Pós-Graduação em Ciência da Computação

"Projeto e Implementação de Links em Documentos XML"

Por

VALÉRIA ARGOLO ROSA

Dissertação de Mestrado

Universidade Federal de Pernambuco

posgraduacao@cin.ufpe.br

www.cin.ufpe.br/~posgraduacao

RECIFE, FEVEREIRO/2003

UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

VALÉRIA ARGOLO ROSA

"Projeto e Implementação de Links em Documentos XML"

ESTE TRABALHO FOI APRESENTADO À PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA DA COMPUTAÇÃO.

ORIENTADOR: Prof Dr. Roberto Souto Maior de Barros

RECIFE, FEVEREIRO/2003

“A todos aqueles que acreditam em

novos desafios e sabem que é

possível realizá-los.”

Agradecimentos

A Deus, por todas as bênçãos e graças recebidas.

Aos meus pais, Nilton e Glorinha e meus irmãos, Paty, Beto e Edú, por tudo que representam para mim.

Às minhas tias, Stela e Helena, pelo carinho, incentivo e orações.

Ao meu namorado, Rodrigo, pelo apoio e compreensão nas minhas ausências.

Ao professor e orientador Roberto, pelo incentivo e tranqüilidade com que conduziu todo o período de orientação.

Às queridas amigas, Mai e Cau, pelo carinho, amizade, incentivo, orações e por todos os momentos que passamos juntas.

A todos os colegas do Mestrado, que me ajudaram direta ou indiretamente, em especial Silvinha, Roque, Marcão, Zirinha, Jaiminho e Chico.

Aos amigos Mara, Chico, Lela, Marilia, Gil, D. Zita, Hélio e Bruno obrigada pelo carinho, apoio, incentivo, ensinamentos e acolhida em Recife. Jamais esquecerei vocês.Valeu!

À colega do DEBI-Campus de Itapetinga-BA, Rita de Cássia, pelo incentivo com que me fez dar o primeiro passo para me inscrever nesse mestrado.

Aos Amigos e Colegas do DQE-Campus Jequié, em especial, aos professores Joélia, Edson, Telesson, Baraquizio, Cândido, Vanda e Regina, obrigada pelas palavras de carinho e apoio.

Ao professor Benedito Acioly, por ter acreditado em nós.

Aos professores da UFSCar-SP, em especial Prof. Sérgio Donizetti Zorzo, pelos ensinamentos e a confiança depositada em mim.

Finalmente, agradeço a Ricardo Lugo, por sua colaboração na realização desse trabalho.

Resumo

Uma das características principais da linguagem HTML, e o que a tornou bastante popular é o fato de ser uma linguagem de hypertexto, ou seja, permite associar, através de links, as informações de uma página a outra página da Web. Porém, apesar do sucesso, o sistema de links da HTML é muito restrito a características bastante simples, o que motivou a criação de uma tecnologia de links mais robusta e inteligente. Esta tecnologia, chamada de XLink, é a linguagem de links associada à linguagem XML, que permite criar links multidirecionais, controlar como e quando os links são ativados, entre outras coisas.

Além do XLink, foi desenvolvida também uma linguagem de ponteiros, o XPointer, que provê uma maneira para os localizadores em links XML apontarem para locais específicos dentro dos recursos.

Uma vez que o XLink e o XPointer são tecnologias novas, os principais browsers ainda não fornecem um suporte completo a essas linguagens, apesar do XLink já ser uma recomendação da W3C (Órgão que define o padrão para a WWW).

Este trabalho tem como objetivo principal implementar uma solução, baseada na especificação de XLink (desenvolvida pela W3C), que auxilie na compreensão do funcionamento dos links em documentos XML, utilizando o browser padrão Internet Explorer.

A solução desenvolvida, chamada de IXLink, é um interpretador implementado em JavaScript, que pode ser usado para qualquer tipo de documento XML que contenha XLink.

Esse interpretador é capaz de processar a sintaxe do XLink com alguns dos seus respectivos atributos e valores, e parte do XPointer. Simulando dessa forma o funcionamento dos links XML e possibilitando um maior entendimento das especificações. O seu código pode ser facilmente adaptado para a utilização em outras páginas Web, dando assim um suporte aos webdesigners na construção de sites dinâmicos e interativos.

Palavras-chave: XML, XLink, XPointer, JavaScript, Interpretador

Abstract

One of the key features of HTML, which made it quite popular, is the fact that it is a hypertext language. In other words, it is possible to link the information of a web page to another web page. However, in spite of its success, HTML links are very restricted to simple features. This fact was the motivation to the development of a new technology to support more robust and intelligent links. This technology, called XLink, is the language of links of XML. It allows for the construction of multidirectional links, to control 'how' and 'when' links are activated, among others things.

In addition to XLink, it was developed a language of pointers, the XPointer. It provides the means for the locators in XML links to point to specific places inside the resources.

Because XLink and XPointer are new technologies, the mains browsers still do not provide complete support to them, even though XLink is a W3C recommendation.

The main objective of this work is to implement a solution based on the XLink specification developed by W3C that aims to help understanding how links in XML documents work.

The solution developed, called IXLink, is an interpreter implemented in JavaScript to run in the Internet Explorer browser, that can be used for any type of XML documents that contain XLinks. This interpreter is capable of processing the syntax of XLink limited to some of its attributes and values, and part of XPointer, simulating in that way the support for links in XML and making it possible a better understanding of the specifications. Its code can be easily adapted to be used in other web pages, providing support to webdesigners in the implementation of dynamic and interactive sites.

Keywords: XML, XLink, XPointer, JavaScript, Interpreter.

Sumário

Índice de Figuras

Índice de Tabelas

Capítulo 1 - Introdução

Neste capítulo apresentamos uma visão geral da dissertação, ressaltando motivações e objetivos, como também uma descrição da organização do trabalho.

1.1. Introdução

Desde que a *World Wide Web-WWW* foi projetada por Tim Berners-Lee em 1991 e o Mosaic-née Netscape foi desenvolvido em 1993 para visualizar a Web em qualquer computador pessoal, a Internet passou de interessante a essencial, bem como, de auxiliar a completamente central [Pitts-Moultis 2000].

Atualmente os sites da Web são uma parte necessária de uma infra-estrutura empresarial e também costumam ser parte da vida privada das pessoas. A quantidade de informações disponíveis na Internet tornou-se praticamente incontável. Ninguém sabe exatamente quantas páginas existem na Web, embora provavelmente sejam cerca de dois bilhões [Pitts-Moultis 2000].

O extraordinário crescimento da Web deve-se, principalmente, à facilidade com que se pode partilhar documentos a nível mundial. Estes documentos são, em geral, construídos em HTML. Esta é uma linguagem simples de aprender e que permite explorar as potencialidades do hipertexto, sendo esta uma das suas características principais e o que a tornou bastante popular.

Entretanto, rapidamente se percebeu que a rede existente poderia ser utilizada para documentos mais elaborados e complexos. E a HTML, com muitas limitações, não possuía as características consideradas fundamentais para o desenvolvimento destes novos documentos como, por exemplo, uma sintaxe rígida e permitir a validação de dados ou extensões à linguagem. Além de HTML ter um sistema de links muito restrito e simples.

Face a essas limitações, verificou-se a necessidade de criar uma nova linguagem que suportasse as exigências dos novos tipos de documentos e aplicações na Web. Então em 1996 o W3C - *World Wide Web Consortium*, órgão que define os padrões para a WWW, começou a trabalhar num novo padrão, o qual chamou de XML (*eXtensible Markup Language*).

XML define regras para escrever esses documentos de modo que se tornem legíveis para o computador, evitando ambigüidades e dependência de plataformas. A meta é obter documentos que sirvam melhor ao gerenciamento de informações [Bender 2001].

Associada à linguagem XML, criou-se uma tecnologia de links mais robusta e inteligente, chamada de XLink (*XML Linking Language*) [DeRose 2001], que permite criar links multidirecionais, controlar como e quando os links são ativados, entre outras. Desde junho de 2001, o XLink é uma recomendação da W3C.

Em conjunto com o XLink, foi desenvolvida uma linguagem de ponteiros, chamada XPointer, que provê uma maneira para os localizadores em links XML apontarem para locais específicos dentro dos recursos [Esteves 2001]. A partir de 11 de setembro de 2001, XPointer passou a ser uma candidata à recomendação[1].

1.2. Motivações

Várias características e funcionalidades foram definidas pela W3C nas especificações de XLink e XPointer, características essas que fazem dos links XML criativos e sofisticados, adicionando funcionalidade muito mais poderosa que os links da HTML [St. Laurent 2002].

Porém, há mais ou menos um ano, os principais browsers (Internet Explorer e Netscape) não ofereciam suporte a essas tecnologias. Atualmente, apenas o Netscape oferece algum suporte e, mesmo assim, incompleto.

A implementação de um interpretador que objetivasse a simulação do funcionamento dos links em documentos XML tornou-se uma alternativa para uma compreensão prática do que foi definido nas especificações, principalmente na especificação de XLink.

1.3. Objetivos

Os objetivos desse trabalho são:

[1] Candidata a Recomendação – linguagem oficial do grupo que significa que grupos de trabalho e empresas interessadas podem iniciar a construção de programas e dar retorno técnico.

- Descrever as características das tecnologias XLink e XPointer, de forma mais clara e concisa que as definidas nas especificações elaboradas pela W3C.
- Implementar uma solução, baseada nas especificações de XLink e XPointer, que auxilie na compreensão do funcionamento dos links em documentos XML, utilizando o browser Internet Explorer.
- Apresentar, de forma prática, o funcionamento dos links em documentos XML.

1.4. Organização da Dissertação

O trabalho é composto por seis capítulos, incluindo este introdutório. O Capítulo 2 descreve um estudo sobre a linguagem XML, abordando principais conceitos, benefícios, comparações com as linguagens HTML e SGML e estrutura dos dados.

No Capítulo 3 é apresentado um estudo sobre XLink, descrevendo conceitos, princípios do projeto, origem, principais características e uso do XLink.

No Capítulo 4 é apresentada uma visão geral sobre XPointer, descrevendo conceitos, princípios do projeto, origem e uso do XPointer.

O Capitulo 5 descreve o projeto e implementação do IXLink, interpretador que será capaz de processar a sintaxe do XLink com alguns dos seus respectivos atributos e valores e parte do XPointer. E apresenta o funcionamento dos links em documentos XML, usando o que foi implementado.

Finalmente, no Capítulo 6, serão apresentadas as conclusões da dissertação, bem como as principais contribuições e sugestões para trabalhos futuros.

Capítulo 2 - A Linguagem XML

Este capítulo é uma visão geral sobre a linguagem XML, abordando os principais conceitos, vantagens, comparações com as linguagens HTML e SGML, estrutura de dados e contextualização histórica.

2.1. Introdução

Em meados de 1996, um grupo de trabalho do *Word Wide Web Consortium* (W3C), organização encarregada em desenvolver e manter a maior parte dos padrões da Web, deu início ao desenvolvimento de uma linguagem de marcação para solucionar as limitações da HTML e que tivesse o poder e a generalidade da SGML[2] (*Standard Generalized Markup Language*) e que, ao mesmo tempo, fosse fácil de ser implementada na Web [Light 1999]

Esse trabalho deu origem a XML (*Extensible Markup Language*), uma linguagem de marcação extensível que possuía os seguintes objetivos de projeto [Bray 2000]:

1. Deve ser utilizada de forma direta e objetiva na Internet.
2. XML deve suportar uma ampla gama de aplicativos.
3. XML deve ser compatível com a SGML.
4. XML deve ser fácil o bastante para escrever programas que processem documentos XML.
5. O número de recursos adicionais em XML deve ser mantido em um nível mínimo, idealmente zero.
6. Os documentos XML precisam ser legíveis e relativamente claros.
7. O projeto XML deve ser preparado rapidamente.
8. O design XML deve ser formal e conciso.
9. Os documentos XML precisam ser fáceis de serem criados.
10. A concisão na marcação XML é de pouca importância.

[2] SGML – Linguagem Padrão de Marcação Generalizada que fornece um esquema de marcação simples, independente de plataforma e extremamente flexível.

A primeira espeficicação de XML, a XML 1.0, foi publicada em 10 de fevereiro de 1998. A especificação mais recente data de 06 de outubro de 2000. Mas, atualmente, existe uma nova versão dessa linguagem, XML 1.1, candidata à recomendação pela W3C.

2.2. O que é XML?

XML é uma linguagem de marcação que permite que outras linguagens de marcação sejam definidas a partir dela. XML permite que um desenvolvedor defina seu próprio conjunto de *tags* (marcas) de forma que estas possam dar sentido aos dados que devem ser armazenados, formando assim documentos autodescritivos [Young 2000].

XML é um subconjunto de SGML que incorpora aspectos como marcação simples, independência de plataforma e flexibilidade, mas sem a complexidade presente na SGML. Os documentos XML são relativamente fáceis de serem criados e usados na Web [Castro 2001].

A XML surgiu para solucionar alguns problemas encontrados na HTML. Esta é uma aplicação da SGML que possui um conjunto de tags pré-definidas. Essas tags servem basicamente para apresentar o conteúdo de documentos em páginas Web e não dão nenhum sentido a eles. XML, entretanto, possibilita [Furgeri 2001]:

- A criação de tags conforme as necessidades do usuário, possibilitando o surgimento de linguagens baseadas em XML.
- A visualização do mesmo documento de diferentes formas, por meio da utilização de folhas de estilos, ordenando e filtrando informações segundo certos critérios;
- A estrutura criada pelo documento XML permite que ferramentas baseadas em bancos de dados possam consultar e processar seu conteúdo;
- Ferramentas desenvolvidas para a manipulação de XML facilitam a criação de documentos, permitindo que o desenvolvedor se dedique apenas ao conteúdo das informações, já que a estrutura é controlada pela ferramenta.
- A XML traz benefícios tanto para aquele que produz a informação como para aquele que a recebe. A maneira como a estrutura de um documento XML é formada possibilita que os processos sejam automatizados por meio de software que reconhecem as tags do documento XML, integrando o site da Internet com o software

interno de uma empresa, reduzindo custos e tornando os processos mais eficientes, tanto para o produtor dos documentos quanto para quem vai ler seu conteúdo;

- A XML é um padrão permanente, já que o conteúdo de seu documento XML pode ser lido e atualizado pela ferramenta que estiver sendo utilizada;
- Os browsers, apoiados por linguagens de programação, podem fazer grande parte do processamento das informações; o conteúdo do documento pode ser manipulado e reorganizado; cálculos podem ser realizados para gerar novos conteúdos instantaneamente, o que proporciona a geração de novos documentos;
- Os recursos fornecidos pela XML podem ser usados para criar uma rede de conhecimento, interligando documentos com informações complementares, mesmo que eles estejam em lugares diferentes na Web, ou ainda, que pertençam a diferentes empresas.

2.2.1. As diferenças entre XML e HTML

XML é uma linguagem de marcação mais avançada do que a HTML e não é, de forma alguma, um substituto para a mesma. Embora a HTML seja uma aplicação da SGML e XML um subconjunto, HTML e XML diferem consideravelmente. E as diferenças são mais notáveis nos problemas que são encontrados na HTML e que XML consegue solucionar. Com XML, tem-se:

- Melhor controle em relação ao layout;
- Menos esforço no servidor Web devido à capacidade de acessar informações do lado do cliente;
- O uso de vários tipos de links;
- A capacidade de publicar qualquer tipo de informação tanto na Internet quanto em intranets;
- Um número menor de problemas para exibir páginas longas.

Algumas diferenças entre HTML e XML são apresentadas na Tabela 2-1[Pitts-Moultis 2000]:

Tabela 2-1 Tabela de Diferenças entre HTML e XML

HTML	XML
Usado basicamente para fins de exibição; possui pouco conhecimento sobre as informações nela contidas.	Usado para estruturar dados, ao invés de para exibi-los.
Uma linguagem fechada que não dá ao desenvolvedor flexibilidade de criar rótulos para fins específicos. Cada conjunto novo de rótulos é criado por meio do *World Wide Web Consortium.*	Totalmente aberta para elementos novos e os navegadores serão capazes de incorporá-los em páginas.
Relativamente fácil de aprender.	Não tão fácil de ser aprendido, requer um planejamento maior.
As marcas não diferenciam o uso de letras maiúsculas de minúsculas.	Os elementos, atributos e tudo o mais diferenciam o uso de letras maiúsculas de minúsculas.
Os espaços em branco em um documento são ignorados.	Os espaços em branco não são ignorados.

2.2.2. As diferenças entre XML e SGML

Embora XML seja derivada da SGML, existem algumas diferenças entre elas, conforme mostra a Tabela 2-2.

Tabela 2-2 Tabela de Diferenças entre XML e SGML

SGML	XML
Linguagem complexa. A especificação tem mais de 500 páginas.	Linguagem muito mais compacta. A especificação de XML tem apenas 50 páginas
Necessidade de validar documentos. Os documentos devem primeiramente ser validados e usados com uma DTD e também devem usar folhas de estilos para exibir as informações dentro do documento	Não há necessidade de uma DTD e a validação nem sempre é necessária em documentos XML. Basicamente, tudo que um documento XML precisa é de uma folha de estilos. Tornando assim documentos mais portáveis e mais acessíveis na Web do que os SGML.
SGML é mais complexa e usada em diversas áreas de publicação de informações.	XML é mais simples e criado especificamente para uso na Internet
Documentos mais demorados de serem criados.	Documentos mais rápidos de serem criados.
Devido à sua complexidade, torna desanimadora para os programadores a tarefa de incorporá-la em seus aplicativos.	É muito mais fácil escrever aplicativos que interpretam documentos XML

2.3. Padrões da Estrutura XML

XML é acompanhada por uma série de padrões definidos pelo W3C. Entre eles:

- **XML Namespaces**

Descreve a sintaxe de namespace, ou espaço de nomes, e que serve para criar prefixos para os nomes de tags, evitando confusões que possam surgir com nomes iguais para tags que definem dados diferentes.

- **Folhas de Estilo**

XML tem compatibilidade com duas linguagens que trabalham com folhas de estilo: XSL (XML Stylesheet Language) e CSS (Cascading Style Sheet). Elas especificam como os documentos XML devem ser apresentados na tela, no papel ou em um editor. A XSL é mais poderosa mas a CSS possui mais implementações.

- **DOM e SAX**

DOM (Document Object Model) e SAX (Simple API for XML) são APIs de acesso aos documentos XML. Elas permitem que aplicativos leiam documentos XML sem se preocuparem com a sintaxe (semelhante aos tradutores). DOM é uma API orientada a objetos e SAX é uma API baseada em eventos [Pitts-Moultis 2000].

- **XLink e XPointer**

XML Linking Language (XLink) é uma linguagem de construção de links, similar aos links HTML, sendo que é mais poderosa, porque os links podem ser multidirecionais e podem existir a nível de objetos, e não somente a nível de página.

XML Pointer Language (XPointer) é uma linguagem de ponteiros criada para fornecer uma maneira para os localizadores em links XML apontar para locais específicos dentro dos recursos.

Os objetos de estudo deste trabalho são os padrões XLink e XPointer. Esses padrões recebem um enfoque maior nos capítulos seguintes.

2.4. Documentos XML

XML é uma linguagem usada para descrever e manipular documentos estruturados. Todo documento escrito em XML deve ser criado de forma concisa e clara [Furgeri 2001]. Um documento XML é um arquivo texto que contém elementos, atributos, dados de caracter, comentários, entidades, instruções de processamento, seção CDATA, entre outros.

Um documento para ser considerado um documento XML deve ser bem formado. Para isto ele deve atender as seguintes regras definidas pelo W3C [Bray 2000]:

- Ter a declaração XML na primeira linha do documento,
- Incluir um ou mais elementos;
- Ter exatamente um elemento raiz que deve conter todos os outros elementos;
- O nome da marca de fim (`</CLIENTE>`) deve coincidir com a marca de início (`<CLIENTE>`);
- As *tags* devem ser aninhadas adequadamente, isto é, não pode haver sobreposição;
- Cada atributo de um elemento deve ter um nome exclusivo;
- Cada uma das entidades analisadas, referidas dentro do documento, deve ser bem formada.

