na Figura 5.

```
<?xml version='1.0' encoding='utf-8' ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
   <xsl:output method="xml"/>
   <xsl:template match="/">
      <customers>
         <xsl:for-each select="/customers/customer">
            <xsl:element name="{name()}">
               <xsl:for-each select="@*">
                  <xsl:element name="{name()}">
                     <xsl:value-of select="."/>
                  </xsl:element>
               </xsl:for-each>
            </xsl:element>
         </xsl:for-each>
      </customers>
   </xsl:template>
</xsl:stylesheet>
```

Figura 5: XSLT de clientes
Fonte: XML TO XML WITH XSLT, 2006.

O documento XML original (que possui as informações em atributos) ao ser vinculado a folha de estilos XSLT obteria como resultado um outro documento XML válido para esta outra aplicação conforme pode ser observado na figura 6.

Então, a partir de um documento XML seja em qualquer estrutura que contenha informações úteis (relevantes), podemos criar uma folha de estilos XSLT que, ao ser vinculada ao XML original, geraria um documento XML somente com os dados úteis em uma estrutura igual ou distinta a de origem, embora agora, fosse aceita por esta outra aplicação.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <customers>
  - <customer>
      <CustomerID>ALFKI</CustomerID>
      <CompanyName>Alfreds Futterkiste</CompanyName>
      <ContactName>Maria Anders</ContactName>
      <ContactTitle>Sales Representative</ContactTitle>
      <Address>Obere Str. 57</Address>
      <City>Berlin</City>
      <PostalCode>12209</PostalCode>
      <Country>Germany</Country>
      <Phone>030-0074321</Phone>
      <Fax>030-0076545</Fax>
    </customer>
  - <customer>
      <CustomerID>ANATR</CustomerID>
      <CompanyName>Ana Trujillo Emparedados y helados</CompanyName>
      <ContactName>Ana Trujillo</ContactName>
      <ContactTitle>Owner</ContactTitle>
      <Address>Avda. de la Constitución 2222</Address>
      <City>México D.F.</City>
      <PostalCode>05021</PostalCode>
      <Country>Mexico</Country>
      <Phone>(5) 555-4729</Phone>
      <Fax>(5) 555-3745</Fax>
    </customer>
  </customers>
```

Figura 6: XML resultante da aplicação da folha de estilo XSLT
Fonte: XML TO XML WITH XSLT, 2006.

Como pôde ser observado, o problema de transmitir dados entre as aplicações utilizando XML em diferentes estruturas possui o XSLT como a ferramenta ideal para a solução.

CAPÍTULO III – SEPARANDO OS DADOS PARA APRESENTAÇÃO

No capítulo anterior, pôde ser visto a aplicação do XSLT para a transformação de documentos XML em diferentes estruturas. O objetivo deste capítulo é estudarmos a utilização do XSLT para não somente para converter XML para XML, mas sim, para transformar um documento XML em qualquer outro documento como foi citado no início deste estudo.

Aproveita-se neste capítulo para superficialmente exemplificar o funcionamento do XSLT para converter os documentos XML e qual a finalidade de cada uma das funções que foram abordadas do capítulo anterior e que voltarão, neste, a ser abordadas como, por exemplo, *xsl:output*, *xsl:templante*, *xsl:apply-templantes*, *xsl:value-of* que são funções simples e básicas. É importante ressaltar que ser função básica e simples não signifique que não sejam poderosas.

3.1. Formatos para apresentação

Segundo a primeira especificação, o XSLT foi criado em sua com o objetivo de transformar XML em XML, HTML ou texto, já na segunda especificação foi inserido a transformação para XHTML.

Mas segundo Kay (2004) e também na visão de Tidwell (2000), o XSLT é uma linguagem poderosa e flexível para transformar documentos XML em qualquer outra coisa.

> Esta outra coisa pode ser um documento HTML, outro documento XML, um formato de documento portável (PDF), um gráfico de vetor escalar (SVG), uma linguagem modelagem realmente virtual (VRML), código Java, um texto liso, uma imagem JPEG, ou qualquer outro formato que você deseje. (TIDWELL, 2000, p. 01. Tradução nossa).

Documentos XML, HTML, texto, tudo pode ser criado a partir de um documento XML. Na verdade, “você escreve uma folha de estilo XSLT para definir as regras para transformar um documento XML e o processador XSLT faz o trabalho.” (TIDWELL, 2000, p. 01. Tradução nossa).

3.2. Apresentação de dados

3.2.1. Conceito

A apresentação de dados se distingue da conversão de dados pelo fato de que a conversão de dados faz com que o computador entenda a transformação enquanto que a apresentação de dados ou publicação de dados é realizada com o intuito de tornar o documento XML entendível para as pessoas.