O código apresentado na Figura 2-1 é um documento mal formado, pois o aninhamento dos elementos `nome` e `snome` não estão corretos:

```
<? xml version =”1.0” ?>
<nome>
  <pnome> João</pnome>
 <snome> da Silva
</nome></snome>
```

Figura 2-1 Documento XML mal formado

Para este ser considerado um documento XML é preciso alterar a ordem das tags de fechamento de `nome` e `snome`, como mostra a Figura 2-2.

```
<? xml version =”1.0” ?>
<nome>
  <pnome> João</pnome>
  <snome> da Silva</snome>
</nome>
```

Figura 2-2 Documento XML Bem Formado

Um documento que obedece estas regras pode ser representado por uma estrutura de árvore hierárquica. Um elemento do documento torna-se nó na árvore e o seu conteúdo torna-se nó filho do elemento. Por exemplo, o elemento raiz do documento torna-se o nó raiz da árvore.

Se esse elemento possui outros elementos, esses elementos serão filhos do nó raiz e assim sucessivamente, obedecendo ao aninhamento do documento. A Figura 2-3 mostra a árvore do documento da Figura 2-2.

Um elemento que esteja embutido em outro elemento é chamado de *filho*. O elemento no qual ele está embutido é seu *pai*. Na Figura 2-3, o elemento ***nome*** possui dois filhos: o elemento ***pnome*** e o elemento **snome**. Cada um desses elementos tem um filho que é o seu conteúdo texto.

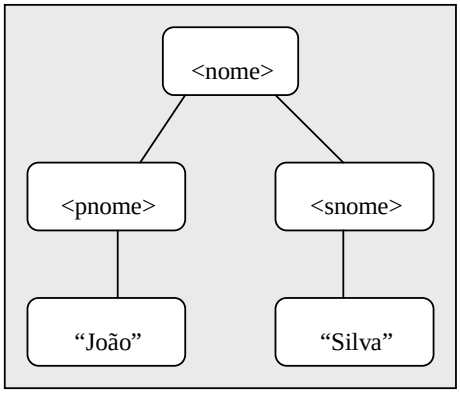


Figura 2-3 Árvore do documento nome.xml

Além de bem formado, um documento XML também pode ser válido. Para que um documento seja válido é necessário que uma DTD (Declaração do Tipo de Documento) ou um esquema XML esteja associado a ele e que ele esteja de acordo com as especificações descritas na DTD ou no esquema. O exemplo da Figura 2-4 mostra um documento XML válido de acordo com a DTD em destaque.

```
<? xml version =”1.0” ?>
▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫
▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫
▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫
▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫
▫▫▫
<? xml version =”1.0” ?>
<nome>
    <pnome>João</pnome>
    <snome>Silva</snome>
</nome>
```

Figura 2-4 Documento XML válido

2.5. A sintaxe de XML

Um documento XML é composto por blocos de construção que podem conter: elementos, atributos, cadeias de caracteres, instruções de processamento, comentários, seções CDATA, referências, entre outros [Marchal 2000].

- **Elementos**

Um elemento XML é a unidade mais básica do documento. Ele é composto de uma tag de abertura, o conteúdo e uma tag de fechamento [Heitlinger 2001].

A tag de abertura é formada pelo nome da tag escrito entre os sinais de menor que (<) e maior que (>).Um elemento geralmente é terminado com uma tag de fechamento, formada pelo mesmo nome precedido por uma barra inclinada, e delimitado pelos sinais de menor que e maior que (</nome>) [Castro 2001].

O conteúdo de um elemento pode ser outros elementos e/ou texto. A Figura 2-5 mostra o exemplo do elemento **nome** que tem como conteúdo uma cadeia de caracter.

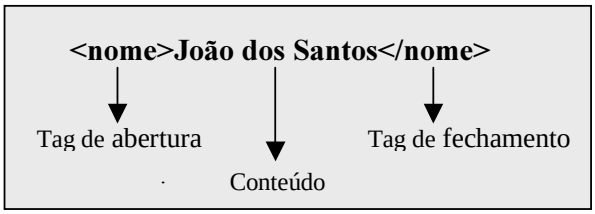


Figura 2-5 – Exemplo de elemento

O nome da tag é criado pelo próprio desenvolvedor e deve descrever a finalidade do elemento e seu conteúdo em particular. O nome da tag deve começar com uma letra ou um sublinhado, seguido por letras, dígitos, sublinhados, hífens ou pontos, mas não pode haver espaços em branco no nome.

- **Elemento raiz (root)**

O primeiro elemento de um documento XML é chamado de elemento raiz (root). Ele engloba todos os outros elementos do documento e sua utilização é obrigatória. Esse elemento pode ser comparado à raiz de um disco rígido utilizado nos computadores, a partir do qual todas as informações são armazenadas em pastas [Castro 2001].

No exemplo apresentado na Figura 2-6 o elemento `<catálogo>` é o elemento raiz do documento XML. Ele é composto de dois elementos `<item>`.

```
<catálogo>
      <item>
              <nome> José dos Santos</nome>
              <e-mail href=”mailto:jsantos@sjnet.com.br”/>
      </item>
      <item>
            <nome><pnome>João</pnome><snome>da Silva</snome></nome>
            <e-mail href= mailto:jsilva@sjnet.com.br / >
      </item>
</catálogo>
```

Figura 2-6 Elemento raiz <catálogo >

- **Elemento vazio**

Os elementos que não possuem conteúdo são conhecidos como elementos vazios. Normalmente, eles são incluídos no documento pelo valor de seus atributos.

Um elemento vazio é formado pelo nome do elemento seguido de caracter “/” entre os sinais de menor que (<) e maior que (>). Como mostra o exemplo abaixo:

```
<e-mail href=”mailto:jsantos@sjnet.com.br”/>
```

Para XML, este elemento é equivalente a:

```
<e-mail href= “mailto:jsantos@sjnet.com.br”></e-mail>
```

- **Atributos**

Os atributos podem ser utilizados para fornecer propriedades especiais para as *tags* presentes no documento XML. Eles podem ser utilizados de diversas maneiras e são muito úteis para os documentos XML, pois podem anexar informações aos elementos.

Cada atributo possui um nome e um valor. Os nomes seguem as mesmas regras dos nomes de elementos. O nome é separado do valor pelo caracter de igualdade (=). O valor do atributo deve aparecer entre aspas ou apóstrofos. A Figura 2-7 mostra o exemplo de um atributo.

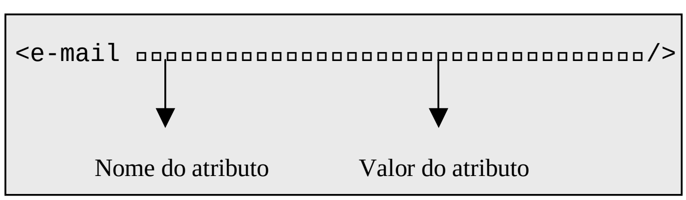


Figura 2-7 – Exemplo de atributo

Os elementos podem ter zero ou mais atributos na tag de início ou na tag de elemento vazio. Mas o mesmo atributo não pode aparecer mais de uma vez no mesmo elemento.

➢ Instruções de Processamento

São elementos especiais adicionados ao documento XML para informar certos detalhes às ferramentas que realizarão sua interpretação. A idéia é permitir que o documento XML transmita alguma informação para o software que irá interpretar seu conteúdo.

Uma instrução de processamento é delimitada pelos caracteres <? e ?>. O código apresentado na Figura 2-8 mostra a instrução de processamento que informa o arquivo xsl que deve ser utilizado para apresentar o documento de catálogo:

```
<?xml version="1.0"?>
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
<catalogo>
   ...
</catalogo>
```

Figura 2-8 Código XML com Instrução de Processamento

➢ Declaração do Tipo de Documento (DTD)

Trata-se de uma instrução adicionada ao documento XML que deverá ser utilizado para verificar se todas as tags usadas estão seguindo certos padrões. Essa declaração do tipo do documento é opcional. Ela apontará para um documento usado para descrever quais tags podem aparecer no documento, o número e a seqüência que podem aparecer, seus atributos e os valores que esses atributos podem possuir. Um exemplo de DTD foi mostrado na Figura 2-4.

➢ Declaração XML

A declaração XML, se existir, é a primeira linha do documento. Ela identifica o documento como um documento XML. A declaração também relaciona a versão da XML usada no documento, o padrão de codificação de caracteres e se o documento possui ou não entidades externas. Para isto ele define os atributos version, standalone e encoding, respectivamente. O código abaixo é um exemplo de uma declaração XML:

```
<?xml version="1.0" standalone =”yes” encoding=”UTF-8”?>
```

➢ Comentários

Assim como nas linguagens de programação, XML também possui o recurso de comentário que facilita a compreensão dos itens presentes no documento. Sempre que necessário, os comentários podem ser utilizados para adicionar notas e observações no documento e não serão levados em consideração pelo interpretador do documento XML.

Para inserir comentários em um documento, delimite-os entre "<!- -" e "- ->":

```
<?xml version="1.0"?>
<!—Inicio do documento catalogo.xml-->
<catalogo>
   ...
</catalogo>
```

- **Entidades**

Uma entidade se refere a um caracter ou a um bloco de texto que será "importado" pelo documento XML toda vez que ele aparecer. Dessa forma, um interpretador do documento pode pesquisar a entidade referenciada e substituir seu conteúdo a partir do ponto em que foi encontrado.

As entidades são inseridas no documento por meio de referências de entidade (o nome da entidade entre um caracter & e um sinal de ponto-e-vírgula). Para a aplicação, a referência de entidade é substituída pelo conteúdo dessa entidade. Se for considerado que foi definida uma entidade 'eua', que possui o valor 'Estados Unidos da América', ao ler a linha

<país>&eua;</país>

o processador XML a substituirá por

<país>Estados Unidos da América</país>.

XML predefine entidades para os caracteres utilizados na marcação (sinais de menor e maior, aspas, entre outros). As entidades são usadas para substituir os caracteres no conteúdo do elemento ou do atributo. As entidades predefinidas são:

- < para criar um sinal de menor que (<);
- > para criar um sinal de maior que (>);
- & para criar um caractere &;
- &após; para criar uma marcação de apóstrofo (').
- " para criar uma marcação de aspas (").

- **Seções CDATA**

Uma seção CDATA é definida entre os caracteres "<[CDATA[" e "]]>". Todo o conteúdo presente entre estes delimitadores não será analisado pelo processador XML. Por exemplo, é

possível utilizar dentro de uma seção CDATA caracteres reservados (<,>, &, etc) sem precisar usar as entidades, pois esses não serão analisados [Punin 2002].

As seções CDATA podem aparecer em qualquer lugar após a tag de abertura do elemento raiz até imediatamente antes da tag de fechamento do elemento raiz. Entretanto, essas seções podem ser aninhadas. A sua sintaxe é: `<![CDATA [...conteúdo...]]>`.

2.6. Considerações Finais

Neste capítulo foram mostrados alguns conceitos relacionados à linguagem XML, que se consolida como um padrão para a troca de informações entre sistemas e aplicações incompatíveis.

XML tem aberto novos horizontes, sendo possível a sua utilização em vários ramos de sistema de computação, por exemplo, E-commerce, E-business, Recurso de busca, desenvolvimento de catálogos, entre outros [Heitlinger 2001].

Hoje, XML é um conjunto de recomendações publicadas pelo W3C, que especifica a semântica e a sintaxe de XML e de tecnologias padrão. Muitas dessas tecnologias já estão com status de recomendação, exemplo XLink, XSLT, DOM Nível 1 e 2, XPath, entre outros o que significa que são extremamente estáveis e têm amplo suporte da indústria. Entretanto, muitas outras tecnologias, exemplo XPointer, Voice XML, XQuery, DOM Nível 3, entre outros estão ainda em andamento, em diversos estágios, desde submissão inicial (*working draft*) até candidatos à recomendação.

Diante do estudo sobre XML apresentado neste Capítulo, percebe-se que essa linguagem poderá ser utilizada em qualquer aplicação na qual seja necessária a troca de dados de forma padronizada. E pode-se perceber também que já é considerada muito importante para o futuro da Web, sendo uma linguagem apropriada para a manipulação de dados.

O estudo realizado neste capítulo servirá como base para o entendimento do objeto principal de estudo que são os links em documentos XML.

Capítulo 3 - XLink

Este capítulo é um estudo sobre XLink, abordando os principais conceitos, princípios do projeto, contextualização histórica, e descrevendo a criação de links simples e estendidos em documentos XML, com base na especificação XLink desenvolvida pela W3C.

3.1. Introdução

XLink é a linguagem de links XML para criar estruturas que descrevem desde ligações unidirecionais simples, como as existentes hoje em HTML, até ligações mais sofisticadas como, por exemplo, ligações que residem em locais distintos dos recursos relacionados [Arciniegas 2000].

O XLink é na verdade derivado de uma série de outras especificações de links. A especificação XLink foi inicialmente chamada *Extensible Linking Language* e, posteriormente, XML-Link, até o W3C ter decidido sobre seu nome final, XLink. Essa linguagem de links fornece os mecanismos para compatibilidade com formatos anteriores da HTML. Os três padrões que contribuíram na formação do XLink foram [Pitts-Moultis 2000]:

- HTML – ajudou a definir vários tipos de elementos SGML que representam links direcionais simples para recursos, independentemente do tipo de protocolo (HTTP, FTP, Gopher, Telnet e assim por diante) usado para conectar a esses recursos;
- HyTyme – O HyTyme, usado dentro da especificação SGML, define as estruturas de links alinhados e desalinhados e algumas características semânticas, inclusive controle de acionamento de links e apresentação de objetos;
- TEI – *Text Encoding Initiative Guidelines (diretrizes da iniciativa de codificação de texto)* – As diretrizes TEI fornecem estruturas para criar links, agregar objetos e conjuntos de links.

A capacidade dos links XML vão além dos links básicos da HTML, acrescentando uma série de novas funcionalidades, incluindo a capacidade de criar links inteligentes sem necessidade de recorrer à outra linguagem de programação [Marchal 2000].

Fazendo um paralelo entre os links XML e HTML, pode-se destacar [Moller 2001]:

- Os links XML permitem que qualquer tipo de elemento indique a existência de um link, enquanto que em HTML existe um elemento predefinido que indica a existência de links:o elemento âncora <A> </A>;

- Os links XML permitem a criação de links unidirecionais, com os links simples, e multidirecionais, com os links estendidos, enquanto HTML permite apenas criação de links unidirecionais, ou seja, navegação apenas em uma direção;
- Com os links XML pode-se criar links fora do documento fonte, possibilitando, assim, a criação de um banco de dados de links (linkbase). Os links HTML estão embutidos no documento fonte, não sendo possível criar links fora do documento;
- Os links XML podem conectar mais de dois recursos, enquanto que os links HTML conectam apenas dois recursos.

Uma das principais vantagens da implementação de relacionamentos XML através de links é decorrente da independência entre os recursos e seus links. Devido a esta separação, os recursos não precisam ter seu conteúdo alterado para participarem de um relacionamento. Como acontece na HTML que, para implementar uma referência através da tag<A HREF>, é necessário alterar o conteúdo da origem do link (onde este será inserido) [Maler 2001].

Apesar do XLink ser uma recomendação da W3C, o que significa que é uma tecnologia estável, pronta para ter suporte da indústria, os browsers padrão (Internet Explorer e Netscape) até o momento não dão suporte completamente o XLink.

3.2. Princípios do projeto de XLink

Os princípios do projeto do XLink são [Maler 1998a]:

- **XLink deve ser diretamente utilizável na Internet.**

Pelo fato dos sites mudarem de local, páginas serem eliminadas e informações serem modificadas, o XLink deve acomodar a situação de links quebrados, recursos que não podem ser localizados e links que levam o usuário para a direção errada. O XLink também suporta links multidirecionais em aplicativos de software.

- **XLink deve ser utilizável por uma ampla gama de domínios de emprego de links e por classes de software aplicativo para links**

Quando usa-se links na XML, não deve haver nenhum favoritismo de um domínio sobre o outro. Independemente para onde o link aponta ou que tipo de documento o link será armazenado – por exemplo, uma referência cruzada em uma publicação técnica ou um link para uma página Web – não deve haver nenhum tipo de preferência. Além disso, não deve haver nenhum tipo de preferência para vários tipos de navegadores, software aplicativos ou sistemas de edição que são usados para criar ou exibir os links.

- **A linguagem de expressão do XLink deve ser XML, significando que XLink segue as mesmas regras e sintaxe.**

A premissa aqui é de que qualquer tipo de estrutura deve seguir a sintaxe de elementos e atributos XML. Pelo fato de poder usar vários elementos – embora em HTML, usa-se apenas um elemento simples (como o elemento <A>) quando se coloca um link dentro de um elemento deve-se seguir a sintaxe padrão de XML, independentemente do link utilizado. E pelo fato de se poder desenhar os próprios elementos de link, precisa-se fornecer os atributos, as informações de conteúdo e os parâmetros da mesma maneira que se faz ao criar e especificar qualquer elemento.

- **O projeto do XLink deve ser preparado rapidamente.**

Os links e o código usado para criar links devem ser fáceis de ser criados. O código usado para criar os links deve ser objetivo e fácil de ser compreendido. Deve-se estar apto a criar links rapidamente, pois já se sabe como criar elementos e atributos.

- **O projeto de XLink deve ser formal e conciso.**

O link especificado deve ser explicado de tal maneira que a explicação não confunda as pessoas que lerão o código. E o código deve ser conciso o bastante de modo que os computadores também possam entendê-lo. A idéia é a de não complicar a maneira pela qual os links se relacionam entre si e a sintaxe XML usada para definir os links. Pode-se, por exemplo, incluir informações específicas dentro da declaração de elementos de links sobre o número de recursos necessários em um link, conseqüentemente definindo melhor a topologia do link.

- **Os links do XLink devem ser legíveis aos usuários por meio de seus rótulos e ao mesmo tempo compreensíveis.**

Na verdade, isso significa mais do que simplesmente garantir que os links sejam legíveis. As estruturas dos links podem estar em formato compactado, criptografado ou binário ao serem transmitidas ou processadas internamente. Porém, elas devem estar na forma de texto dentro do documento XML para serem consideradas XLinks.

- **Os links do XLink podem residir fora dos documentos nos quais os recursos participantes residem.**

Os links podem ser armazenados dentro ou fora de documentos. Para oferecer escalabilidade e liberação das limitações dos links da HTML, o XLink deve suportar links sofisticados. Isto significa que alguns links podem ser out-of-line[3] ou inline[4]. Fica a critério do desenvolvedor decidir que tipo de links usar para qual finalidade.

- **XLink deve representar a estrutura abstrata e o significado dos links.**

Deve haver alguma pequena indicação sobre o comportamento básico do link. Os projetistas da especificação XLink não queriam encorajar a marcação procedimental; eles queriam indicar que comportamento básico de um link seria aceitável.

- **Os links do XLink devem ser factíveis de ser implementados.**

Embora alguns dos recursos de link sejam difíceis de ser implementados devido à sua complexidade, os links devem ser no mínimo fáceis de ser gerenciados e controlados e relativamente fáceis de ser moldados.

3.3. Terminologia XLink

XLink compreende vários termos dentre os quais são destacados:

[3] out-of-line- termo do XLink que descreve um link cujo conteúdo não serve como um dos recursos do link original. Os links out-of-line não têm de ocorrer no mesmo documento. São usados dentro de links multidirecionais [Pitts-Moultis and Kirk 2000].

[4] inline – termo do XLink que descreve um link que serve como um de seus recursos. Um exemplo de tal link é o elemento <A> da HTML [Pitts-Moultis and Kirk 2000].

.