> A diferença entre conversão de dados e publicação está caso anterior, o dado é destinado para a entrada de software de outra parte, enquanto que neste caso ela é destinada para ser lida (você espera) por seres humanos. Publicação neste contexto não significa somente textos e multimídias pródigos, também significa dados: tudo da atividade ou produção tradicional

> ou dos relatórios distribuídos nos quais os gerentes saibam como o negócio está indo, para produzir faturas telefônicas ou boletas bancárias para os clientes e quadro de horários de estação ferroviária para o público geral. XML é o ideal para tais aplicações de publicação de dados, como também é o formato mais tradicional de publicação de texto que é o campo original do SGML. (KAY, 2004, p. 19. Tradução nossa).

A disponibilização de detalhes de apresentação em documentos XML não é uma má prática; Disponibilizar detalhes em documento XML quanto qualquer outro tipo de informação é, do mesmo modo, correto. Quando os dados a serem publicados estão em um formato padrão como é o XML, torna-se bem mais fácil realizar qualquer tipo de formatação ainda mais se utilizar o XSLT, em outros formatos teria a necessidade de utilizar aplicativos proprietários.

> XML foi projetado para permitir que as informações sejam armazenadas independente da maneira como ela é apresentada, o que conduz às vezes ouvirmos falácias de pessoas que pensam que utilizar XML para detalhes de apresentação é algo ruim. [...] Os detalhes da apresentação são tão certos serem codificados em XML quanto qualquer outro tipo de informação. Então, nos podemos ver o papel do XSLT no processo de publicação como convertendo dados sem apresentação em dados com apresentação, ambos são, no primeiro momento, formatos XML. (KAY, 2004, p. 19. Tradução nossa).

Mas qual a importância da publicação de documentos XML? Existem, nos dias de hoje, duas principais formas de publicação de informações que são os jornais, revistas, periódicos, ou melhor, impressões em papel e a internet. Imagine um exemplo de uma empresa que atue no meio jornalístico (jornal em papel), periódico (revistas) e eletrônico (website de notícias); Esta empresa poderia, a cada notícia, criar um único documento XML possuindo marcações como título, assunto, palavra-chave, entre outras e, apenas publicar este XML de um modo diferente para cada ambiente, isto é, utilizar distintas folhas de estilo XSLT para cada um dos ambientes.

> Os dois importantes veículos para publicação de informação nos dias de hoje são impressões em papel e a internet. O cenário de impressões em papel é mais difícil devido à alta expectativa dos usuários para a qualidade visual. Os objetos de formatação XSL tentam definir um modelo baseado em XML de um arquivo de impressão para exibição em alta qualidade no papel ou na tela. Devido ao número completo de parâmetros necessários para conseguir isto, o padrão irá levar um tempo para chegar à maturidade. Mas a Internet é um ambiente de menor exigência onde tudo que nós necessitamos fazer é converter os dados para HTML e deixar o navegador fazer o melhor que pode na exibição disponível. Converter XML to HTML é a mais comum aplicação de XSLT nos dias de hoje. Ele é atualmente um processo de dois estágios: primeiro converte um modelo baseado em XML que seja estruturalmente equivalente ao HTML desejado, e então o serializa[6] em notação HTML melhor que a do rigoroso XML. (KAY, 2004, p. 19. Tradução nossa).

Portanto a apresentação de dados utilizando o XSLT é bem mais utilizada que a conversão de dados entre aplicações. O objetivo desta seção é demonstrar a utilidade do XSLT para a apresentação de dados de documentos XML ao mesmo tempo em que incentiva o uso de documento XML para armazenamento de informações. Na seção seguinte veremos algumas práticas de apresentação de dados em diferentes formatos como o HTML e também em outros formatos como PDF e, até mesmo, SGV.

3.2.2. Exemplificação

Nada melhor que exemplos práticos do XSLT funcionando para demonstrar a utilidade e o poder do XSLT. A partir de um documento XML único cuja única

[6] “No moderno contexto da ciência da computação, serialização é o processo de salvar um objeto em um armazenamento médio (como um arquivo, ou um espaço de memória) ou transmitir um objeto através de uma conexão à internet como um soquete em uma série de bytes ou em algum formato interpretável por pessoas como o XML.” (SERIALIZATION, 2006. Tradução nossa).

marcação é uma chamada *greeting* (cumprimento) e cujo valor seja ‘Hello, Terra!’ conforme podemos ver na figura 7.