Link - Uma conexão ou relacionamento entre dois ou mais objetos de dados ou porções de objetos de dados. Um link define um relacionamento específico entre dois ou mais recursos ou parte desses recursos;

Atributo - Um par nome-valor dentro de um elemento entre tags que modifica certas características do elemento;

Recurso - É qualquer objeto endereçável por um link, isto é, qualquer coisa que possa ser acessado através da rede. Por exemplo, arquivos, imagens, documentos e programas;

Recurso Local - É um conteúdo de link inline. Se o documento contendo o link for um dos recursos do link, então o documento será considerado como um recurso local;

Recurso Participante - Um recurso que faz parte de um link. Qualquer recurso é um link em potencial e se torna um link participante quando ele é identificado por um localizador como parte do link;

Recurso Remoto - Descreve qualquer recurso participante de um link ao qual um localizador aponta;

Elemento de link (Linking element) - é um elemento especial em um documento XML que certifica a existência de um link e contém uma descrição das características do link;

Linkbase - Coleção de links;

Travessia (Traversal) - é a ação de usar um link para acessar um recurso. A travessia pode ser iniciada pela ação de um usuário (por exemplo, dando um clique em uma porção exibida em um elemento de link) ou pode ocorrer sob o controle de um programa;

Arc - É a informação sobre como atravessar um par de recursos, inclusive a direção da travessia e possivelmente a informação de comportamento de aplicação. Se dois arcos em um link especificam o mesmo par de recursos, mas eles trocam de lugares como iniciando e finalizando recursos, então o link é multidirecional, que não é o mesmo que somente retornar depois de atravessar um link;

Links Outbound - links que são associados do recurso local ao recurso remoto;

Links Inbound - links que são associados do recurso remoto ao recurso local;

Links Third-Party - links que são associados do recurso remoto ao recurso remoto.

Localizador (Locator) - uma string de caracteres que aparece em um documento de link, endereço de um recurso, e pode ser usado para localizar aquele recurso;

***URI* (*Uniform Resource Identifier*)** - É o termo genérico que se utiliza para uma seqüência que define a localização de um recurso. Um URL é um exemplo de um URI;

***URL* (*Uniform Resource Locator*)**- Define a localização de um determinado recurso da Internet.

3.4. Tipos de Links

Existem dois tipos de links em XML, que são: os links simples (simple link) e os links estendidos (extended link). Esses links são criados com atributos, e não com elementos específicos, como na HTML. Pois em XML não existe um conjunto fixo de elementos, uma vez que qualquer elemento pode ser um link, desde que este elemento defina a existência e descreva as características de um link, utilizando o mecanismo de Namespace: **http://www.w3.org/1999/xlink,** para que haja assim o reconhecimento de link por aplicações capazes de realizar o processamento adequado. Na Figura 3-1 é mostrada a sintaxe do XLink.

```
<myElement
  xmlns:xlink="http://www.w3.org/1999/xlink"><!--mecanismo de Namespace-->
          .
          .
          .   >            <!–tag de abertura do elemento de link-->
</myElement>               <!–tag de fechamento do elemento de link-->
```

Figura 3-1 Sintaxe do XLink

3.4.1. Links Simples (Simple Links – Links unidirecionais)

Segundo a especificação de XLink da W3C, um link simples é um link que associa exatamente dois recursos, um local e um remoto, com um arco que vai do local para o remoto. Assim, um link simples é sempre um link outbound.

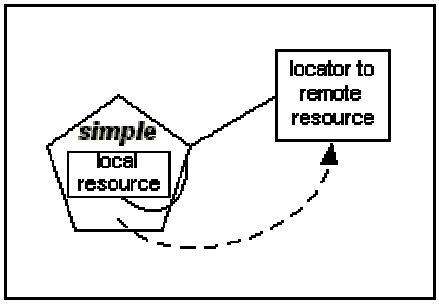


Figura 3-2 Representação do XLink Simples [DeRose 2001]

Características dos Links Simples

As características dos links simples são [Ray 2001]:

- Existem dois recursos envolvidos no link: um recurso local, que mantém a informação do link, e um recurso remoto. O recurso local precisa estar dentro de um documento XML;
- O link define um destino, que identifica o recurso remoto;
- O comportamento do link é definido por características, expressos através de atributos no elemento do link;

As características são as seguintes:

  - A atuação do link descreve como ele é disparado. Ele pode ser automático, como no caso de um gráfico importado para o documento; ou pode exibir interação do usuário, por exemplo, um leitor deverá dar um clique em um link de hipertexto para dizer ao browser para seguir o link;
  - O link pode fazer coisas diferentes com o recurso remoto. Ele pode incorporar o conteúdo na formatação do documento local, ou pode realmente substituir o documento local pelo recurso remoto;

- Pode haver alguma informação associada a um link, como um rótulo de texto ou uma descrição curta.

Aplicabilidade dos links simples

Com os links simples pode-se fazer [Ray 2001]:

- Dividir um documento entre vários arquivos e usar links para conectá-los;

- Oferecer navegação entre componentes de documento usando links para criar um menu de destinos importantes, um sumário ou um índice;
- Fazer citações para outros documentos em qualquer lugar na Internet, com links oferecendo um meio de buscá-los e exibí-los;
- Importar dados ou texto e exibí-los no documento usando links para incluir figuras, saída do programa ou trechos de outros documentos;
- Oferecer uma apresentação na mídia. Pode-se vincular um link a um clipe de vídeo ou som para incluí-los na sua apresentação.

3.4.2. Links Estendidos (Extended Links – Links multidirecionais)

Segundo a especificação de XLink da W3C, um link estendido é composto de conexões entre um conjunto de recursos, que podem ser locais ou remotos [DeRose 2001].

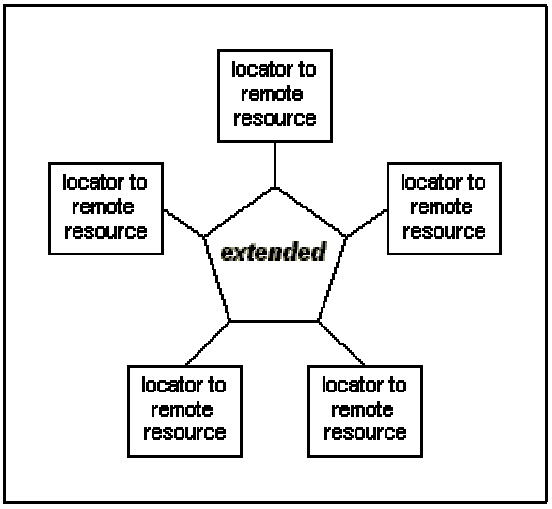


Figura 3-3 Representação do XLink Estendido [DeRose 2001]

Características dos links estendidos

As características dos links estendidos são:

- Um link estendido é composto de conexões entre um conjunto de recursos;

- Os links estendidos oferecem uma completa funcionalidade do XLink, como links inbound e third-party, como também um link que tem números arbitrários de recursos participantes. Como resultado a estrutura pode ser bastante complexa, inclusive elementos apontando para recursos remotos, elementos contendo recursos locais, elementos especificando regras de arco de travessia e elementos com rótulos que são legíveis a humanos e a máquinas [DeRose 2001];
- Um link estendido pode envolver vários recursos, vários caminhos entre esses recursos, caminhos bidirecionais e links.

Aplicabilidade dos links estendidos

Com os links estendidos pode-se fazer o seguinte [Holzner 2001]:

- Configurar um link para apontar para vários sites espelhos de um site principal e deixar que o browser selecione o que estiver mais próximo;
- Vincular um conjunto inteiro de documentos (incluindo subconjuntos), no qual o browser deverá procurar o recurso desejado;
- Configurar uma série de caminhos que permitam ao usuário navegar entre um conjunto de documentos em várias direções.
- Armazenar um link estendido separadamente de todos os recursos que estão sendo associados.

3.5. Criação de Links: Simples e Estendidos

3.5.1. Links Simples - Passo a Passo

1º Passo: **Definir um Elemento de Link *(Linking element)*:**

Um elemento de link é um elemento especial em um documento XML que certifica a existência de um link e contém uma descrição das características do link.

Seguindo as mesmas regras da sintaxe XML, é composto de uma tag de abertura, onde vai constar a descrição das características do link, e uma *tag* de fechamento, conforme mostra a Figura 3-4.

```
<element_link
            .           <!--descrição das características do link-->
            .
               >        <!–tag de abertura do elemento de link-->
</element_link>         <!–tag de fechamento do elemento de link-->
```

Figura 3-4 Definição de elemento de link

Um elemento de link é definido tanto para a criação de links simples quanto para links estendidos.

2º Passo: **Utilizar o mecanismo de Namespace de XLink**

Como já dito anteriormente é necessário utilizar o mecanismo de Namespace para que o elemento de link seja reconhecido por aplicações e estas sejam capazes de realizar o processamento adequado.

3º Passo: **Utilizar o atributo que identifica o tipo do elemento (atributo de definição de tipo:type)**

Para identificar o tipo do elemento de link, utiliza-se o atributo *type.* No caso de links simples este atributo terá sempre o valor "*simple*", no caso de links estendidos o valor será *extended.*

Sintaxe: `xlink:type = <type>  <type>::=“simple”/”extended”`

A Figura 3-5 mostra um exemplo de XLink simples.

```
<SITE
            xmlns:xlink= “http://www.w3.org/1999/xlink”
            xlink:type = “simple”> SITE OFICIAL XML
<SITE>
```

Figura 3-5 Exemplo de um XLink simples

*Observação***:** É importante colocar alguma informação, sempre após a tag de abertura, que identifique o link. Por exemplo, no código acima, o texto SITE OFICIAL W3C mostra ao usuário que é um link.

4º Passo: **Utilizar o atributo localizador/locação: href (opcional)**

O link simples contém somente um alvo, o qual é armazenado no próprio objeto fonte, usando o atributo *href.*

Este atributo fornece os dados que permitem que uma aplicação encontre um recurso (ou fragmento de recurso) remoto.

É o URI de destino do link e tem a seguinte sintaxe:

Sintaxe: `xlink:href = <URI de referência que identifica um recurso remoto>`

A Figura 3-6 mostra um exemplo de XLink Simples usando o atributo de localização *href.*

```
<SITE
            xmlns:xlink= “http://www.w3.org/1999/xlink”
            xlink:type = “simple”
            xlink:href = “http://www.w3.org”>SITE OFICIAL XML
</SITE>
```

Figura 3-6 Exemplo de XLink Simples usando o atributo de localização *href*

5º Passo: **Utilizar atributos que especificam como o link deve ser tratado: Atributos de Comportamento (opcional)**

Os atributos de comportamento são os atributos que controlam o método de ativação (atributo *actuate*) e o estilo de exibição (atributo *show*).

Atributo *actuate*

Para especificar quando um XLink deve ser atravessado utiliza-se o atributo *actuate,* que pode ter os seguintes valores: “*onRequest”, “onLoad”, “other”, “none*”. Como apresentado na Tabela 3.1.

Tabela 3-1 Tabela de valores e funções do atributo *actuate*

Atributo	Valor	Sintaxe	Função

Actuate	“onRequest”	xlink:actuate= “onRequest”	Solicitar uma ação do usuário para que o recurso seja ativado (como um link HTML padrão).
	“onLoad”	xlink:actuate= “onLoad”	Ativar o recurso automaticamente sem solicitar uma ação do usuário (como imagem da HTML).
	“other”	xlink:actuate = “other”	Deixar que a aplicação use outra marca, não definida pelo XLink, presente no link para determinar como o link deverá ser ativado.
	“none”	xlink:actuate = “none”	Nenhum comportamento é especificado.

Atributo *show*

Para descrever o comportamento de exibição de um link, após ter sido acionado (seja automaticamente ou pelo usuário) e atravessado, utiliza-se o atributo *show* que pode ter os seguintes valores: *“new”, “replace”, “embed”, “other”, “none”.* Como apresentado na Tabela 3.2.

Tabela 3-2 Tabela de valores e funções do atributo show

Atributo	Valor	Sintaxe	Função
show	“replace”	xlink:show = “replace”	Substituir a janela atual com o recurso de destino.
	“new”	xlink:show = “new”	Exibir o recurso de destino em uma nova janela.
	“embed”	xlink:show = “embed”	Incluir o conteúdo de destino no conteúdo de origem (como o elemento IMG em HTML).
	“other”	xlink:show = “other”	Deixar que a aplicação use outra marca presente no link para determinar como o link deverá ser exibido.
	“none”	xlink:show = “none”	Nenhum comportamento é especificado.

***6º Passo:* Utilizar atributos que descrevem o significado de um XLink: atributos de semântica (opcional)**

Os atributos de semântica (*role e title*) descrevem o significado dos recursos dentro do contexto de um link. Conforme apresentado na Tabela 3.3.

Tabela 3-3 Tabela de valores e funções dos atributos semânticos

Atributo	Valor	Sintaxe	Função
role	Referência de uma URI.	xlink:role = <URI>	Descrever o significado de um recurso em um relacionamento. O seu valor identifica algum recurso que descreve a propriedade especificada. Projetado para ser lido pelo software.
title	Descrição curta do link	xlink:title=<descrição do link>	Descrever o significado de um recurso de um modo compreensível para o homem.

Observação:

O desenvolvedor não precisa necessariamente seguir a mesma ordem dos passos e nem obrigatoriamente utilizar todos os atributos num mesmo código uma vez que, como a maioria deles é opcional, fica a critério utilizá-los ou não.

A Figura 3-7 mostra um exemplo de um XLink simples usando os atributos de comportamento e de semântica:

```
<SITE
       xmlns:xlink="http://www.w3.org/1999/xlink"
       xlink:type = "simple"
       xlink:href= "http://www.w3c.org/"
       xlink:actuate = "onRequest"
       xlink:show = "new"
       xlink:title= "SITE W3C">SITE OFICIAL XML
</SITE▫
```

Figura 3-7 Exemplo de XLink simples usando os atributos de comportamento e de semântica

Neste exemplo o link vai ser ativado com uma ação do usuário e seu conteúdo será exibido em uma nova janela.

A utilização de XLink pode se dar ao nível da especificação do DTD do documento, como no exemplo de declaração de links simples apresentada na Figura 3-8.

```
<!-- SITE = simple-type -->
<!ELEMENT SITE ANY>
<!ATTLIST SITE
XLink:type (simple) #FIXED "simple"
XLink:href CDATA #IMPLIED
XLink:role NMTOKEN #FIXED "student"
XLink:title CDATA #IMPLIED
XLink:show (new
           |replace
           |embed
           |other
            |none) #FIXED "new"
XLink:actuate (onLoad
              |onRequest
```

```
                              |other
                              |none) #FIXED "onRequest">
```

Figura 3-8 DTD do documento para o elemento <SITE>

Possíveis combinações com atributos: *show e actuate*

- `xlink:show= “new” xlink:actuate= “onLoad”`

Quando esses dois atributos são utilizados com esses valores num só código, uma nova janela vai ser aberta imediatamente, após o documento XML contendo o XLink ser carregado. A Figura 3-9 mostra um exemplo de XLink simples com esse tipo de combinação.

```
<SITE
            xmlns:xlink= “http://www.w3.org/1999/xlink”
            xlink:type = “simple”
            xlink:href = “http://www.w3.org”
            xlink:actuate = “onLoad”
            xlink:show= “new”> SITE OFICIAL XML
</SITE>
```

Figura 3-9 Exemplo de XLink simples com a combinação dos atributos *show= “new”* e *actuate= “onLoad”*

- `xlink:show=“new” xlink:actuate= “onRequest”`

Com esta combinação, o link não vai ser ativado automaticamente, pois vai precisar que o usuário tome alguma medida para que o link seja ativado. E quando for acionado, ele vai ser carregado numa nova janela. A Figura 3-10 mostra um exemplo de XLink simples com esse tipo de combinação.

```
<SITE
                  xmlns:xlink= “http://www.w3.org/1999/xlink”
                  xlink:type = “simple”
                  xlink:href = “http://www.w3.org”
                  xlink:actuate = “onRequest”
                  xlink:show= “new”> SITE OFICIAL XML
</SITE>
```

Figura 3-10 Exemplo de XLink simples com a combinação dos atributos show= “new”e actuate= “onRequest”

- `xlink:show= “replace”  xlink:actuate= “onRequest”`

Com a combinação desses atributos e valores, o link vai ser ativado com uma ação do usuário e o conteúdo destino substituirá a janela atual. A Figura 3-11 mostra um exemplo de XLink simples com esse tipo de combinação.

```
<SITE
                  xmlns:xlink= “http://www.w3.org/1999/xlink”
                  xlink:type = “simple”
                  xlink:href = “http://www.w3.org”
                  xlink:actuate = “onRequest”
                  xlink:show= “replace”>SITE OFICIAL XML
</SITE>
```

Figura 3-11 Exemplo de XLink simples com a combinação dos atributos show= “replace”e actuate= “onRequest”

- `xlink:show= “replace”  xlink:actuate= “onLoad”`

A combinação desses dois atributos e valores num só link resultará, após o carregamento do documento, o link substituirá a página atual de forma automática. A Figura 3-12 mostra um exemplo de XLink simple com esse tipo de combinação.

```
<SITE
            xmlns:xlink= “http://www.w3.org/1999/xlink”
            xlink:type = “simple”
            xlink:href = “http://www.w3.org”
            xlink:actuate = “onLoad”
            xlink:show= “replace”> SITE OFICIAL XML
</SITE>
```

Figura 3-12 Exemplo de XLink simples com a combinação dos atributos show= “replace”e actuate= “onLoad”

- `xlink:show= “embed”     xlink:actuate= “onLoad”`

Quando esses dois atributos são utilizados com esses valores num só link, o recurso apontado pelo link vai ser embutido no documento atual automaticamente, sem precisar que o usuário tome alguma medida para que o link seja embutido. Isto é similar a embutir uma imagem em HTML (`<html:img src=... >`). A Figura 3-13 mostra um exemplo de XLink simples com esse tipo de combinação.

```
<SITE
            xmlns:xlink= “http://www.w3.org/1999/xlink”
            xlink:type = “simple”
            xlink:href = “w3c_home.gif”
            xlink:actuate = “onLoad”
            xlink:show= “embed”> LOGOMARCA DO W3C
</SITE>
```

Figura 3-13 Exemplo de XLink simples com a combinação dos atributos show= “embed”e actuate= “onLoad”

3.5.2. Links Estendidos – Passo a Passo

1º Passo: Definir um Elemento de Link *(Linking element)*:

Da mesma forma que o link simples.

2º Passo: Utilizar o mecanismo de Namespace de XLink

Exemplo: `xmlns:xlink= “http://www.w3.org/1999/xlink”`

3º Passo: UTILIZAR O ATRIBUTO QUE IDENTIFICA O TIPO DO ELEMENTO: (atributo de definição de tipo:*type*)

No caso de links estendidos (extended links) este atributo terá sempre o valor “*extended*”.

Sintaxe: `xlink:type = “extended”`

Num link estendido o atributo *type* é usado também para identificar os quatro tipos de subelementos que podem constar num link estendido. Que são: *locator, resource, arc, title*

- ✓ ***xlink:type = "locator"***
- ✓ ***xlink:type = "resource"***
- ✓ ***xlink:type = "arc"***
- ✓ ***xlink:type = "title"***

Os atributos semânticos *role* e *title* também podem ser usados com o link estendido. E eles realizam a mesma função que realizam no elemento de link simples, sendo também atributos opcionais. A Figura 3-14 mostra um exemplo de link estendido.

```
<EXTENDEDLINK
                  xmlns:xlink= "http://www.w3.org/1999/xlink"
                  xlink:type= "extended"
                  xlink:role= "http://www.w3.org/XML"
                  xlink:title= "SITE OFICIAL XML">
</EXTENDEDLINK>
```

Figura 3-14 Exemplo de um link estendido

4º Passo: Definir o tipo de Subelemento

Para definir o tipo de subelemento, é necessário saber como ele é usado. A Tabela 3-4 apresenta a função de cada um.

Tabela 3-4 Tabela de definição dos subelementos

Subelemento	Sintaxe	Função
resource	xlink:type ="resource"	Localizar um recurso local: é usado para definir os participantes locais do link.
locator	xlink:type ="locator"	Localizar recursos remotos, que podem estar no mesmo documento ou em outro documento.
arc	xlink:type ="arc"	Definir conexões entre dois localizadores participantes de um link estendido.
title*	xlink:type ="title"	Associar informação semântica com um link estendido. Permite mais informação do que um atributo "title" sozinho.

* O subelemento *title* tem somente significado específico de XLink se for subelemento de um elemento do tipo *extended, locator ou arc*.

Observação:

Para cada definição de tipo de subelemento, é necessário definir antes um elemento de link.

5º Passo: DEFINIR OS ATRIBUTOS QUE PODEM CONSTAR NOS TIPOS DE SUBELEMENTOS

Subelemento resource

O subelemento do tipo ***resource*** pode ter os atributos semânticos *(role e title)* e o atributo de travessia (*label*), conforme mostra a Tabela 3-5.

Tabela 3-5 Tabela de atributos do subelemento *resource*

Atributos	Sintaxe	Função
role	xlink:role= <referência a URI>	Descrever o significado de um recurso em um relacionamento. Projetado para ser lido pelo software.
title	xlink:title= <descrição curta do link>	Descrever o significado de um recurso de um modo compreensível para o homem.
label	xlink:label = <rótulo de texto>	Fornecer rótulos que são referenciados pelos atributos *from* e *to,* do subelemento arc.

A Figura 3-15 mostra um exemplo de link estendido com o subelemento resource [Harold 2001]:

```
<WEBSITE
      xmlns:xlink= http://www.w3.org/1999/xlink
      xlink:type = “extended”       xlink:title= “Cafe au Lait”>
      <NAME xlink:type = “resource” xlink:label= “source”>
            Cafe au lait
      <NAME>
      .
      .
      .

<WEBSITE>
```

Figura 3-15 Exemplo de link estendido com o subelemeto *resource*

Subelemento locator

O subelemento do tipo ***locator*** pode ter também os atributos semânticos *(role e title*) e o atributo de travessia (*label*), como também o atributo *href*. Conforme mostra a Tabela 3-6.

Tabela 3-6 Tabela de atributos do subelemento *locator*

Atributos	Sintaxe	Função
role	xlink:role= <referência a URI>	Descrever o significado de um recurso em um relacionamento. Projetado para ser lido pelo software.
title	xlink:title= <descrição curta do link>	Descrever o significado de um recurso de um modo compreensível para o homem.
label	xlink:label = <rótulo de texto>	Fornercer rótulos que são referenciados pelos atributos *from* e *to*, do subelemento arc.
href*	xlink:href= <referência a um URI>	Localizar um recurso ou fragmento de recurso.

* O atributo *href* realiza a mesma função que realiza nos links simples, porém no subelemento *locator* ele é obrigatório, ou seja, um valor é exigido neste atributo.

A Figura 3-16 apresenta um exemplo de link estendido com o subelemento *locator* [Harold 2001]:

```
<WEBSITE xmlns:xlink="http://www.w3.org/1999/xlink"
         xlink:type="extended" xlink:title="Cafe au Lait">

  <NAME xlink:type="resource" xlink:label="source">
    Cafe au Lait
  </NAME>

  <HOMESITE xlink:type="locator"
            xlink:href="http://ibiblio.org/javafaq/"
            xlink:label="us"/>

  <MIRROR xlink:type="locator"
          xlink:href="http://sunsite.kth.se/javafaq"
          xlink:title="Cafe au Lait Swedish Mirror"
          xlink:label="se"/>

  <MIRROR xlink:type="locator"
          xlink:href=http://sunsite.informatik.rwth-aachen.de/javafaq/
          xlink:title="Cafe au Lait German Mirror"
          xlink:label="sk"/>

  <MIRROR xlink:type="locator"
          xlink:href=http://sunsite.cnlab-switch.ch/javafaq/
          xlink:title="Cafe au Lait Swiss Mirror"
          xlink:label="ch"/>

</WEBSITE>
```

Figura 3-16 Exemplo de um link estendido com o subelemento *locator*.

Subelemento arc

O subelemento do tipo **arc**, pode ter os atributos de travessia (*from e to*), atributos de comportamento (*show e actuate*) e atributos semânticos (*role, arcrole e title*), como mostra a Tabela 3-7:

Tabela 3-7 Tabela de atributos do subelemento arc

Atributos	Sintaxe	Função
from	xlink:from	Definir o ponto inicial do link
to	xlink:to	Definir o ponto final do link.
show	xlink:show= (“replace” | “new”| “embed”| “other”| “none”)	Controlar o método de exibição.
actuate	xlink:actuate= (“onLoad” | “onRequest” | “other” | “none”)	Controlar o método de ativação
arcrole	xlink:arcrole= <URI>	Define o significado de um arco em um determinado *link*, permitindo que existam vários relacionamentos entre um dado par de participantes.
title	xlink:title= <descrição curta do link>	descrição curta do link

A Figura 3-17 apresenta um exemplo de XLink estendido com o subelemento arc [Harold 2001].

```
<WEBSITE xmlns:xlink="http://www.w3.org/1999/xlink"
         xlink:type="extended" xlink:title="Cafe au Lait">

  <NAME  xlink:type="resource" xlink:label="source">
    Cafe au Lait</NAME>

  <HOMESITE xlink:type="locator"
            xlink:href="http://ibiblio.org/javafaq/"
            xlink:label="us"/>

  <MIRROR  xlink:type="locator"
           xlink:href="http://sunsite.kth.se/javafaq"
           xlink:title="Cafe au Lait Swedish Mirror"
           xlink:label="se"/>

  <MIRROR xlink:type="locator"
          xlink:href="http://sunsite.informatik.rwth-aachen.de/javafaq/"
          xlink:title="Cafe au Lait German Mirror"
          xlink:label="sk"/>

  <MIRROR xlink:type="locator"
          xlink:href="http://sunsite.cnlab-switch.ch/javafaq/"
          xlink:title="Cafe au Lait Swiss Mirror"
          xlink:label="ch"/>

  <CONNECTION xlink:type="arc" xlink:from="source"
              xlink:to="ch"    xlink:show="replace"
              xlink:actuate="onRequest"/>

  <CONNECTION xlink:type="arc" xlink:from="source"
              xlink:to="us"    xlink:show="replace"
```

```
              xlink:actuate="onRequest"/>

□□<CONNECTION  xlink:type="arc" xlink:from="source"
              xlink:to="se"    xlink:show="replace"
              xlink:actuate="onRequest"/>
  <CONNECTION xlink:type="arc" xlink:from="source"]
              xlink:to="sk"    xlink:show="replace"
              xlink:actuate="onRequest"/>
</WEBSITE>
```

Figura 3-17 Exemplo de link estendido com subelemento arc

Observação:

Este exemplo mostra uma conexão de um recurso local para cada recurso remoto. Como os recursos remotos possuem diferentes *labels* haverá assim várias conexões que corresponderá ao *label* de cada recurso remoto. Se o label fosse igual, necessitaria apenas de uma conexão para fazer o relacionamento entre os recursos. Dessa forma o código ficará bastante reduzido.

Subelemento title

Uma situação comum para o uso do tipo de elemento ***title*** é no caso da internacionalização (aceitação do nível de marcação em qualquer lugar do planeta) ou na localização.

Este subelemento pode ter o atributo *xml:lang*, que é um rótulo de idioma para qualquer elemento e o seu valor é uma string contendo um código de idioma com duas letras, assim:

Ex.: `xml:lang = “en”`

O código “en” significa English. Geralmente este atributo é usado quando se quer combinar várias versões de um texto em um documento e cada uma delas é rotulada com uma versão diferente.

Na Tabela 3-8 encontram-se os tipos de elementos (colunas) nos quais os atributos globais (linhas) são permitidos, com uma indicação se um valor é requerido (R) ou opcional (O) [DeRose 2001].

Tabela 3-8 Tabela dos atributos globais

	Atributos	**simple**	**extended**	**locator**	**arc**	**resource**	**title**
Definição de tipo	type	R	R	R	R	R	R
Localização	href	O		R			

Semântica	role	O	O	O		O	
	arcrole	O			O		
	title	O	O	O	O	O	
Comportamento	show	O			O		
	actuate	O			O		
Transversal	label			O		O	
	from				O		
	to				O		

Linkbase

Viu-se que um link estendido é composto de associações entre vários recursos, podendo ser locais e remotos. Porém, os links estendidos também podem funcionar através dos arcos *third-party*, que são aqueles links constituídos apenas de recursos remotos. Documentos que contêm coleções de links *inbound* e *third-party* são chamados bases de dados do link, ou linkbases.

Um linkbase é um conjunto de links estendidos definidos em um arquivo separado e independente.

Para que um arquivo possa utilizar o linkbase, o desenvolvedor deve informá-lo da existência do linkbase, através do atributo *arcrole.* A Figura 3-18 mostra a sintaxe do atributo arcrole com seu respectivo valor.

```
xlink:arcrole = “http://www.w3.org/1999/xlink/properties/linkbase”
```

Figura 3-18 Sintaxe do atributo arcrole informando a existência do linkbase

Exemplo de como criar um Linkbase

Cria-se um linkbase da mesma forma que se cria um link qualquer. Primeiramente definindo um elemento de link e fornecendo todas as características de um link, não esquecendo de que um linkbase faz parte de um link estendido [Martin 2001].

Imagine o seguinte problema:

- Tem-se um arquivo mostrado na Figura 3.19. Este arquivo contém [Martin 2001]:

- ✓ Página de índice com um link para cada página de categoria de alimento;
- ✓ Cada página de categoria de alimento possui um link para a página anterior e a próxima e também um link para o índice

Problema: renomear, mover, adcionar ou excluir os arquivos de categoria, sem precisar alterar cada página.

Solução: Criar um linkbase. Caso necessite fazer qualquer tipo de alteração, basta modificar apenas o banco de dados de links, sem precisar alterar cada documento participante no link. Portanto a separação de link do conteúdo é uma maneira poderosa de controlar e manter links entre documentos [Martin 2001], facilitando assim o manejamento ou manutenção do link.

Exemplo do documento *menulink.xml* é apresentado na Figura 3-19.

```
<MENU xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="extended">


      <INDEX    xlink:type="locator" xlink:href="index .xml"
                xlink:label="index"/>
    <CATEGORIA  xlink:type="locator" xlink:href="menu1.xml"
                xlink:title="BEBIDAS" xlink:label="1" />
    <CATEGORIA  xlink:type="locator" xlink:href="menu2.xml"
                xlink:title="MASSAS" xlink:label="2" />
    <CATEGORIA xlink:type="locator" xlink:href="menu3.xml"
                xlink:title="APERITIVOS" xlink:label="3" />
    <CATEGORIA xlink:type="locator" xlink:href="menu4.xml"
                xlink:title="SOBREMESAS" xlink:label="4" />


  - <!-- Link Anterior  -->


    <CONEXAO xlink:type="arc" xlink:from="2" xlink:to="1" />
    <CONEXAO xlink:type="arc" xlink:from="3" xlink:to="2" />
    <CONEXAO xlink:type="arc" xlink:from="4" xlink:to="3" />


  - <!-- Proximo Link -->


    <CONEXAO xlink:type="arc" xlink:from="1" xlink:to="2" />
    <CONEXAO xlink:type="arc" xlink:from="2" xlink:to="3" />
    <CONEXAO xlink:type="arc" xlink:from="3" xlink:to="4" />
 </MENU>
```

```
          Exemplo de um linkbase:

<LINKBASE      xmlns:xlink="http://www.w3.org/1999/xlink"
               xlink:type="extended">
   <START      xlink:type="resource" xlink:label="link" />
   <LINKS      xlink:type="locator" xlink:label="linkbase"
               xlink:href="menulink.xml" />
   <LOAD       xlink:type="arc"
               xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkba
               se" xlink:from="linkt" xlink:to="linkbase"
               xlink:actate="onLoad" />
</LINKBASE>
```

Figura 3-19 Exemplo de um linkbase

3.6. Considerações Finais

Diante do que foi discutido sobre links XML, pode-se dizer que esses links oferecem uma maior flexibilidade e funcionalidade, fornecendo uma gama muito mais poderosa de opções para links que a HTML possui.

O estudo realizado neste capítulo servirá como base para a implementação dos links em documentos XML, abordado no Capítulo 5.

Capítulo 4 XPointer

Neste Capítulo uma visão geral sobre XPointer é apresentada, descrevendo conceitos, princípios do projeto, origem e uso do XPointer, baseada na especificação XPointer desenvolvida pela W3C.

4.1. Introdução

Devido a necessidade de endereçar qualquer região arbitrária dentro de um documento XML, o W3C decidiu criar um componente complementar ao XLink, que pudesse especificar fragmentos de documentos. Esse componente complementar é chamado XPointer, a linguagem de ponteiros XML [Wilde 2002].

Enquanto os XLinks especificam como um documento é vinculado a outro documento, XPointers especificam locais dentro de um documento, baseados na especificação XPath (especificação genérica para descrever locais dentro de documentos XML, projetada para satisfazer as regras sintáticas do URL) [Ray 2001].

XPointer inclui as funcionalidades da linguagem XPath com alguns recursos adicionais. O XPointer introduz os conceitos de pontos (points) e intervalos (ranges) como declarações posicionais dentro de um documento e fornece algumas funções para manipular estas novas referências de posições [Martin 2001].

Graças ao XPointer pode-se criar uma ligação para qualquer parte de uma página sem ter necessidade de uma âncora interna como na HTML, e também sem ter necessidade de modificar a página. Além de que através do XPointer pode-se localizar um ponto preciso na estrutura de um documento XML. Solucionando, assim, uma das limitações dos ponteiros HTML, que só poderia apontar para um documento alvo inteiro.

Além de XPath, outros padrões influenciaram no desenvolvimento da especificação XPointer [Pitts-Moultis 2000]:

- HTML - Este sistema popularizou um importante tipo de especificação de localização, o URL (agora URI);
- HyTime - Este padrão ISO define tipo específico de localização para todos os tipos de dados;
- Text Encoding Initiative Guidelines [TEI] - Esta aplicação fornece uma sintaxe formal para “ponteiros estendidos”, localização para marcação estruturada que dá suporte ao objetivo inicial do XPointer.

4.2. Diretrizes de projeto do XPointer

Os desenvolvedores da especificação XPointer sabiam que eles estavam criando uma recomendação associada para trabalhar em conjunto com o XLink. Para esse fim, eles definiram os seguintes objetivos para o projeto da linguagem [Maler and DeRose 1998b]:

- **Os XPointers devem ser voltados para documentos XML**

Os XPointers devem oferecer algum tipo de interoperabilidade entre o cliente e o servidor. A manipulação de strings deve funcionar não apenas entre servidores e documentos, mas também entre países, oferecendo internacionalização do XPointer em si.

- ***Os XPointers devem ser diretamente utilizáveis na Internet***

Isto significa que os XPointers devem ser compreensíveis quando lidos pelos usuários e que as referências XPointers devem ser capazes de ser usadas e vinculadas através da Internet.

- ***Os XPointers devem ser diretamente utilizáveis em URIs***

Os XPointers devem ser usados com os URIs como parte de um XLink usado dentro de um documento. Da mesma forma, deve-se manter os XPointers simples e evitar quaisquer referências adicionais que se refiram ou permitam o acesso direto a um URI.

- **O projeto do XPointer deve ser formal e conciso**

Seus XPointers devem ser construídos de forma que sejam auto-suficientes e, ao mesmo tempo, concisos o bastante para especificarem várias localizações que se queira endereçar em um documento XML. Em outras palavras, seu código XPointer não deve ser desordenado ou conter elementos e/ou atributos desnecessários.

- **A sintaxe do XPointer deve ser relativamente compacta e legível pelos usuários**

Os XPointers não devem ser extensos em suas declarações e devem se encaixar de forma organizada em um valor de atributo XML. Mais importante ainda, a referência do XPointer deve ser facilmente compreendida pelo usuário XML em geral, e o leitor deveria ser capaz de interpretá-la simplesmente observando-a.

- **Os XPointers devem ser otimizados para poderem ser utilizados**

Este é provavelmente um dos conceitos mais importantes dos XPointers. Cada um deste deve imitar facilmente a maneira pela qual as pessoas pensam em relação a encontrar informações em um documento. Isso pode significar que o XPointer realmente se refira a uma única posição de várias formas, como o usuário faria, caso estivesse se referindo à posição.

- **Os XPointers devem ser fáceis de ser implementados**

Embora o W3C entenda que os XPointers possam se tornar relativamente complexos, o objetivo é que a linguagem XPointer seja fácil de ser implementada em documentos XML existentes e novos.

Diante dessas diretrizes, pode-se observar que vários objetivos de projeto da linguagem XPointer são semelhantes àqueles do XLink.

4.3. Terminologia XPointer

O XPointer compreende vários termos, dos quais serão destacados:

Atributos ***ID*** (Identificador de Fragmento) – São atributos definidos para ter um valor exclusivo (ou seja, não repetível) em todo o documento XML;

Atributo ***IDREF*** – Atributo que contém uma lista de valores, separados por um espaço em branco, encontrados nos atributos ID do documento. Ele é uma referência a um ID em algum lugar no mesmo documento;

Ponto (point) – É um local específico em um documento;

Intervalo (range) – Representa toda a estrutura e conteúdo XML entre um ponto inicial e um ponto final;

Location - Uma generalização do nó de XPath que inclui pontos e intervalos além de nós;

Conjunto de Localização (Location Set) – Uma lista ordenada de locais;

Etapa de localização ***(Location Step***)– Fornece um meio de selecionar nós em um documento XML;

Caminho de Localização (Location Path) – Endereça um grupo de nós em um documento.

Consiste de uma ou mais etapas de localização separadas por “/”.

Sub-Recurso (Sub-Resource) – A parte do documento de XML que o XPointer identifica. Enquanto o documento de XML inteiro seria considerado o recurso, a parte acessada por XPointer seria o sub recurso.

4.4. Como usar XPointer

O XPointer é associado ao lado direito de um URL por um símbolo “#” e só pode ser usado com o atributo `xlink:href` [Will 2002].

Sintaxe: ***#xpointer (expressão)***

Onde *expressão* é uma frase XPath que identifica a parte específica do arquivo ao qual deseja-se conectar. A Figura 4-1 mostra um exemplo de XPointer.

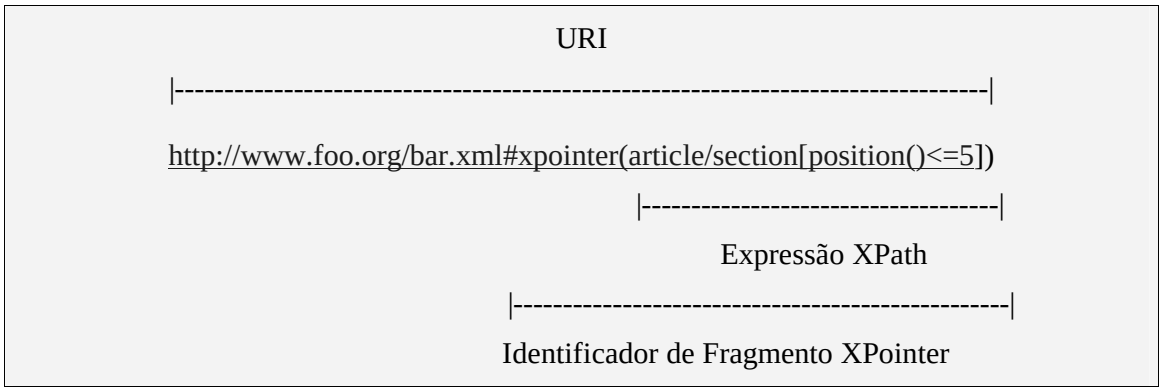


Figura 4-1 Exemplo de XPointer

4.4.1. Uma expressão XPath

Quando uma expressão XPath é processada o resultado pode ser [Harold and Means 2001]:

- Uma coleção de nós;
- Um valor booleano;
- Um número;
- Uma seqüência de caracteres.

A linguagem XPointer não manipula diretamente uma expressão XPath, mas sim o objeto resultante de sua avaliação [Kelly 1998].