```
<?xml version="1.0"?>
<greeting>
   Hello, Terra!
</greeting>
```

Figura 7: XML Simples
Fonte: TIDWEEL, 2000, p. 21. (Texto adaptado)

Iremos aplicar um simples estilo XSLT (figura 8) que irá transformar o documento XML em um formato HTML.

```
1  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
2     <xsl:output method="html"/>
3
4     <xsl:template match="/">
5        <xsl:apply-templates  select="greeting"/>
6     </xsl:template>
7
8     <xsl:template match="greeting">
9        <html>
10          <body>
11             <h1>
12                <xsl:value-of select="."/>
13             </h1>
14          </body>
15       </html>
16    </xsl:template>
17 </xsl:stylesheet>
```

Figura 8: XSLT simples de transformação para HTML
Fonte: TIDWEEL, 2000, p. 22.

A saída esperada e obtida com a aplicação do estilo XSLT (figura 8) ao XML (figura 7) pode ser vista na figura 9. Nesta figura estão sendo exibidos tanto o código do resultado como o resultado visual no navegador.

```
<html>
   <body>
      <h1>Hello, Terra!</h1>
   </body>
</html>
```

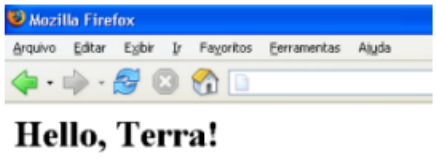


Figura 9: Resultado HTML
Fonte: TIDWEEL, 2000, p. 23. (Texto adaptado)

Segundo Tidwell (2000), vamos agora ver como o processador XSLT transformou o documento XML.

1. O XSLT é analisado gramaticalmente (*parse*) e convertido em uma estrutura de árvore.
2. O documento XML é também analisado gramaticalmente (*parse*) e convertido em uma estrutura de árvore.
3. O processador XSLT está agora na raiz (*root*) do documento XML. Este é o contexto original.
4. Existem um *xsl:templante* que combina o documento raiz (*root*). Observe linhas 4 a 6 da figura 8.

   Uma única barra (/) é uma expressão XPath que significa “a raiz do documento XML”.
5. Agora o processo começa outra vez dentro do *xsl:templante*. A única instrução é aplicar um *xsl:templante* em qualquer elemento *greeting* do contexto corrente. O contexto corrente dentro da folha de estilos é definido pelo atributo *match*. Isto significa que o processador XSLT estará procurando por qualquer elemento *greeting* no documento raiz.

   Por apenas existir um documento *greeting* na raiz do documento, o processador XSLT deve tratá-lo. (Se existissem mais de um elemento *greeting*, o processador XSLT trataria de cada um deles na ordem como aparece no documento). Encontrando o elemento *greeting*, o *xsl:templante* então aplicará a ele o segundo *xsl:templante* (observe linhas 8 a 16 da figura 8).

6. Ao entrar no *xsl:templante* do elemento *greeting* serão escritos na saída os três primeiros elementos (<html>, <body> e <h1>).

   O elemento xsl:value-of (linha 12 da figura 8) escreve o valor de algo na saída. No nosso caso estamos utilizando a expressão XPath, que é representada por um simples ponto, para indicar o nó corrente, ou melhor, exibirá todo o texto da marcação *greeting*.

   Depois serão exibidos os três últimos elementos (</h1>,</body> e </html>) e como não existe mais o que fazer neste *templante*, então será encerrado.
7. Agora voltamos ao *templante* do elemento raiz. Como processamos todos os elementos *greeting*, então finalizaremos também com este *template*.
8. Como não se existe mais nenhum elemento, o processador XSLT é encerrado.

Agora que sabemos o que realmente ocorre no processamento de uma folha de estilos XSLT veremos mais duas folhas de estilo XSLT que realizam a transformação do mesmo XML em formatos PDF e SGV.

Ainda segundo Tidwell (2000), para converter um XML para um arquivo PDF, deve-se inicialmente converter o XML em um arquivo de formatação de objetos, este arquivo segue a Extensible Stylesheet Language for Formatting Objects (XSL-FO) que é um vocabulário XML que descreve o conteúdo a ser renderizado. Como existe XML padrões como o MathML (matemática), ChemicalML (química), VoiceXML (áudio), existe também o XSL-FO para objetos de formatação.

A figura 10 exibe a folha de estilos XSLT que converte o XML em um outro XML cujo formato é padrão (XSL-FO).

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <xsl:output method="xml"/>

    <xsl:template match="/">
        <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
            <fo:layout-master-set>
                <fo:simple-page-master margin-right="75pt" margin-left="75pt" page-height="11in"
                    page-width="8.5in" margin-bottom="25pt" margin-top="25pt" master-name="main">
                    <fo:region-before extent="25pt"/>
                    <fo:region-body margin-top="50pt" margin-bottom="50pt"/>
                    <fo:region-after extent="25pt"/>
                </fo:simple-page-master>
                <fo:page-sequence-master master-name="standard">
                    <fo:repeatable-page-master-alternatives>
                        <fo:conditional-page-master-reference master-reference="main" odd-or-even="any"/>
                    </fo:repeatable-page-master-alternatives>
                </fo:page-sequence-master>
            </fo:layout-master-set>

            <fo:page-sequence master-reference="standard">
                <fo:flow flow-name="xsl-region-body">
                    <xsl:apply-templates select="greeting"/>
                </fo:flow>
            </fo:page-sequence>
        </fo:root>
    </xsl:template>

    <xsl:template match="greeting">
        <fo:block line-height="40pt" font-size="36pt" text-align="center">
            <xsl:value-of select="."/>
        </fo:block>
    </xsl:template>

</xsl:stylesheet>
```