XPath modela um documento XML como uma árvore de nós, conforme as Figuras 4-2 e 4-3.

```
<?xml version="1.0"?>
   <guitars>
      <guitar type=”Electric”>
         <model>Strat</model>
         <model>Tele</model>
         <model>Les Paul</model>
      <guitar/>
   </guitars>
```

Figura 4-2 Exemplo de documento XML

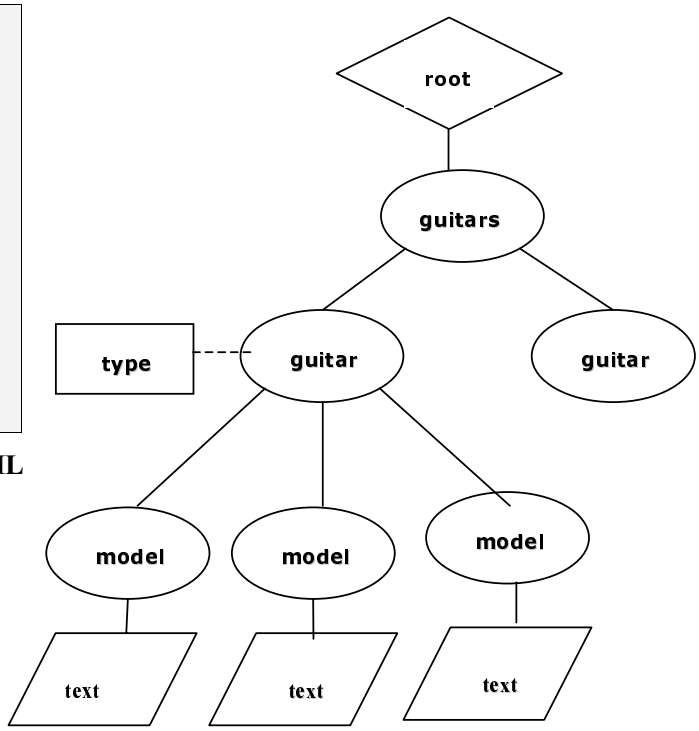


Figura 4-3 Modelo de dados XPath

As expressões XPath são aplicadas dentro de um contexto e essas expressões são construídas através da etapa de localização (Location Step), que vai fornecer um meio de selecionar nós em um documento XML.

Numa expressão pode haver mais de uma etapa de localização que é separada por “/”.

Ex: `/guitars/guitar/model`

As etapas de localizações são avaliadas da esquerda para a direita e irão operar em relação ao nó de contexto (nó atual).

Uma etapa de localização possui três partes:

- Eixos (Axis);
- Teste de Nós (Node Test);

- Predicados (Predicate)

Sintaxe : `axis::node_test [predicate].`

A Figura 4-4 mostra um exemplo de etapa de localização

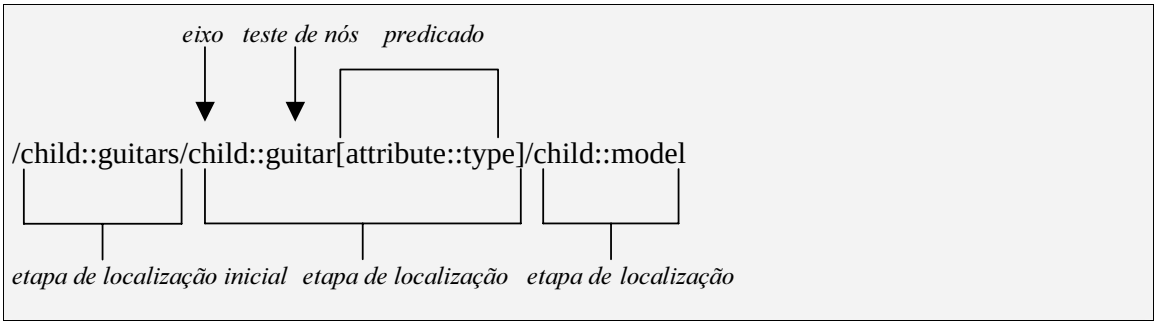


Figura 4-4 Exemplo de uma etapa de localização

Eixo*:*

Informa em que direção seguir em relação ao nó contexto.

É usado para definir uma região inicial para aplicar o predicado e o teste de nós ao avaliar a expressão. Os eixos possíveis estão apresentados na Tabela 4-1 que estão relacionados com a Figura 4-5:

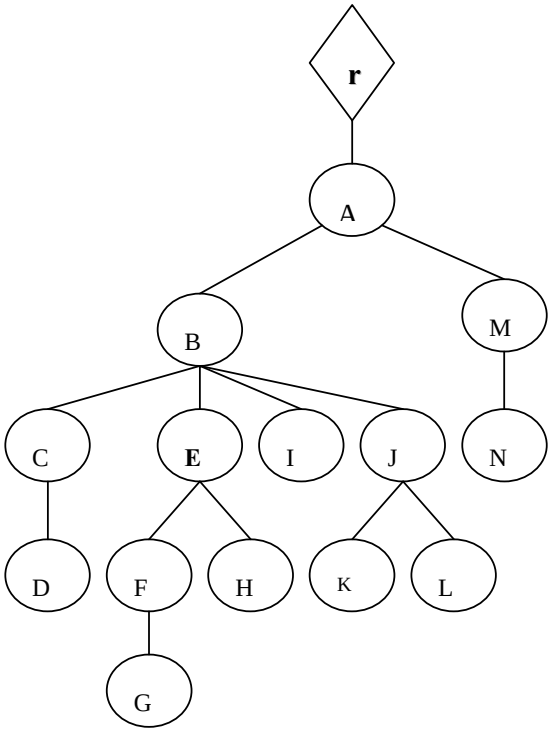


Figura 4-5 Árvore de nós XPath

Tabela 4-1 Tabela de eixos XPath

self	E
child	F, H
parent	B
descendant	F, G, H
descendant-or-self	E, F, G, H
ancestor	B, A, root
ancestor-or-self	E, F, G, H
following	I, J, K , L, M, N
following-sibling	I, J
preceding	C, D
preceding-sibling	C

Definição dos eixos [W3CSchools 2002]:

self::node() – seleciona apenas o nó de contexto;

child::node() – seleciona os filhos do nó de contexto;

parent::node() – seleciona o pai do nó de contexto, caso exista um;

descendant::node() – seleciona os descendentes do nó de contexto. Um descendente é um filho ou um neto ou bisneto e assim por diante, do nó de contexto. Um eixo descendente nunca seleciona um nó atributo ou namespace;

descendant-or-self::node() – seleciona o nó de contexto e seus descendentes;

ancestor::node() – seleciona os antepassados do nó de contexto, que são: os pais, avós e bisavós, e assim por diante, deste nó. Um eixo *ancestor* sempre terá em seu conjunto de nós o nó raiz, a menos que ele seja o próprio nó raiz;

ancestor-or-self::node() – seleciona o nó de contexto e seus antepassados;

following::node() – seleciona todos os nós no mesmo documento que o nó de contexto e que estão depois do nó de contexto, na ordem em que aparecem no documento, excluindo antepassados do nó contexto, e excluindo nós atributos e nós namespace;

following::sibling::node() – seleciona todos os nós que estão depois do nó de contexto e tem os mesmos pais no nó de contexto. Se o nó de contexto é um nó atributo ou um nó namespace, o conjunto de nós selecionados pelo eixo following-sibling será vazio;

preceding::node() – seleciona todos os nós no mesmo documento que o nó de contexto e que estão antes do nó de contexto, na ordem em que aparecem no documento, excluindo antepassados do nó de contexto, e excluindo nós atributos e nós namespace;

preceding-sibling::node() – seleciona todos os nós que estão antes do nó de contexto e tem os mesmos pais no nó de contexto. Se o nó de contexto é um nó atributo ou um nó namespace, o conjunto de nós selecionados pelo eixo preceding-sibling será vazio;

A Tabela 4-2 mostra abreviações de alguns eixos:

Tabela 4-2 Tabela de Abreviações de eixos

eixo	**Sintaxe Normal**	**Sintaxe Abreviada**
"child::" – pode ser omitido numa expressão.	/child::guitar/child::model	/guitar/model
"attribute::" – o eixo attribute pode ser abreviado para "@".	/child::guitar/attribute::type	/guitar/@type
"/descendant-or-self::node()/" – pode ser abreviado por "//"	/descendant-or-self::node()/child::guitar	//guitar
"self::node()" - este eixo pode ser abreviado para "." "parent::node() – este eixo pode ser abreviado para ".."	self::node()/parent::node()/child::model	./../model

Testes de nós

Permitem que elementos específicos ou tipos de nós sejam selecionados do eixo especificado.

Determinam quais os nós selecionados pelo eixo que farão parte do valor final da expressão. Os testes de nós possíveis são apresentados na Tabela 4-3 [Clark 1999]:

Tabela 4-3 Tabela de possíveis Testes de Nós

Testes de nós	**Significado**
*	**Seleciona todos os elementos no eixo especificado.**
nomes literais	Seleciona todos os nós elemento com determinado nome.
node( )	Seleciona qualquer tipo de nó, exceto nós atributos e namespace.
text( )	Seleciona um nó de texto. Apesar dos parênteses, o teste de nó não recebe nenhum argumento de fato.
comment( )	Seleciona todos os nós comentário. Apesar dos parênteses, o teste de nó não recebe nenhum argumento de fato.
prefix:*	Um prefixo seguido por um asterisco seleciona todos os elementos no namespace que combinam o prefixo. Por exemplo: Se o prefixo

	svg é traçado ao URI http://www.w3.org/2000/svg , então svg:* combina todos os elementos de SVG.
@prefix:*	Combina todos os atributos no namespace especificado. Por exemplo, se xlink é traçado ao URI http://www.w3.org/1999/xlink, então @xlink:* combina todos os atributos de XLink no documento como xlink:type, xlink:show, xlink:actuate, xlink:href, xlink:role e assim sucessivamente.
processing-instruction( )	Seleciona todos os nós de instruções de processamento. Dentro dos parênteses, pode-se especificar o nome da instrução de processamento a selecionar.

Predicados

Os predicados têm como função gerar um novo conjunto de nós resultantes da filtragem dos nós selecionados pelo teste de nó, utilizando como critério de filtragem o contexto atual e a expressão do predicado, que pode ser uma expressão booleana, uma chamada de função ou até mesmo uma referência a uma variável.

O XPath fornece diversas funções que podem ser usadas para testar os nós. Entre essas funções pode-se destacar:

função position() - avalia a posição do elemento em questão.

[illegible]

Esta expressão seleciona o primeiro filho do elemento cd do nó atual.

Este exemplo e os demais estão relacionados com o exemplo de documento.xml apresentado na Figura 4-6.

[illegible]

Abreviado:

```
/catalog/cd/price[price>9,90]
```

Este exemplo seleciona todos os nós preço com um preço maior que 9,90.

Como resultado este exemplo vai retornar o seguinte:

<price> 10,90 </price>

```
<?xml version="1.0" encoding="ISO-8859-
1"?>
<catalog>
<cd country="USA">
<title>Empire Burlesque</title>
<artist>Bob Dylan</artist>
<price>10.90</price>
</cd>

<cd country="UK">
<title>Hide your heart</title>
<artist>Bonnie Tyler</artist>
<price>9.90</price>
</cd>

<cd country="USA">
<title>Greatest Hits</title>
<artist>Dolly Parton</artist>
<price>9.90</price>
</cd>
</catalog>
```

Figura 4-6 Documento XML

4.4.2. Identificador de Fragmento XPointer

Existem três maneiras de especificar os identificadores de fragmentos em XPointer. Uma delas é a especificação completa, conhecida também como Full XPointer, que é bem complexa e permite uma grande flexibilidade ao apontar um documento XML. Ela se baseia na recomendação XPath do W3C.

As duas outras maneiras através dos quais os identificadores de fragmentos podem ser especificados em XPointer são chamados de Identificação de Fragmentos de Nomes Reduzidos (Bare Names) e Identificação de Fragmentos de Seqüências de Filho (Child Sequences).

- **Identificação de Fragmentos por Notação Completa (Full XPointer)**

É uma forma completa de endereçamento que consiste de uma ou mais partes de XPointers. Se baseia na recomendação XPath, utilizando o conceito de etapa de localização. Uma etapa de localização representa a construção que se usará na maioria das vezes quando se desejar criar um XPointer. Fornece um meio de selecionar nós de um documento XML e opera em relação ao nó de contexto, que é simplesmente o nó atual no documento XML quando a etapa de localização está sendo avaliada. Porém, se um nó não for especificado de alguma maneira, o nó atual será o elemento raiz do documento. E se existir mais de uma etapa de localização num XPointer, pode haver mais do que um nó atual sendo avaliado, construindo assim o conceito de caminho de localização, que consiste de um ou mais etapas de localizações separadas por "/".

Assim como em XPath, uma etapa de localização é composta de três partes: eixos, testes de nós e predicado, sendo esse último opcional.

Um Identificador de Fragmento por Notação Completa consiste em especificar todos os limites do fragmento XML, segundo a ordem posicional da esquerda para a direita. Como mostra o exemplo abaixo e referenciado na Figura 4-7.

Ex: `http://www.algumaURL.com/remotoDoc.xml#xpointer(//AAA/BBB[1])`

```
<AAA>
  <BBB myid="b1" bbb="111">
    Text in the first element BBB.</BBB>
  <BBB myid="b2" bbb="222">
    Text in another element BBB.
    <DDD ddd="999">
       Text in more nested element.</DDD>
    <DDD ddd="888">
       Text in more nested element.</DDD>
   <DDD ddd="777">
       Text in more nested element.</DDD>
  </BBB>
  <CCC ccc="123" xxx="321">
    Again some text in some element.</CCC>
</AAA>
```

Figura 4-7 Exemplo de Documento XML sendo referenciado por Notação XPointer completa

Identificação de Fragmentos de Nomes Reduzidos (Bare Names)

Para fornecer uma funcionalidade semelhante aos ponteiros HTML, uma notação é fornecida para elementos com IDs específicos[Martin 2001].

Se o fragmento for simplesmente um valor de ID, o ponteiro aponta para o elemento com aquele valor de ID. Mas para que isto funcione, o documento que está sendo apontado deve ter um esquema que especifique o atributo ID para o elemento[Martin 2001].

A notação por nome reduzido do exemplo mostrado na Figura 4-5 seria:

```
http://www.algumaURL.com/remotoDoc.xml#xpointer(id (“b1”))
```

Ou poderá ser escrito também da seguinte forma:

```
http://www.algumaURL.com/remotoDoc.xml#b1
```

Observe-se que esse exemplo é bem parecido com a âncora de HTML

Estes exemplos selecionam um nó com um id **b1** que é encontrado no documento remoto nomeado como Doc.xml.

Identificação de Fragmentos de Seqüência de Filho (Child Sequences)

O identificador de fragmento de Seqüência de Filho permite que um documento seja apontado ao passar pela árvore de elementos filho.

Usando Sequência de Filho XPointer permite ter acesso a nós fornecendo números em uma seqüência de elementos filho como mostrado abaixo:

```
http://www.algumaURL.com/remotoDoc.xml#xpointer/1/1
```

O exemplo acima quer dizer que é para o XPointer ir para o documento remoto Doc.xml e acessar o primeiro elemento encontrado lá. Depois de encontrar este elemento, achar o seu primeiro elemento filho e depois selecionar o quarto filho deste elemento.

4.5. Acréscimos de XPointer para o XPath

Será visto agora como o XPointer estende o XPath:

- XPointer acrescenta tipos de locais como *pontos* e *intervalos* para o conjunto de nós existentes em XPath. A Tabela 4-4 mostra os tipos de nós do XPath e XPointer [DeRose 2001a].

Tabela 4-4 Tipos de Nós do XPath e XPointer

XPath	XPointer
NodeType ::= 'comment' |'text' |'processing-instruction' |'node'	NodeType ::= 'comment' |'text' |'processing-instruction' |'node' |'point' |'range'

PONTO

O XPointer define o conceito de uma localização de pontos como distinto de um nó. Enquanto uma localização de pontos pode ser um nó, ela também pode ser uma determinada localização dentro do conteúdo de caracter. O ponto indica um posição precedente ou anterior a um nó ou a um caracter [Harold 2001a].

Ex.: <GREETING>Hello</GREETING>

Neste exemplo existem exatamente 3 nós e 14 diferentes pontos no documento. Os nós são o nó raiz que contém o nó de elemento GREETING, que contém um nó de texto. Em ordem os pontos são [Harold 2001a]:

- O ponto antes do nó raiz;
- O ponto antes do nó de elemento GREETING;
- O ponto antes do nó de texto contendo o texto “Hello” (como também espaço em branco);
- O ponto antes do espaço em branco entre <GREETING> e Hello;
- O ponto antes do **H** em Hello;
- O ponto entre o **H** e o **e** em Hello;

- O ponto entre o **e** e o primeiro **l** em Hello;
- O ponto entre o primeiro **l** e o segundo **l** em Hello;
- O ponto entre o segundo **l** e o **o** em Hello;
- O ponto depois do **o** em Hello;
- O ponto depois do espaço em branco entre Hello e </GREETING>;
- O ponto depois do nó de texto nó contendo o texto “Hello”;
- O ponto depois do elemento GREETING;
- O ponto depois do nó raiz.

INTERVALO

XPointer também define o conceito de uma localização de intervalos, que é especificada por dois pontos: um ponto de começo e um ponto de fim. [Harold 2001a].

A habilidade de especificar intervalos permitiria, por exemplo, que um ponteiro apontasse para todas as ocorrências de uma determinada palavra em um documento alvo.

Para especificar um intervalo, anexa-se */range-to(end-point)* para um caminho de localização especificando o ponto inicial do intervalo. Os parênteses contêm um caminho de localização especificando o ponto final do intervalo. A Figura 4-8 mostra um exemplo de XPointer usando a função */range-to (end-point).*

```
xpointer(/child::FAMILYTREE/child::PERSON[position()= 1]
 /range-to(/child::FAMILYTREE/child::PERSON[position()=last()]))
```

Figura 4-8 Exemplo de XPointer com funções

Este exemplo seleciona tudo entre o primeiro elemento PERSON da *tag* inicial e o último elemento PERSON da *tag* final.

- XPointer acrescenta diversas funções novas, como mostra a Tabela 4-5 [Wilde 2002]:

Tabela 4-5 Visão geral das funções XPointer

Nome da Função	Tipo de resultado	Argumentos
end-point	location-set	location-set
here	location-set	n/a
origin	location-set	n/a
range	location-set	location-set
range-inside	location-set	location-set
range-to	location-set	location-set
start-point	location-set	location-set
string-range	location-set	location-set, string, number?, number?

4.6. Erros XPointer

Erros em XPointer podem ser controlados diferentemente dos erros em XPath. Em situações onde uma expressão XPath não acha nenhum local para verificar a identidade entre os detalhes dos dados dentro dos documentos XML, um conjunto de nós vazios são retornados e nenhuma condição de erro é ativada. Porém XPointer é mais rígido com erro que trata porque permite acesso para documentos remotos.

Existem três erros que podem ser provocados por um XPointer inadequado. Qualquer analisador que seja compatível com XPointer precisará tratar destes erros de alguma maneira [Martin 2001]. Na Tabela 4-6 estão apresentados os erros.

Tabela 4-6 Possíveis erros associados com XPointer

Erro	Descrição
Erro de Sintaxe	Este erro ocorre quando uma expressão de XPointer contém uma sintaxe incorreta.
Erro de Recurso	Este erro ocorre quando uma expressão de XPointer está sintaticamente correta, mas está anexado a um recurso inadequado (como um documento que não seja um documento XML).
Erro de Sub-Recurso	Este erro ocorre quando uma expressão XPointer está sintaticamente correta e está anexado a um documento XML bem-formado, mas não resulta em qualquer localização válida.

4.7. Exemplo de XLink com o uso do XPointer

O exemplo da Figura 4-9 ilustra um documento XML. E a Figura 4-10 apresenta o código de XLink usando XPointer com o identificador de fragmento de Seqüência Filho.

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<doc>
      <titulo>LINGUAGEM XPOINTER </titulo>
      <subtitulo>Neste capitulo será apresentado uma visão geral sobre XPointer,
               descrevendo conceitos, princípios do projeto, origem e uso do
               XPointer.</subtitulo>
      <texto>
            <texto1> Child Sequences </texto1>
            <texto2> Bare Names </texto2>
            <texto3> Full Names </texto3>
      </texto>
</doc>
```

Figura 4-9 Exemplo de um documento XML sendo referenciado por um Identificador de Fragmento de Sequência Filho

```
<?xml version='1.0' >
<doc
   xmlns:xlink="http://www.w3.org/1999/xlink">
     <link
         xlink:type="simple"
         xlink:href="cap4-xpointer.xml#xpointer(/1/3/2)"
         xlink:actuate="onRequest">Link</link>
</doc>
```

Figura 4-10 Código de um exemplo de XLink com XPointer

4.8. Considerações Finais

Neste capitulo foi dada uma visão geral sobre a tecnologia XPointer, uma vez que a mesma acrescenta um significado importante ao mecanismo XLink fornecendo um grau de precisão de links que é necessário para os documentos robustos e descritos pela linguagem XML.

Capítulo 5 IXLink: Projeto e Implementação

Este capítulo descreve o projeto e a implementação do IXLink (Interpretador de XLink), o qual foi desenvolvido baseado na especificação de XLink da W3C e apresenta o funcionamento dos links XML.

5.1. Introdução

IXLink é um interpretador capaz de processar a sintaxe XLink com alguns dos seus respectivos atributos e valores, baseado na especificação de XLink. Foi desenvolvido com o intuito de simular o funcionamento dos links XML no browser Internet Explorer.

Tendo em vista a questão do prazo limitado para o seu desenvolvimento, a linguagem escolhida para implementação foi JavaScript. Esta é uma linguagem compacta, simples, baseada em orientação a objetos, destinada para o desenvolvimento de aplicações cliente/servidor na Internet e tem como vantagem a possibilidade de simular várias funções do browser [Holzner 2001].

Mesmo sendo JavaScript uma linguagem limitada, através dela foi possível atingir o nosso objetivo: simular o funcionamento do XLink utilizando o Internet Explorer.

5.2. Características do IXLink

Uma ferramenta que é capaz de processar o XLink deve reunir uma série de características descritas na especificação. Características essas que envolvem a criação do comportamento do link tomando-se como base o tipo de link, o recurso ao qual ele está se conectando, as circunstâncias com as quais o usuário irá se deparar no link, quais opções se quer fornecer ao usuário quando o link for percorrido, entre outras.

Vários atributos e valores são fornecidos com o intuito de tornar os links XML em links criativos, flexíveis e poderosos, links capazes de superar a limitação dos links HTML, permitindo uma maior funcionalidade e aplicabilidade aos relacionamentos entre recursos [St. Laurent 2002].

O IXLink traz suporte à maioria das características do XLink, incluindo endereçamento a uma região do documento com a utilização do XPointer. Mais precisamente, ele interpreta os links simples, estendidos e parte do XPointer.

Pode ser usado para qualquer tipo de documento XML que contenha XLink e só pode ser executado no browser Internet Explorer. Através do IXLink pode-se analisar o funcionamento dos links em documentos XML, de forma que se possa entender melhor o que está definido na especificação.

O código do IXLink pode ser facilmente adaptado para a utilização em aplicações específicas que necessitem do uso de links XML e traz suporte aos seguintes atributos:

- *href*, o qual tem como valor o URI, que identifica o recurso destino;
- *type*, onde seu valor deve ser fornecido e, conforme a especificação de XLink, pode ser: “simple”, “extended”, “locator”, “arc”, ou “resource”;
- *show*, onde seu valor pode ser ou não fornecido. Caso seja fornecido deve ser um dos valores: “new”, “replace” ou “embed”;
- *actuate*, onde seu valor pode ser: “onRequest” ou “onLoad”, sendo um atributo também opcional;
- *title*, onde seu valor é uma descrição do significado do link;
- Identificador de Fragmento de Sequência de Filho (Child Sequence), usando XPointer.

Com suporte a esses atributos podemos constatar várias características e aplicabilidades mencionadas no capitulo 3, seções 3.4.1 e 3.4.2 desta dissertação.

Na Seção 5.3 será mostrado como esses atributos se comportam no IXLink.

Nem todos os atributos citados anteriormente são suportados pelo browser Netscape, como por exemplo os atributos referentes a links estendidos e o atributo show= “embed”. O browser Internet Explorer não suporta a especificação XLink. Entretanto, através da ferramenta IXLink poderá interpretar os atributos citados.

5.3. Funcionamento do XLink usando IXLink

Com base na especificação, será apresentado o funcionamento de alguns atributos e valores do XLink usando o IXLink. Conforme o exemplo do código da Figura 5-1, o IXLink

interpreta os atributos *href* e *type* como o identificador do recurso de destino e identificador do tipo de link respectivamente.

O atributo *title* fornece o significado de um link quando se posiciona o ponteiro do mouse no ícone [?], como mostra a Figura 5-2.

```
<SITE
          xmlns:xlink="http://www.w3.org/1999/xlink"
          xlink:type = "simple"
          xlink:href="http://www.w3c.org"
          xlink:title="SITE W3C">SITE OFICIAL
</SITE>
```

Figura 5-1 Exemplo do código de XLink simples

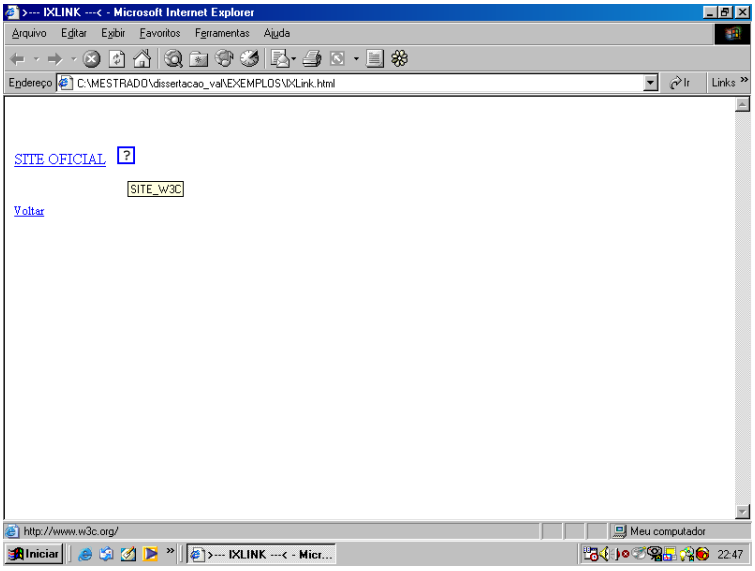


Figura 5-2 Exemplo do código de elemento<SITE> exibido no IXLink

Porém, em outra ferramenta este atributo poderia ter sido implementado de outra forma, como por exemplo, sua função poderia ser exibida ao posicionar o ponteiro do mouse sobre a referência de link, como ocorre no Netscape, ou poderia também ser exibido na barra de status do próprio browser, pois o funcionamento deste atributo fica a critério do desenvolvedor [DeRose 2001]. Quando os atributos de comportamento não forem utilizados, o que prevalece no IXLink é o comportamento *default*, isto é, o link vai ser ativado através de uma ação do usuário e o conteúdo do recurso destino substitui a janela de origem.

Um outro código de link simples usando o atributo *show = "embed"* pode ser visto na Figura 5-3. Nesse código, o valor do *xlink:href* corresponde a uma imagem. Utiliza o atributo *show= "embed"* para embutir a imagem na janela atual. Com o uso do atributo *actuate=*

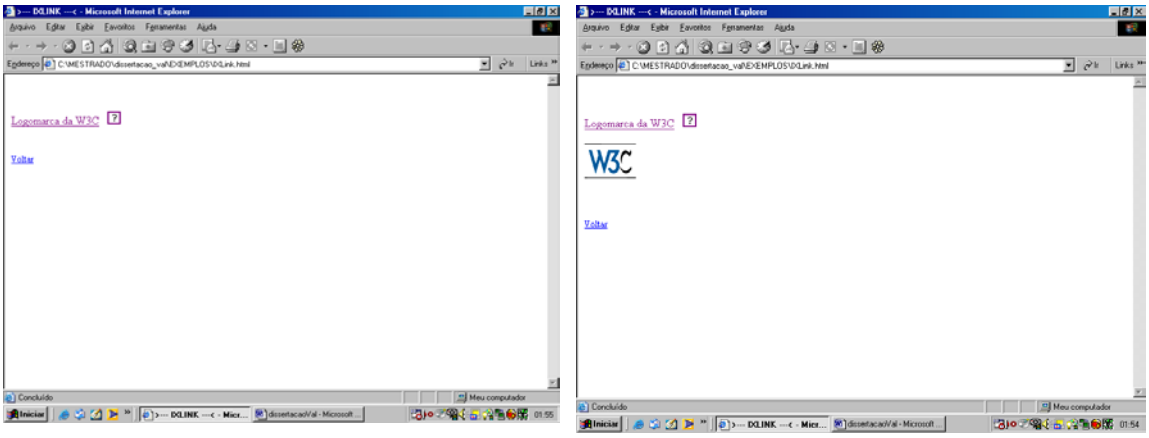
"onRequest", reforça a idéia de que essa imagem só vai ser embutida após uma ação do usuário. Caso fosse utilizado o atributo "*onLoad*", a imagem seria exibida automaticamente após o link ser carregado.

```
<IMAGEM
          xmlns:xlink="http://www.w3.org/1999/xlink"
          xlink:type = "simple"
          xlink:href="http://www.w3.org/w3c_home.gif"
          xlink:actuate= "onRequest"
          xlink:show= "embed"
          xlink:title="imagem embutida">Logomarca da W3C
</IMAGEM>
```

Figura 5-3 Exemplo de um código de link simples com o atributo show= "embed"

No browser Netscape não existe suporte ao atributo *show= "embed"*. Portanto o valor *"embed"* só funciona no IXLink. É similar ao efeito conseguido pelo seguinte fragmento HTML:

`<IMG SRC= http://www.exemplo.org/smiley.gif ALT = ":)"`[DeRose 2001]. Porém no IXLink esse valor funciona tanto para imagem quanto para texto, o que difere da HTML, que só funciona para imagem. A Figura 5-4 mostra este exemplo de código exibido no IXLink.



A B

Figura 5-4 Exemplo do código <IMAGEM> exibido no IXLink

A Figura 5-4A representa um link a espera de um "clique" do usuário e a Figura 5-4B representa o resultado após o "clique" do usuário.

A Figura 5-5 apresenta um código de link estendido com múltiplos destinos [Harold 2001]. Esse código corresponde a um link estendido contendo um recurso local e quatro remotos.

Representa um relacionamento entre um site principal e sites espelhos, como mostra a Figura 5.6.

```
<WEBSITE         xmlns:xlink="http://www.w3.org/1999/xlink"
                  xlink:type="extended"
                  xlink:title="Cafe au Lait">

<NAME
xlink:type="resource"
xlink:label="source">Cafe au Lait  </NAME>

<HOMESITE
xlink:type="locator"
xlink:href="http://ibiblio.org/javafaq/"
xlink:label="home"/>

<MIRROR
xlink:type="locator"
xlink:href="http://sunsite.kth.se/javafaq"
xlink:label="mirror"
xlink:title="Site espelho Sueco"/>

<MIRROR
xlink:type="locator"
xlink:href="http://sunsite.informatik.rwth-aachen.de/javafaq/"
xlink:label="mirror"
xlink:title="Site espelho Alemão"/>

<MIRROR
xlink:type="locator"
xlink:href="http://sunsite.cnlab-switch.ch/javafaq/"
xlink:label="mirror"
xlink:title="Site espelho suiço"/>

<CONNECTION
xlink:type="arc"
xlink:from="source"       xlink:to="home"
xlink:show="replace"
xlink:actuate="onRequest"/>

<CONNECTION
xlink:type="arc"
xlink:from="source"       xlink:to= “mirror"
xlink:show="replace"
xlink:actuate="onRequest"/>

</WEBSITE>
```

Figura 5-5 Exemplo de um código de XLink estendido com múltiplos destinos

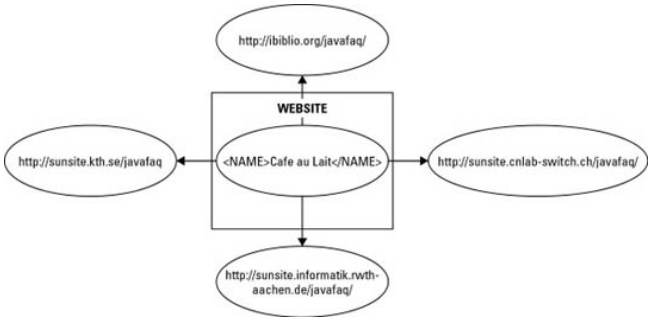


Figura 5-6 Um link estendido com um recurso local e quatro recursos remotos

O IXLink interpreta esse código como link estendido através do atributo *type= "extended"* De acordo com a especificação ele interpretará também todos os subelementos do elemento *extended*, como por exemplo, os subelementos *resource*, *locator* e *arc*. A conexão entre os recursos se dá através do elemento *arc* com os atributos *from* e *to* que identificam o recurso origem e o recurso destino respectivamente, usando o valor do atributo *label.* Como os links simples, o elemento *arc* usa os atributos *xlink:show* e *xlink:actuate* para definir o comportamento de um link.

No IXLink, o link ao ser carregado exibe o recurso local, o qual o usuário deverá dar um "clique" sobre o ícone  (expandir) para estender os recursos remotos, conforme mostra as Figuras 5.7 e 5.8.

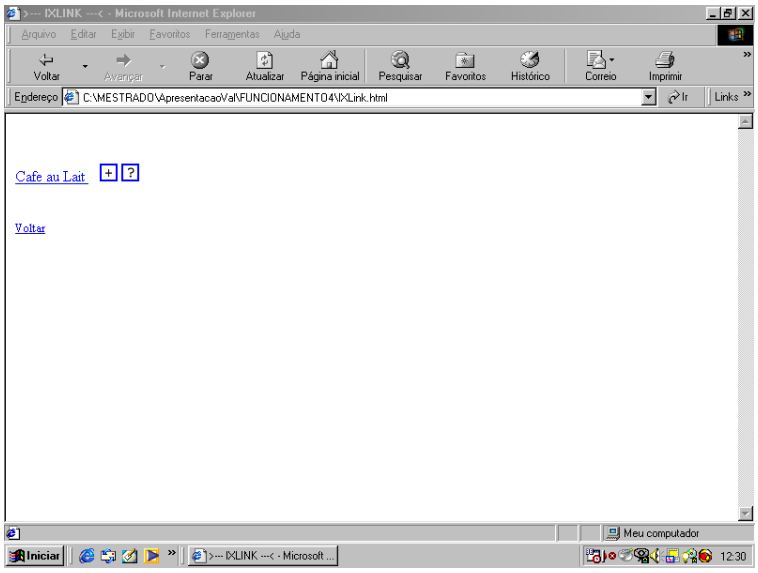


Figura 5-7 Exemplo do código de elemento <WEBSITE> executado no IXLink.

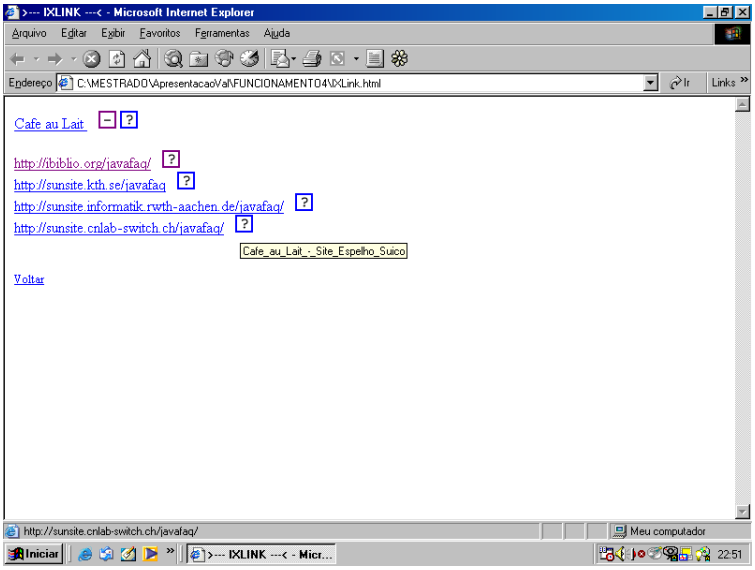


Figura 5-8 Exemplo do código de elemento <WEBSITE> executado no IXLink, após ação do usuário.

Após clicar no ícone [+] surgirá um menu de onde o usuário poderá escolher um destino diferente, conforme mostra a Figura 5.8.

Um outro código analisado foi com links entre recursos diferentes, sendo um documento do tipo doc, documento do tipo XML e páginas Web. Este código é apresentado na Figura 5.9 e mostra o relacionamento entre um recurso local e cinco recursos remotos, que são três sites Web, um documento tipo Word e um documento xml.

De acordo com o código, os três sites que correspondem ao conteúdo W3C, ZVON.ORG e XML.ORG serão exibidos automaticamente, assim que o usuário clicar no ícone [+], pois o atributo comportamental dos links é o *actuate= "onLoad",* enquanto os demais links ficarão à espera de uma ação do usuário, como mostra as Figuras 5.10 e 5.11.

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<LINKS
          xmlns:xlink="http://www.w3.org/1999/xlink"
          xlink:type="extended"
          xlink:title= "Clique no ícone + para surgir o menu de links">
    <name
           xlink:type="resource"
           xlink:label="local">LINKS XML </name>

    <link
           xlink:type="locator"
           xlink:href="http://www.w3c.org"
           xlink:label="site"
```

```
             xlink:title="Site oficial XML">W3C</link>

     <link
             xlink:type="locator"
             xlink:href="http://www.zvon.org"
             xlink:label="site"
             xlink:title="Site ZVON">ZVON.ORG</link>

     <link
             xlink:type="locator"
             xlink:href="www.xml.org"
             xlink:label="site"
             xlink:title="Site XML">XMLORG</link>

     <link
           xlink:type="locator"
           xlink:href="capitulo2.doc"
           xlink:label="doc"
           xlink:title="Dissertaçao de Valéria Rosa">Capitulo 2 XML</link>

    <link
           xlink:type="locator"
           xlink:href="dissertacao.xml"
           xlink:label="embutido"
           xlink:title="Texto embutido"> Para mais informação: Clique
Aqui</link>

<CONECTION  xlink:type="arc"    xlink:from="local"   xlink:to="site"
            xlink:actuate= "onLoad">
</CONECTION>

<CONECTION  xlink:type="arc"    xlink:from="local"   xlink:to="doc"
            xlink:actuate= "onRequest"     xlink:show= “new”>
</CONECTION>

  <CONECTION  xlink:type="arc"    xlink:from="local"   xlink:to="embutido"

           xlink:show="embed">

</CONECTION>

</LINKS>
```

Figura 5-9 Exemplo de um código de XLink estendido com links entre recursos diferentes.

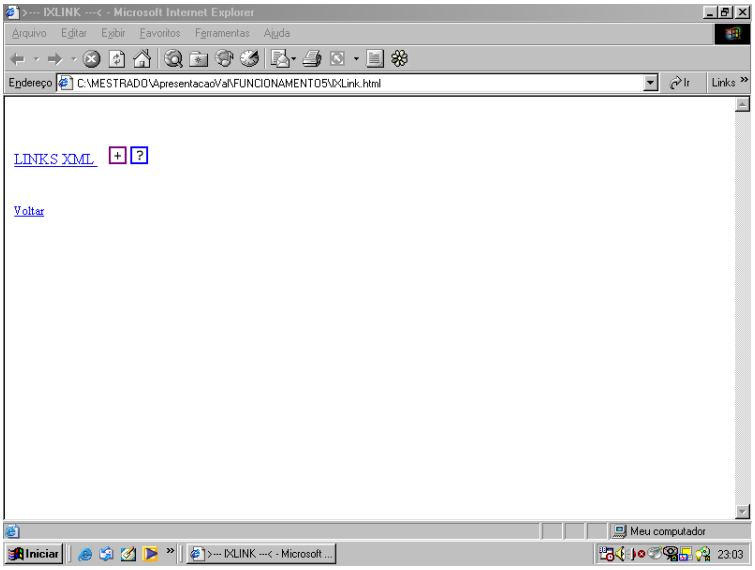


Figura 5-10 Exemplo do código de elemento <LINKS> após link carregado.