Figura 10: XSLT de conversão para XSL-FO
Fonte: TIDWEEL, 2000, p. 34-35.

O formato resultado por este XSLT (figura 10) é um XML no padrão XSL-FO facilmente convertido para PDF por qualquer aplicativo no mercado, exemplo de aplicativo é o Apache FOP que faz isto transparentemente. Após rodar o aplicativo com o resultado da transformação feita pelo XSLT, o arquivo PDF ficaria como na figura 11.

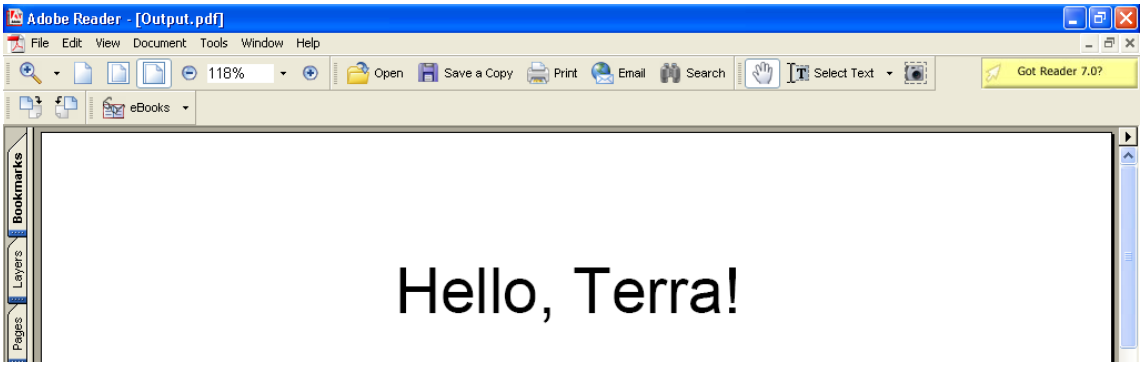


Figura 11: Resultado PDF
Fonte: TIDWEEL, 2000, p. 36. (Texto adaptado)

A partir de mesmo XML (figura 7) podemos gerar um arquivo SGV que é um arquivo de gráficos vetoriais escaláveis através de uma folha de estilo como mostrada na figura 12.

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="xml"
   doctype-public="-//W3C//DTD SVG 20001102//EN"
   doctype-system="http://www.w3.org/TR/2000/CR-SVG-20001102/DTD/svg-20001102.dtd"/>

 <xsl:template match="/">
  <svg width="10cm" height="4cm">
   <g>
    <defs>
     <radialGradient id="MyGradient" cx="4cm" cy="2cm" r="3cm" fx="4cm" fy="2cm">
      <stop offset="0%" style="stop-color:red"/>
      <stop offset="50%" style="stop-color:blue"/>
      <stop offset="100%" style="stop-color:red"/>
     </radialGradient>
    </defs>
    <rect style="fill:url(#MyGradient); stroke:black" x="1cm" y="1cm" width="8cm" height="2cm"/>
    <text x="5.05cm" y="2.25cm" text-anchor="middle" style="font-family:Verdana; font-size:24;font-weight:bold; fill:black">
     <xsl:apply-templates select="greeting"/>
    </text>
    <text x="5cm" y="2.2cm" text-anchor="middle" style="font-family:Verdana; font-size:24;font-weight:bold; fill:white">
     <xsl:apply-templates select="greeting"/>
    </text>
   </g>
  </svg>
 </xsl:template>