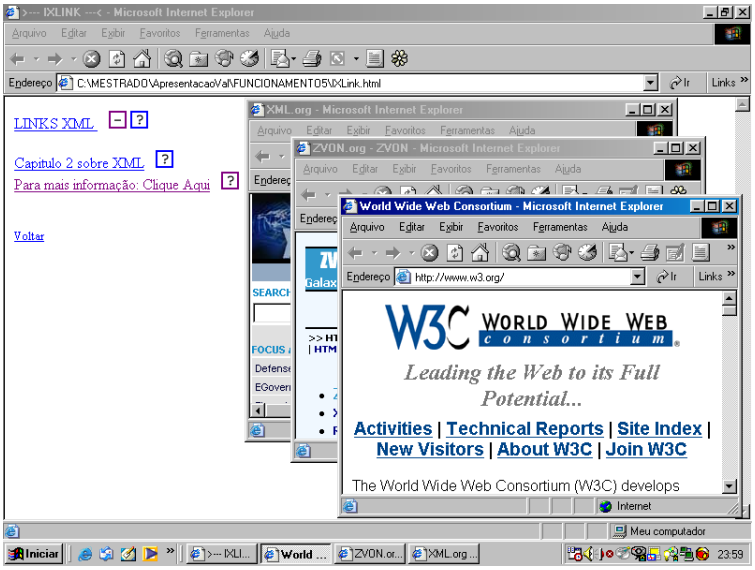


Figura 5-11 Exemplo do código de elemento <LINKS>, após um clique do usuário sobre o ícone expandir.

O documento do tipo word, vai ser exibido em uma nova janela, abrindo o software de origem, porém uma outra ferramenta poderia interpretar documentos não xml, solicitando ao usuário um download. No IXLink achou-se que isso não seria necessário porque no próprio browser Internet Explorer existe a opção de download: basta o usuário clicar com o botão direito sobre a referência de link e um submenu surgirá com essa opção. Por isso achou-se mais eficiente implementar documentos não XML exibindo no software de origem. Com relação ao documento XML, devido ao uso do atributo *show= "embed",* ele será embutido no

documento origem, após o usuário clicar no conteúdo *Para mais informação: Clique Aqui* conforme a Figura 5-12.

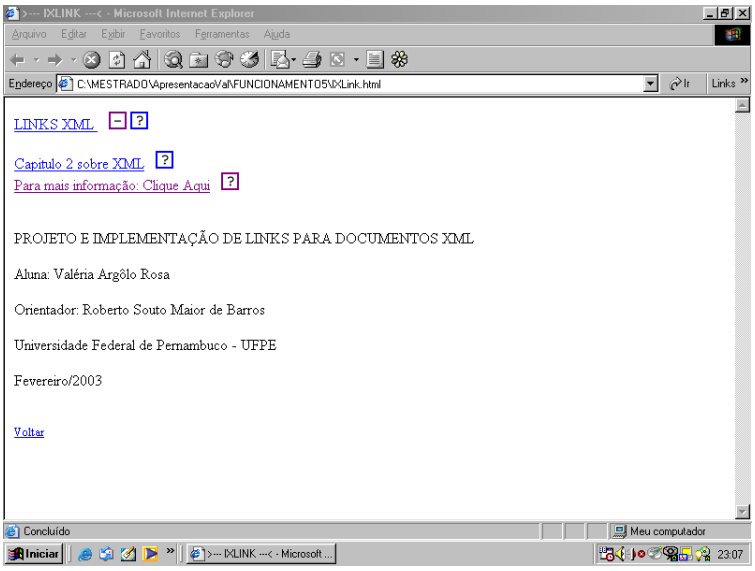


Figura 5-12 Exemplo do código de elemento <LINKS> com o texto embutido.

Usando XPointer juntamente com XLink pode-se endereçar qualquer parte de um documento XML. O IXLink traz um suporte parcial de XPointer, onde utiliza-se do modo de sequência filho (Child Sequence) para selecionar um fragmento do documento XML. Um bom exemplo para ilustrar como esse tipo de endereçamento funciona, é um catálogo de CDs, onde encontra-se o titulo do CD, o nome do Artista e o preço do CD. Usando XLink com XPointer pode-se fazer uma consulta nesse catálogo, tendo como retorno apenas o que foi especificado no código. Na Figura 5.13 é apresentado o código de um exemplo de consultas de CDs, o qual é chamado de consulta.xml e na Figura 5.14 é apresentado o código do arquivo do catálogo de CDs, catalogo.xml.

De acordo com o código de consulta, após o usuário clicar no link, será retornado para ele, o preço do CD, substituindo a janela atual. No terceiro link, por utilizar o atributo *show= "embed",* o preço será embutido na janela atual.

As Figuras 5-15, 5-16A e 5-16B mostram como esse código é exibido no IXLink, que interpreta o modelo Sequência de Filho conforme a especificação. A Figura 5-16A mostra o exemplo do código após um "clique" do usuário no 1º link e a Figura 5-16B mostra o exemplo do código após um "clique" do usuário no 3ª link.

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<consulta>
     <TITULO> CONSULTA PREÇO </TITULO>
     <cd
            xmlns:xlink="http://www.w3.org/1999/xlink"
            xlink:type = "simple"
            xlink:href="catalogo.xml#xpointer(/1/2/3)"
            xlink:actuate="onRequest"
          xlink:show= "replace"
            xlink:title="PREÇO DE CD">JORGE VERSILO - LEVE
     </cd>
     <cd
            xmlns:xlink="http://www.w3.org/1999/xlink"
            xlink:type = "simple"
            xlink:href="catalogo.xml#xpointer(/1/3/3)"
            xlink:actuate="onRequest"
            xlink:show= "replace"
            xlink:title="PREÇO   DE   CD">Marisa   M./Carlinhos   B./Arnaldo   A.   -
TRIBALISTAS
     </cd>
     <cd
            xmlns:xlink="http://www.w3.org/1999/xlink"
            xlink:type = "simple"
            xlink:href="catalogo.xml#xpointer(/1/4/3)"
            xlink:actuate="onRequest"
            xlink:show= "embed"
            xlink:title="PREÇO DE CD"> Ivete Sangalo - FESTA
     </cd>
</consulta>
```

Figura 5-13 Exemplo de código do arquivo consulta.xml. Uso de XLink com XPointer.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalogo>
  <titulo> CATALOGO DE CDS </titulo>
  <cd1>
    <titulo> Titulo:Leve</titulo>
    <artista>Artista:Jorge Versilo</artista>
    <preco>Preço:24,00</preco>
  </cd1>
  <cd2>
    <titulo>Titulo:Os Tribalistas</titulo>
    <artista>Artista:Marisa Monte, Carlinhos Brows e Arnaldo Antunes</artista>
    <preco>Preço:29,50</preco>
  </cd2>
  <cd3>
    <titulo>Titulo:Se eu não te amasse tanto assim</titulo>
    <artista>Artista:Ivete Sangalo</artista>
    <preco>Preço:20,90</preco>
  </cd3>
</catalogo>
```

Figura 5-14 Exemplo do código do arquivo catalogo.xml

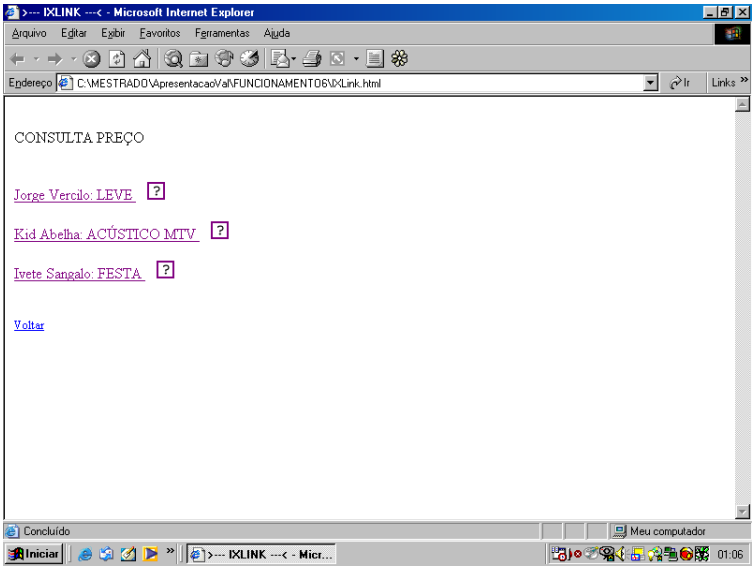
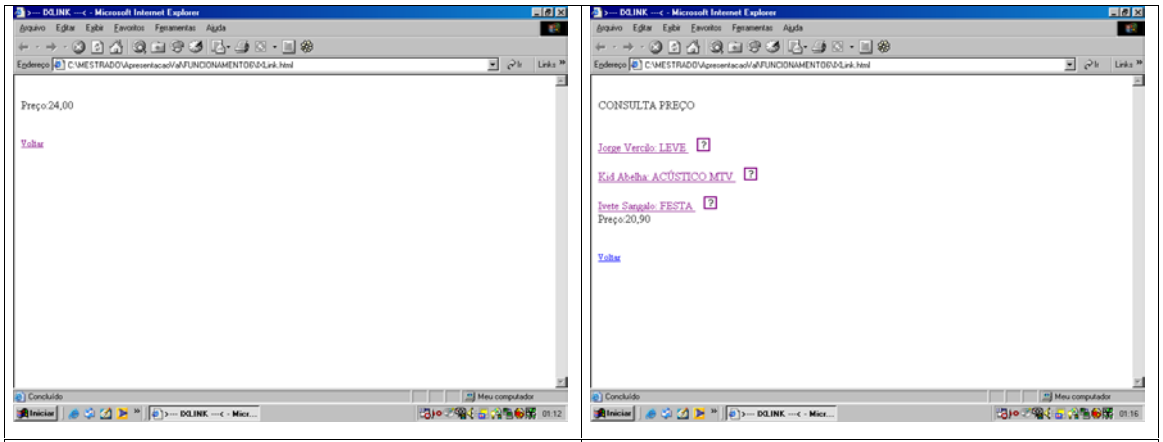


Figura 5-15 Exemplo do código do arquivo consulta.xml executado no IXLink



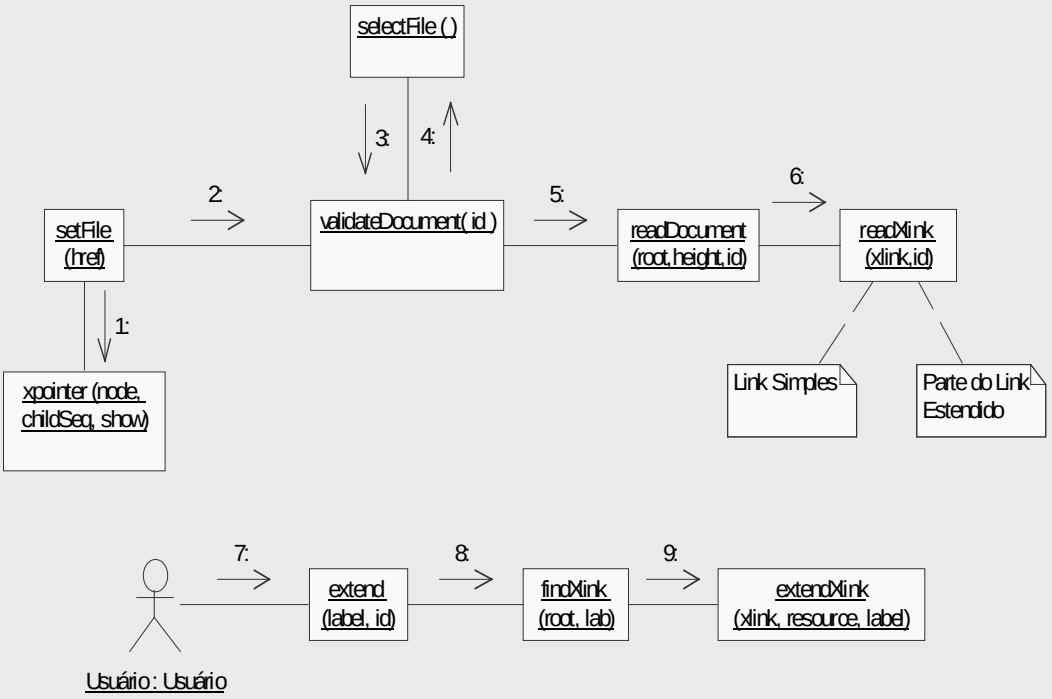
A	B

Figura 5-16 Exemplo do código após um "clique" do usuário

Esses foram alguns exemplos de código usados para verificar o funcionamento no IXLink, porém fica a critério do desenvolvedor em criar links XML, usando a criatividade de acordo com a necessidade da aplicação em uso.

5.4. A estrutura do IXLink

A estrutura do IXLink é apresentada na Figura 5.17. O interpretador lê o arquivo documento XML que contém o XLink, recupera os seus atributos e interpreta segundo a especificação.



1: A função setFile (href) chama a função xpointer (node, childSeq,show);
2: A função setFile (href) chama a função validateDocument(id);
3: A função selectFile ( ) chama a função validateDocument(id);
4: Caso ocorra erro no item 3, o item 4 será executado, caso contrário, o item 5;
5: A função readDocument (root, height, id) chama a função readXlink (xlink, id);
6: A função readXlink (xlink, id) interpreta os links simples e parte dos links estendidos;
7: O usuário chama a função extend (label, id);
8: A função extend (label, id) chama a função findXlink(root, lab);
9: A função findXlink (root, lab) chama a função extendXlink (xlink, resource, label).

Figura 5-17 Estrutura do Interpretador IXLink

Para ler os documentos XML foi utilizado a API DOM (Document Object Model) através do parser MSXML embutido no browser Internet Explorer. Para um melhor entendimento dessa estrutura subseções descrevem cada função definida no interpretador.

5.4.1. Carregando um documento XML contendo XLink

Existem duas maneiras de carregar um documento XML contendo XLink. Através das funções definidas como *selectFile() e setFile(href).*

- Função *selectFile()*

É chamada ao carregar o IXLink. Seleciona o arquivo XML através de um prompt que chama a função *validateDocument*.

A Figura 5.18 apresenta o código da função *selectFile()* e a Figura 5.19 apresenta o resultado da execução desse código no browser Internet Explorer.

```
function selectFile ()
{
     file = prompt("Digite o nome do Arquivo XML:")
     filesDIV.innerHTML = "<XML ID=\"xlink\" SRC=\"" + file + "\"></XML>"
     validateDocument("xlink")
}
```

Figura 5-18 Código da função selecFile()

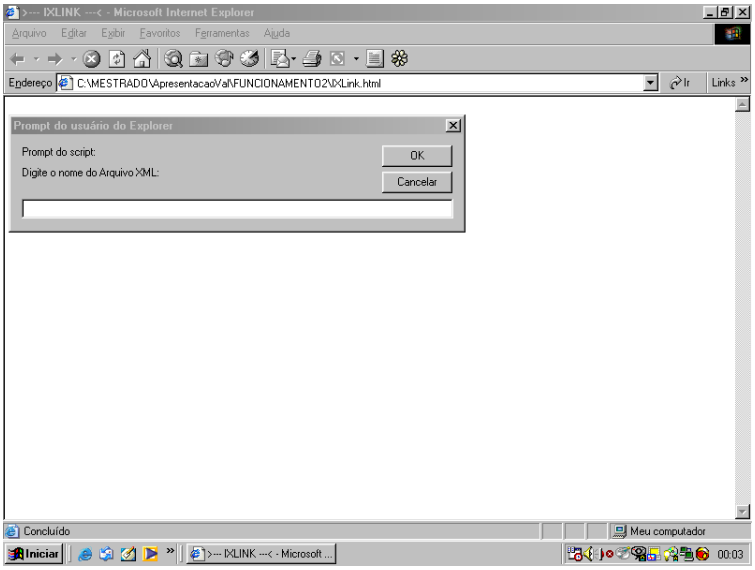


Figura 5-19 Função selectFile executada no Internet Explorer

- Função *setFile(href)*

Configura o arquivo XML padrão (“xlink”)[5]. Recebe como parâmetro um endereço. Essa

[5] “xlink” é definido como o id do documento selecionado.

função permite carregar o documento XML contendo XLink de forma automática, sem precisar solicitar ao usuário o nome do documento. Pois o documento a ser verificado já vai estar definido no cabeçalho do código do interpretador da seguinte forma:

```
<BODY onload= “setFile( ‘nome_documento.xml’)”>
```

A Figura 5.20 apresenta o código da função *setFile(href).* O código foi dividido em 4 blocos para melhor compreensão.

```
function setFile(href) {

       1      ixp = href.indexOf("#xpointer")
[1]    2      start = filesDIV.innerHTML.indexOf("SRC=") + 5
       3      end = filesDIV.innerHTML.indexOf("\">")
       4      oldFile = filesDIV.innerHTML.substring(start, end)
-------------------------------------------------------------------------------
       6      if (ixp <0) {
[2]    7             filesDIV.innerHTML = "<XML ID=\"xlink\" SRC=\"" + href + "\"></XML>"
       8             validateDocument("xlink")
       9             filesDIV.innerHTML += "<br><br><font size = -1><a
      10              href=javascript:setFile(\"" + oldFile + "\")>Voltar</a></font>"
-------------------------------------------------------------------------------
      12     } else{
      13             xml = href.substring(0,ixp)
      14             childSeq = href.substring(ixp + 12, href.length-1 )
[3]   15             filesDIV.innerHTML = "<XML ID=\"xpointer\" SRC=\"" + xml + "\"></XML>"
      16             xmldoc = document.all("xpointer").XMLDocument
      17             if(xmldoc.parseError == false) {
      18             xpointer(xmldoc.documentElement, childSeq, "replace")
      19             filesDIV.innerHTML += "<br><br><font size = -1><a
      20               href=javascript:setFile(\"" + oldFile + "\")>Voltar</a></font>"
-------------------------------------------------------------------------------
[4]   22             } else if(confirm("Erro! Motivo: " + xmldoc.parseError.reason+ " Linha:
      23                                " + xmldoc.parseError.line +
      24             " Texto: " + xmldoc.parseError.srcText + "\n \nVocê deseja selecionar
      25             outro documento?")){
      26     selectFile()
                     }

              }

       }
```

Figura 5-20 Código da função setFile(href)

O primeiro bloco trata da declaração de variáveis. O segundo bloco do código interpreta o documento XML que possui apenas o XLink. Ao ser verificada a inexistência da string #xpointer, a função *setFile (href)* chama a função *validateDocument(xlink).* Essa função será comentada na Seção 5.4.2. O comando representado nas linhas 9 e 10 permite que, entre ligações de documentos XML, o usuário tenha a opção de ***voltar*** para o endereço percorrido anteriormente, dando a idéia de links bidirecionais.

O terceiro bloco é responsável por tratar documentos XML contendo XLink e XPointer, verificando se a sintaxe está correta. Na linha 13 é localizada a substring que identifica o XPointer. Em seguida é extraído dessa substring a sequência de filho. Logo após a sintaxe do

documento é avaliada. Se não houver erros, é chamada a função xpointer tendo como parâmetros o elemento raiz, a sequência de filho e o atributo show= “replace”. As linhas 19 e 20 têm o mesmo objetivo das linhas 9 e 10.

O último bloco trata dos erros de sintaxe. Os erros são apresentados e é solicitado um outro documento com o qual será chamada a função selectFile().

5.4.2. Validando Documentos

A função *validateDocument(id)* foi definida para validar o documento. Esta função recebe como parâmetro o id do documento XML e o valida. A Figura 5.21 apresenta o código da função *validateDocument(id).*

```
function validateDocument(id) {

       1       if (navigator.appName!= "Microsoft Internet Explorer") {
       2               alert("O IXlink foi desenvolvido para rodar no Microsoft Internet
       3                       Explorer")
       4       }
       5
       6       if (id == "xlink") {
       7                       xmlDIV.innerHTML = ""
       8       } else {
       9                       xmlDIV.innerHTML += "<br>"
      10               }
      11
      12               xmldoc = document.all(id).XMLDocument
      13
      14               if(xmldoc.parseError == false) {
      15                       readDocument(xmldoc.documentElement , 0 , id)
      16               } else if(confirm("Erro! Motivo: " + xmldoc.parseError.reason+ " Linha:
      17               " + xmldoc.parseError.line +  " Texto: " + xmldoc.parseError.srcText "\n
      18                       \nVocê deseja selecionar outro documento?")){
      19               selectFile()
               }
        }
```

Figura 5-21 Código da função validateDocument (id)

A linha 1 determina o tipo de browser usado. Se o IXLink for usado em outro browser será exibido uma caixa de diálogo de alerta com a mensagem apresentada na linha 2. A linha 14 trata da validação do documento. Se não houver erro a função *readDocument* será chamada, passando como parâmetros o elemento raiz do documento validado, sua altura e o id do documento.