 <xsl:template match="greeting">
  <xsl:value-of select="."/>
 </xsl:template>

</xsl:stylesheet>
```

Figura 12: XSLT de conversão para SGV
Fonte: TIDWEEL, 2000, p. 21-22.

Após a aplicação da folha de estilos obtém-se um gráfico vetorial como o ilustrado na figura 13.

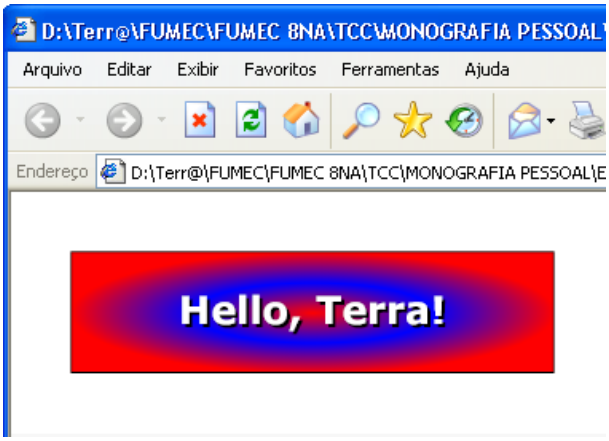


Figura 13: Resultado SVG
Fonte: TIDWEEL, 2000, p. 40. (Texto adaptado)

Pode-se concluir neste capítulo que o conceito de transformação de XML para qualquer outro documento realmente funciona e é eficaz. Conclui-se também que o XSLT possui um território vasto para atuação e que quanto maior a utilização do XML para o armazenamento de dados, maior será a necessidade do uso do XSLT para a conversão de dados e/ou sua publicação.

CAPÍTULO IV – FILTRO DE DADOS

O objetivo neste último capítulo é demonstrar a utilização de uma sub-linguagem do XSLT conhecida como XPath para a realização de filtros sobre as informações armazenadas em um documento XML e, até mesmo, cálculos numéricos, manipulação de string, teste de condições booleanas[7], entre outras úteis funções que podem ser aplicadas.

4.1. XSLT e SQL

Já existe a utilização do XSLT para formatar resultados em documentos XML retornados por um SGBD[8]. O SGBD retorna a tabela de resultados em um XML bem formado no qual, geralmente, as marcações possuem os mesmos nomes da coluna. A utilização do XSLT neste caso é bastante eficiente, embora esta não seja o foco do estudo desta seção.

Nesta seção não trabalharemos sobre esta utilização do XSLT, mas sim, comparar os conceitos sobre a exibição e/ou formatação somente dos dados que se deseja, com a idéia do SQL que seria apenas trazer os dados que se deseja.

---

[7] "Um valor booleano é um valor verdade, seja verdade ou falso, respectivamente codificado por 1 e 0, respectivamente." (BOOLEAN, 2006. Tradução nossa).

[8] "Um sistema gerenciador de banco de dados (SGBD) é o conjunto de programas de computador (softwares) responsáveis pelo gerenciamento de uma base de dados." (SISTEMA DE GERENCIAMENTO..., 2006).

> Superficialmente, SQL e XSLT são linguagens muito diferentes. Mas se você olhar abaixo do superficialmente, elas realmente possuem muito em comum. Primeiramente, para se processar dados específicos, seja em uma base de dados relacional ou em um documento XML, a linguagem de processamento deve incorporar uma sintaxe declarativa de busca para selecionar apenas os dados que necessitam ser processados. Em SQL, esta é a declaração SELECT. Em XSLT, a equivalente é a expressão XPath. (KAY, 2004, p. 07. Tradução nossa).

Segundo Kay (2004), existem várias similaridades entre o XSLT e o SQL:

- Ambas as linguagens aumentam a sintaxe básica de busca adicionando comandos aritméticos, manipulações de string e operações de comparação.
- Ambas as linguagens suplementam a sintaxe básica de busca com facilidades semi-procedurais para descrever o processo a ser executado.
- Ambas as linguagens possuem uma importante propriedade chamada *fechamento* que significa que a saída tem a mesma estrutura de dados da entrada. Para SQL, esta estrutura são tabelas, para XSLT são árvores – a representação de árvore de documentos XML.

Entendendo que, no mundo real, o XSLT e SQL não coexistem, mas que existem vários relacionamentos possíveis entre eles, poderemos agora entender o que é o XPath e observar algumas exemplificações de seu uso.

4.2. XPath

A expressão XPath forma uma essencial parte do XSLT, mas atualmente é definida em separado do XSLT por uma outra recomendação W3C. Devido a isto, o XPath é possível de ser usado independentemente do XSLT.

A sintaxe do XPath é desenvolvida para retornar nós de um documento XML. Segundo Kay (2004), ela acessa os nós específicos do documento preservando a hierarquia e a estrutura. O XSLT entraria no momento em que se necessitar manipular os resultados, por exemplo, rearranjando nós selecionados ou construindo novos nós.

> XPath é uma sintaxe utilizada para descrever partes de um documento XML. Com XPath, você pode referenciar para o primeiro elemento <*para*>, o atributo <*quantity*> de um elemento <*part-number*>, todos elementos <*first-name*> que contenham o texto “Joe” e várias outras variações. Uma folha de estilos XSLT utiliza expressões XPath nos atributos *match* e *select* de vários elementos para indicar como um documento deve ser transformado. (TIDWELL, 2000, p. 42. Tradução e grifo nosso).

Portanto, entende-se que “uma expressão XPath pode ser utilizada para cálculos numéricos ou manipulação de strings ou para testar condições booleanas, mas seu uso mais característico (e esse que lhe deu o nome) é identificar partes de um documento de entrada a serem processadas”. (KAY, 2004, p. 22. Tradução nossa).

Vamos imaginar um exemplo simples (e didático) de uma tabela em um banco de dados cujo nome seja ALUNO e cujos campos sejam NOME, IDADE, CURSO, EMAIL e TELEFONE (campos possuem nomes auto-explicativos) e, esta tabela teria

um *flag*[9] nomeado de FORMADO que indicaria se o aluno já é formado ou não. Se formos mapearmos esta tabela utilizando documentos XML provavelmente ficaria como o ilustrado na figura 14.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<universidade>
   <aluno formado="S">
      <nome>Igor</nome>
      <idade>25</idade>
      <curso>Ciência da Computação</curso>
      <email>igor@fumec.br</email>
      <telefone>11111111</telefone>
   </aluno>
   <aluno formado="S">
      <nome>Aline</nome>
      <idade>22</idade>
      <curso>Administração</curso>
      <email>aline@fumec.br</email>
      <telefone>22222222</telefone>
   </aluno>
   <aluno formado="N">
      <nome>Carlos</nome>
      <idade>23</idade>
      <curso>Ciência da Computação</curso>
      <email>carlos@fumec.br</email>
      <telefone>44444444</telefone>
   </aluno>
   <aluno formado="S">
      <nome>Ricardo</nome>
      <idade>23</idade>
      <curso>Ciência da Computação</curso>
      <email>ricardo@fumec.br</email>
      <telefone>33333333</telefone>
   </aluno>
   <aluno formado="N">
      <nome>Matheus</nome>
      <idade>22</idade>
      <curso>Administração</curso>
      <email>matheus@fumec.br</email>
      <telefone>55555555</telefone>
   </aluno>
</universidade>
```

Figura 14: XML referente à tabela exemplo ALUNO
Fonte: Arquivo pessoal.

Caso houvesse uma demanda para se obter o nome, idade e curso dos alunos que já formaram em ordem ascendente pelo nome, pelo SGDB seria facilmente feito deste modo: *select NOME, IDADE, CURSO from ALUNO where*

[9] “Em ciência da computação, flag referência um ou mais bits que são utilizados para armazenar um valor binário ou código que tenha um significado atribuído.” (FLAG..., 2006. Tradução nossa).

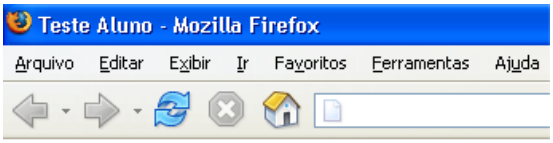
FORMADO = 'S' order by nome. Se estas informações estivessem no XML como exibido na figura 13, poderíamos utilizar uma folha de estilo XSLT como a da figura 15 que utiliza expressões XPath.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
   <xsl:output method="html" indent="yes"/>

   <xsl:template match="/">
      <html>
         <head>
            <title>Teste Aluno</title>
         </head>
         <body>
               <table border="1">
                  <tr>
                     <th>NOME</th>
                     <th>IDADE</th>
                     <th>CURSO</th>
                  </tr>
                  <xsl:for-each select="universidade/aluno[@formado='S']">
                     <xsl:sort select="nome"/>
                     <tr>
                        <td><xsl:value-of select="nome"/></td>
                        <td><xsl:value-of select="idade"/></td>
                        <td><xsl:value-of select="curso"/></td>
                     </tr>
                  </xsl:for-each>
               </table>
         </body>
      </html>
   </xsl:template>
</xsl:stylesheet>
```

Figura 15: Folha de estilo XSLT com expressões XPath
Fonte: Arquivo pessoal.

Ao vincularmos a folha de estilos descrita acima ao documento XML (figura 14) é obtido um resultado como pode ser observado na figura 16. Ao analisar mais detalhadamente percebe-se que não apenas convertemos o documento XML, mas sim, também o publicamos em formato HTML com as informações desejadas. Isto demonstra que tanto a conversão de dados, a apresentação de dados, os filtros de dados, os cálculos e as conversões podem ser interligados e funcionarem em conjunto. Isto comprova o enorme poder do XSLT vinculado a expressões XPath.



NOME	IDADE	CURSO
Aline	22	Administração
Igor	25	Ciência da Computação
Ricardo	23	Ciência da Computação

Figura 16: Resultado XSLT com XPath
Fonte: Arquivo Pessoal.

Enfim, o XSLT utiliza expressões XPath com a intenção de tornar um documento XML mais dinâmico podendo, praticamente, efetuar a maioria das funções permitidas em SGBDs.

CONCLUSÃO

A dificuldade em se trabalhar com integração de aplicações distintas diminuiu consideravelmente com a utilização do XML. O XML apresentou-se como um padrão a substituir arquivos textos, isto indica que o que era antes integrado através de arquivos textos poderia e deveria ser substituído por documentos XML.

Linguagens de programação como C, C++, VB, Java adotaram o XML como o formato universal, pois independentemente de qual plataforma ou de qual linguagem o sistema esteja desenvolvido, ele saberá extrair informações de um documento XML. Tanto que a idéia de Web Services que são serviços disponibilizados na Internet para serem consumidos por qualquer sistema, utilizam dados estruturados em documentos XML.

O problema do XML é que ele possui dados estruturados e esta estruturação freqüentemente varia de um sistema para outro. A partir deste problema, o estudo foi feito sobre uma tecnologia chamada XSLT que realiza a transformação de documentos XML, melhorando sua portabilidade e interoperabilidade.

Um dos maiores problemas que ocorria antes da criação do XSLT era o de duas aplicações de mesma natureza que necessitavam se integrar. Mesmo os dados da primeira aplicação estando em documentos XML e conter as informações necessárias para a integração, o documento não era validado devido ao fato da outra aplicação esperar um documento XML em uma outra estrutura.

Neste aspecto, o estudo prova que a partir de qualquer documento XML, pode-se gerar um outro documento XML com as mesmas informações em estruturas

distintas resolvendo assim qualquer problema de integração por documentos XML, bastando apenas à inserção de uma folha de estilos XSLT para realizar a conversão.

Também como foco deste estudo pode-se encarar o XSLT como a solução de um outro problema: diversas informações contidas em documentos XML poderiam ser apresentadas em um formato mais elaborado como, por exemplo, uma lista de aprovados no vestibular de uma universidade. O sistema acadêmico certamente possui estas informações em documento XML para a interoperabilidade entre os sistemas e, a partir deste documento XML aplicado a uma folha de estilos XSLT pode-se gerar uma página HTML dinâmica ou um documento PDF ou qualquer outro formato texto apresentando somente as informações relevantes.

Conclui-se que qualquer documento XML pode, utilizando folhas de estilos XSLT, ter suas informações facilmente extraídas proporcionando tanto a conversão dos dados em outros documentos XML quanto à apresentação estética destes dados em formatos para o usuário. Conclui-se também que qualquer transformação com a utilização de folhas de estilos XSLT pode-se aplicar um filtro de dados, isto é, ao realizar a transformação do documento XML observou que uma das informações não é necessária, portanto não será transformada.

Este estudo cobriu o foco principal de demonstrar a utilidade e a importância de uma linguagem de transformação de documento XML, tal como demonstrar diversas situações em conversões e apresentações de dados que são os principais cenários da aplicação de folhas de estilo XSLT. Estes assuntos possuem a intenção de abrir perspectivas para serem realizados outros estudos, sejam no quesito técnico que não foi aprofundado ou em análises de problemas reais que foram solucionados com a utilização do XSLT.

REFERÊNCIA BIBLIOGRÁFICA

ABOUT the World Wide Web Consortium, W3C. Disponível em: <http://www.w3.org/Consortium/>. Acesso em: 16 abr. 2006.

BOOLEAN, Wikipédia. Disponível em: <http://en.wikipedia.org/wiki/Boolean>. Acesso em: 11 jun. 2006.

BRAGANHOLO, Vanessa de Paula; HEUSER, Carlos A. XML Schema, RDF(S) e UML: uma comparação. Disponível em: <http://www.dcc.ufrj.br/~braganholo/artigos/ideas2001.pdf >. Acesso em: 04 jun. 2006.

DUCHARME, Bob. **XSLT**. Rio de Janeiro: Ciência Moderna, 2002.

EXTENSIBLE Markup Language (XML), W3C. Disponível em: <http://www.w3.org/XML/>. Acesso em: 21 fev. 2006.

FLAG: computing, Wikipédia. Disponível em: <http://en.wikipedia.org/wiki/Flag_(computing)>. Acesso em: 11 jun. 2006.

HTML, Wikipédia. Disponível em: <http://pt.wikipedia.org/wiki/HTML/>. Acesso em: 16 abr. 2006.

HYPERTEXT Markup Language (HTML) Home Page, W3C. Disponível em: <http://www.w3.org/MarkUp/>. Acesso em: 16 abr. 2006.

INTRODUCTION TO DTD, W3C. Disponível em: <http://www.w3schools.com/dtd/dtd_intro.asp>. Acesso em: 27 maio 2006.

INTRODUCTION TO XML SCHEMA, W3C. Disponível em: <http://www.w3schools.com/schema/schema_intro.asp>. Acesso em: 21 maio 2006.

KAY, Michael. **XSLT programmer's reference**. Birmingham: Wrox Press, 2000.

KAY, Michael. **XSLT 2.0 programmer's reference**. 3 ed. Indianapolis: Wiley Publishing, 2004.

MACORATTI, José Carlos. Conceitos básicos de modelagem de dados. Disponível em: <http://www.macoratti.net/cbmd1.htm>. Acesso em: 27 maio 2006.

MYER, Tom. A really, really, really good introduction to XML. Disponível em: <http://www.sitepoint.com/article/really-good-introduction-xml>. Acesso em: 21 maio 2006.

NAMESPACE: Computer science. Wikipédia. Disponível em: <http://en.wikipedia.org/wiki/Namespace_(computer_science)#XML>. Acesso em: 27 maio 2006.

REFSNES, Jan Egil. XML DTD: an introduction to XML document type definitions. Disponível em: <http://www.xmlfiles.com/dtd/default.asp>. Acesso em: 21 maio 2006.

SERIALIZATION, Wikipédia. Disponível em: <http://en.wikipedia.org/wiki/ Serialization>. Acesso em: 10 jun. 2006.

SISTEMA DE GERENCIAMENTO de banco de dados, Wikipédia. Disponível em: <http://pt.wikipedia.org/wiki/SGBD>. Acesso em: 11 jun. 2006.

TIDWELL, Doug. **XSLT**. Sebastopol: O'Reilly Media, 2001.

W3C, Wikipédia. Disponível em: <http://pt.wikipedia.org/wiki/W3C/>. Acesso em: 16 abr. 2006.

XML, Wikipédia. Disponível em: <http://pt.wikipedia.org/wiki/XML/>. Acesso em: 16 abr. 2006.

XML Path Language (XPath) 2.0, W3C, 03 nov. 2005. Disponível em: <http://www.w3.org/TR/xpath20/>. Acesso em: 05 mar. 2006.

XML Schema Part 0: Primer Second Edition, W3C, 28 out. 2004. Disponível em: <http://www.w3.org/TR/xmlschema-0/>. Acesso em: 21 fev. 2006.

XML TO XML WITH XSLT. Disponível em: <http://www.topxml.com/xslt Stylesheets/xslt_XML_to_XML.asp>. Acesso em: 04 jun. 2006.

XPATH Tutorial, W3Schools. Disponível em: <http://www.macoratti.net/ vb_xpath.htm>. Acesso em: 14 mar. 2006.

XSL Transformations (XSLT) Version 2.0, W3C, 03 nov. 2005. Disponível em: <http://www.w3.org/TR/xslt20/>. Acesso em: 21 fev. 2006.