Em caso de erro, a linha 16 exibe as informações sobre o erro e a linha 18 oferece a opção ao usuário de selecionar um outro documento, chamando a função *selectFile().*

Qualquer erro de validação terá suporte pelo objeto *parseError,* que mantém informações

sobre o erro de análise mais atual, com as propriedades *reason, line, srcText.*

5.4.3. Reconhecendo o Namespace

Para estar de acordo com a especificação, é necessário que haja o reconhecimento do Namespace. A função *readDocument (root, height, id*), que recebe como parâmetros a raiz da árvore ou subárvore, sua altura em relação a raiz principal e o id do documento, trata do existência do xlink através do mecanismo de Namespace. A Figura 5-22 apresenta o código da função readXlink. O código foi dividido em 3 blocos para melhor compreensão.

```
function readDocument(root, heigth, id) {

        1     isXLink = false
        2     if(root.attributes != null) {
        3                     if(root.attributes.length > 0) {
        4                             xmlns = root.attributes.getNamedItem("xmlns:xlink")
        5                     if (xmlns != null && xmlns.value ==
        6                                             "http://www.w3.org/1999/xlink"){
        7                             readXlink(root , id)
[1]     8                             isXLink = true
        9                     }
       10             }
       11     }
-------------------------------------------------------------------------------------
       13     if(root.nodeValue != null && !isXLink) {
       14             xmlDIV.innerHTML += "<br>"
       15             for (i = 2; i < heigth; i++) {
[2]    16                     xmlDIV.innerHTML += "   "
       17             }
       18             xmlDIV.innerHTML += root.nodeValue +"<br>"
       19     }
-------------------------------------------------------------------------------------------
       21     if (root.childNodes.length > 0 && !isXLink) {
       22             for (var  i = 0;i < root.childNodes.length;i++) {
       23                     temp = root.childNodes.item(i)
       24                     if (temp != null){
[3]    25                             readDocument(temp, heigth +1, id)
       26                     }
       27             }
       28     }
       }
```

Figura 5-22 Código da função readDocument

Esta função lê os nós do documento e chama a função *readXlink* ao encontrar um xlink. Esta função tem capacidade de interpretar três tipos de situações:

O primeiro bloco verifica a existência do XLink recuperando o seu namespace xmlns:xlink com o valor "`http://www.w3.org/1999/xlink`", conforme a linha 6. Se for especificado o namespace e seu valor na configuração do link, a função *readXlink* é chamada (linha 7), passando como parâmetros o elemento raiz e o id do documento.

O segundo bloco verifica a inexistência do XLink num documento XML. Se for apenas um

documento XML (linha 14), vai ser gerado um código HTML e apenas o valor do nó raiz será exibido.

O terceiro bloco verifica a existência de nós filhos no documento XML que não contém XLink (linha 22). Nesse caso a função *readDocument* é chamada recursivamente para todos os seus nós filhos (linha 27).

5.4.4. Interpretando Links Simples e parte dos Estendidos

Através da função *readXLink*, os links simples e parte dos estendidos são interpretados. Esta função lê os atributos de xlink e os interpreta segundo a especificação da W3C. Para isso, a função recebe um nó XML referente a um nó xlink, recupera os atributos de xlink desse nó e identifica o tipo de link, o qual tomará as decisões relacionadas ao tipo identificado. A Figura 5-23 apresenta o trecho do código com interpretação dos links simples. O código foi dividido em 5 blocos para melhor compreensão. O primeiro bloco simplesmente declara as variáveis que serão utilizadas no código.

O segundo bloco refere-se à identificação do tipo de XLink e alguns dos seus atributos. A linha 6 identifica o valor do atributo *type*. Caso o seu valor seja especificado como "*simple*", são verificados os seus atributos na lista de atributos *xlink.attributes* (linha 7). Na linha 8 é recuperado o valor do atributo *href*. Em seguida a linha 9 recupera o atributo opcional *show*. Caso seja especificado o atributo *show*, seu valor é recuperado através da função *validateShow* (linha 11). Se esse atributo não for especificado, será recuperado o default, conforme apresenta a Figura 5-24.

O terceiro bloco avalia o atributo opcional *actuate*. Caso este atributo não seja especificado ou tenha o valor definido como "*onRequest*" (linha 16) é verificado se é um documento XML, um gif ou um jpg. Se no documento XML não for especificado o atributo *show= "embed*" será chamada a função *setFile(href)* (linha 20). Mas se o atributo show= "embed" tiver sido especificado, a função *embed (href)* será chamada (linha 22).

```
function readXlink(xlink , id) {
    1          XLinkAttributes = xlink.attributes
    2  [1]     XLinkType = XLinkAttributes.getNamedItem("xlink:type").value
    3          name = xlink.firstChild.firstChild
    4          nameNode = xlink.firstChild
-------------------------------------------------------------------------------------------
    6  if(XLinkType == "simple") {
    7                 Att = xlink.attributes
    8                 href = Att.getNamedItem("xlink:href").value
    9                 show = Att.getNamedItem("xlink:show")
   10  [2]            if (show != null) {
   11                        target = validateShow(show.value)
   12                } else {
   13                        target = validateShow(" ")
   14                }
-------------------------------------------------------------------------------------------
   15  actuate = Att.getNamedItem("xlink:actuate")
   16                if (actuate == null || actuate.value == "onRequest") {
   17                xml = href.lastIndexOf(".xml")
   18                gif = href.lastIndexOf(".gif")
   18                jpg = href.lastIndexOf(".jpg")
   19  [3]                           if( xml > 0 && target != "embed") {
   20                                href = "javascript:setFile(\"" + href + "\")"
   21                        } else if(xml > 0 || gif > 0 || jpg > 0) {
   22                                href = "javascript:embed(\"" + href + "\")"
   23                                target = validateShow(" ")
   24                        }
   25                         xmlDIV.innerHTML += "<br><br><a href=" + href + " target=" +
   26                        target + ">" + xlink.firstChild.nodeValue + "</a>"
   27
-------------------------------------------------------------------------------------------
   29                if (Att.getNamedItem("xlink:title") != null) {
   30  [4]                           title = Att.getNamedItem("xlink:title").value
   31                                title = validateTitle(title)
   32                                xmlDIV.innerHTML += "   <a href=" + href +
   33                                  " target=" + target + " ><img src=\"title.gif\" alt=" +
   34                                title + "></a>"
   35                }
-------------------------------------------------------------------------------------------
   36         } else if(target != "embed") {
   37                        open(href)
   38                } else {
   39  [5]                   embed(href)
   40                }
   41
   42         }
   43  }
```

Figura 5-23 Código da função readXLink com interpretação aos links simples

```
       function validateShow (show) {
              var target
              switch(show) {
                     case "new":
                            target = "_Blank"
                            break;
                     case "replace":
                            target = "_self"
                            break;
                     case "embed":
                            target = "embed"
                            break;
                     default:
                            target = "_self"
                            break;
              }
              return(target)
       }
```

Figura 5-24 Código da função validateShow

A existência do atribto *title* é verificado no quarto bloco. Caso o atributo tenha sido especificado, é chamada a função *validateTitle*, conforme apresentado na Figura 5-25.

```
function validateTitle(title) {
           for (i = 0; i < title.length; i++) {
                 if (title.charAt(i) == " ") {
                              title = title.substring( 0 , i) + "_" + title.substring((i+1),
title.length)
        }
                                                              }
        return(title)
        }
```

Figura 5-25 Código da função validateTitle (title)

Essa função valida o atributo *title* substituindo os espaços em branco por underline ("_"), contornando a deficiência da HTML que ignora qualquer string após ocorrência do primeiro espaço em branco. Com isso, numa configuração de link usando o atributo *xlink:title= "Site da W3C"*, o IXLink interpreta da seguinte forma: Site_da_W3C.

A função *readXlink* também interpreta os links estendidos, porém apenas a primeira conexão do recurso local para o recurso remoto. A Figura 5-26 apresenta o trecho do código com interpretação a uma parte dos links estendidos. O código foi dividido em 7 blocos para melhor compreensão.

O primeiro bloco trata da declaração de variáveis. O segundo bloco refere-se à identificação do tipo de XLink e seu atributo *(title).* A linha 6 identifica o valor do atributo *type*. Caso o seu valor seja "*extended*", é verificado o seu atributo na lista de atributos xlink:attributes (linha 1). Na linha 7 é recuperado o valor do atributo *title*.

O terceiro bloco verifica se o atributo *type* do nó do primeiro filho tem valor igual a "*resource*". Caso afirmativo, recupera o valor do atributo *label* e valida seu modo de exibição através da função *validateShow* com valor nulo indicando que seu modo de exibição será o default.

No quarto bloco são avaliados todos os nós filhos. Para cada um deles é avaliado se o valor do atributo *type* é igual a "*arc*" assim como se o valor do atributo *from* tem o mesmo valor do *label* do nó do primeiro filho e se o valor do atributo *to* é igual ao valor do *label* do primeiro nó irmão. Caso positivo, é definido o modo de exibição através do atributo *show*.

Em seguida, o valor *href* do primeiro nó irmão é recuperado e interpretado da mesma forma que os links simples, já comentado, anteriormente. O sexto bloco apresenta o código que

exibe a imagem “expandir. gif” que permite que o usuário ao clicá-la ative a função *extend (label, id),* passando como parâmetros o *label* do primeiro nó filho e o id.

```
function readXlink(xlink , id) {
    1          XLinkAttributes = xlink.attributes
    2  [1]     XLinkType = XLinkAttributes.getNamedItem("xlink:type").value
    3          name = xlink.firstChild.firstChild
    4          nameNode = xlink.firstChild
-------------------------------------------------------------------------------------------
    6   if (XLinkType == "extended") {
    7           //XLinkTitle = XLinkAttributes.getNamedItem("xlink:title").value
    8  [2]         do {
-------------------------------------------------------------------------------
   10                 if (nameNode.attributes.getNamedItem("xlink:type").value == "resource")
   11       {
   12                         homeSiteNode = nameNode.nextSibling
   13                         attributesHome = homeSiteNode.attributes
   14  [3]                    attributesN = nameNode.attributes
   15                         label = attributesN.getNamedItem("xlink:label").value
   16                         node = xlink.firstChild
   17                         target = validateShow(" ")
-----------------------------------------------------------------------------------------------
   18  [4]                do {
   19                           if(node.attributes.getNamedItem("xlink:type").value == "arc" &&
   20                           node.attributes.getNamedItem("xlink:from").value == label &&
   21                                  node.attributes.getNamedItem("xlink:to").value ==
   22                                  attributesHome.getNamedItem("xlink:label").value) {
   23                                  show = node.attributes.getNamedItem("xlink:show")
   24                                              if (show != null) {
   25                                                     target = validateShow(show.value)
   26                                              } else {
   27                                                     target = validateShow(" ")
   28                                              }
   29                                       }
   30                                       node = node.nextSibling
   31                       } while (node != name.lastChild)
-----------------------------------------------------------------------------------------------
   33                                 href = attributesHome.getNamedItem("xlink:href").value
   34                                 xml = href.indexOf(".xml")
   35   [5]                           gif = href.indexOf(".gif")
   36                                 jpg = href.indexOf(".jpg")
   37                                 if( xml != -1 && target != "embed") {
   38                                        href = "javascript:setFile(\"" + href + "\")"
   39                                 } else if(xml != -1 || gif != -1 || jpg != -1) {
   40                                        href = "javascript:embed(\"" + href + "\")"
   41                                        target = validateShow(" ")
   42                                 }
-----------------------------------------------------------------------------------------------
   44  [6]                           xmlDIV.innerHTML +=  "<br><br><a href=" + href + " target
   45                                  = " + target + ">" + nameNode.firstChild.nodeValue +
   46                                  "</a>    <a href=javascript:extend(\"" + label
   47                                  + "\",\"" + id  + "\")><img src=\"expandir.gif\"
   48                                  alt=\"Expandir_Xlink\"></a>"
-------------------------------------------------------------------------------
   49  [7]                            if (XLinkAttributes.getNamedItem("xlink:title") != null)
   50                                 {
   51                                    title=
   52                                         XLinkAttributes.getNamedItem("xlink:title").value
   53                                    title = validateTitle(title)
   54                                 xmlDIV.innerHTML += " <a href=" + href + " target =
   55                                  " + target + "><img src=\"title.gif\" alt=" + title +
   56                                  "></a>"
   57                               }
   58                         }
   59                         nameNode = nameNode.nextSibling
   60                  } while (nameNode != name.lastChild)
```

Figura 5-26 Código da função readXLink com interpretação de uma parte dos links estendidos

A função *extend (label, id)* é apresentada na Figura 5-27. Esta função obtém o elemento raiz do arquivo xml cujo *id* é passado como parâmetro. Em seguida chama a função *findXlink*.

```
function extend(label , id ){
               xmldoc = document.all(id).XMLDocument.documentElement
               findXLink(xmldoc, label)
       }
```

Figura 5-27 Código da função extend (label, id)

A função *findXlink (root, lab)* procura o XLink que contém o *label* recebido como parâmetro. Em seguida chama a função *extendXLink.* O código da função *findXlink (root, lab)* é apresentada na Figura 5-28.

Primeiramente é analisado se o nó raiz é um XLink. Se for é verificado se é do tipo *extended.* Caso afirmativo, a função vai pesquisar em todos os filhos do nó raiz até encontrar o label que foi passado como parâmetro. Chamando assim a função *extendXlink,* que será comentada na Seção 5.4.5.

```
function findXLink(root, lab) {
     isXLink = false
       if(root.attributes != null) {
          if(root.attributes.length > 0) {
               xmlns = root.attributes.getNamedItem("xmlns:xlink")
                  if (xmlns != null && xmlns.value =="http://www.w3.org/1999/xlink"){
                       isXLink = true
                       if(root.attributes.getNamedItem("xlink:type").value == "extended") {
                              node = root.firstChild
                              stop = false
                              while( node != null && !stop && node != root.lastChild) {
                                     if(node.attributes != null) {
                                            attributesN = node.attributes
                                             labe=attributesN.getNamedItem("xlink:label")
                                             if (label != null && label.value == lab){
                                                    extendXlink(root , node, lab)
                                                    stop = true
                                             }else {

                                            node = node.nextSibling
                                             }
                                     }
                              }
                                                  }
                  }
          }
       }

       if (root.childNodes.length > 0 && !isXLink) {
               for (var  i = 0;i < root.childNodes.length;i++) {
                      if (root.childNodes.item(i) != null){
                              findXLink(root.childNodes.item(i), lab)
                     }
              }
       }
 }
```

Figura 5-28 Código da função findXlink (root, lab)

5.4.5. Interpretando o elemento resource com suas conexões

A função *extendXlink (xlink, resource, label)* foi definida para interpretar o elemento resource dos links estendidos abrindo todas as suas conexões. O código dessa função foi dividido em 6 blocos para melhor compreensão.

No primeiro bloco como, mostra a Figura 29, a função procura os nomes *locator* e *arc* em todos os nós irmãos do nó do primeiro filho. Primeiro é procurado no 1º nó irmão do nó do 1º filho o valor *locator* do atributo *type*. Em seguida procura o valor *arc* em todos os outros nós. A Figura 5-30 apresenta o segundo bloco do código. Nesse bloco, a função cria uma lista com todos os nomes *arc* e *locator*.

O terceiro bloco é apresentado na Figura 5-31. A função busca as conexões referentes ao nome *arc* e recupera seus atributos. O valor do atributo *from* é recuperado e comparado com o *label* passado como parâmetro, e o valor do atributo *to* é recuperado e comparado com o valor do *label* recuperado da lista de nomes locator.

```
       function extendXlink( xlink , resource , label) {
              homesiteNode = resource.nextSibling
              nameLoc = " "
              nameArc = " "
[1]           node = xlink.firstChild.nextSibling
              while ( node != null && !(nameLoc != " " && nameArc != " ")) {

       if( node.attributes.getNamedItem("xlink:type").value == "locator" &&
                           node.attributes.getNamedItem("xlink:title") != null){
                           nameLoc = node.nodeName
                } else if ( node.attributes.getNamedItem("xlink:type").value == "arc" {
                           nameArc = node.nodeName
                }
                    node = node.nextSibling
                }
```

Figura 5-29 Primeiro bloco do código da função extendXlink (xlink, resource, label)

```
                        listNodesLoc = xlink.getElementsByTagName(nameLoc)
[2]                     listNodesArc = xlink.getElementsByTagName(nameArc)
                        attributesWeb = xlink.attributes
                        links=""
```

Figura 5-30 Segundo bloco do código da função extendXlink (xlink, resource, label)

```
      for (var i = 0; i < listNodesArc.length; i++ ) {
                        arc = listNodesArc.nextNode
                        attributesArc = arc.attributes
                        show = attributesArc.getNamedItem("xlink:show")
                        if (label == attributesArc.getNamedItem("xlink:from").value ) {
                                listNodesLoc.reset
[3]                             for (var x = 0; x < listNodesLoc.length; x++ ) {
                                        if (show != null){
                                                target = validateShow(show.value)
                                        } else {
                                                target = validateShow(" ")
                                        }
                                        loc = listNodesLoc.nextNode
                                        attributesLoc = loc.attributes
                                if (attributesLoc.getNamedItem("xlink:label").value ==
                                    attributesArc.getNamedItem("xlink:to").value)
```

Figura 5-31 Terceiro bloco do código da função extendXlink (xlink, resource, label)

No quarto bloco são verificados e validados os atributos do XLink, conforme o código apresentado na Figura 5-32.

```
{
                actuate = attributesArc.getNamedItem("xlink:actuate")
                href = attributesLoc.getNamedItem("xlink:href").value
                if (actuate != null && actuate.value == "onLoad"){
                        open(href)
                                } else {
                                        xml = href.indexOf(".xml")
                                        gif = href.indexOf(".gif")
[4]                                     jpg = href.indexOf(".jpg")
                                        if( xml != -1 && target != "embed") {
                                                href = "javascript:setFile(\"" + href + "\")"
                                        } else if(xml != -1 || gif != -1 || jpg != -1) {
                                                href = "javascript:embed(\"" + href + "\")"
                                                target = validateShow(" ")
                                                        }
```

Figura 5-32 Quarto bloco do código da função extendXlink (xlink, resource, label)

A Figura 5-33 apresenta o quinto bloco que gera o código HTML a partir do que foi configurado do XLink.

```
if(loc.firstChild != null){
       links  += "<a href=" + href + " target=" + target + ">" + loc.firstChild.nodeValue +
"</a>"
                        } else {
       links  += "<a href=" + href + " target=" + target + ">" + href + "</a>"
                                }
       if (attributesLoc.getNamedItem("xlink:title") != null) {
            title = attributesLoc.getNamedItem("xlink:title").value
            title = validateTitle(title)
           links += "   <a href=" + href + " target=" + target + "><img
src=\"title.gif\" alt=" + title + "></a>"
                                                                                    }
       links += "<br>"
                                          }
                                       }
[5]                              }
                        }
               }
               if (attributesWeb.getNamedItem("xlink:title") != null) {
                        titleWeb =
validateTitle(attributesWeb.getNamedItem("xlink:title").value)
                        titleCode = " <a href=" + attributesWeb.nextNode.value + "><img
src=\"title.gif\" alt=" + titleWeb + "></a>"
               } else {
                        titleCode = ""
               }
```

Figura 5-33 Quinto bloco do código da função extendXlink (xlink, resource, label)

No sexto bloco a função insere o código gerado no documento, conforme mostra a Figura 5-34 e exibe o resultado na tela.

```
}

       xmlDIV.innerHTML ="<a href=" + attributesWeb.nextNode.value + ">" +
[6]    resource.firstChild.nodeValue + "</a>" + " " + "   <a
       href=javascript:validateDocument(\"xlink\")><img src=\"recolher.gif\"
       alt=\"Recolher_Xlink\"></a>" + titleCode + "<br><br>" + links
}
```

Figura 5-34 Sexto bloco do código da função extendXlink (xlink, resource, label)

5.4.6. Função xpointer (node, childSeq, show)

Esta função recebe como parâmetros o nó, a sequência de filhos e o atributo show. A Figura 5-35 apresenta o código dessa função.

No primeiro bloco é atribuído um índice de barra à variável index e um índice de barra à variável index2. Se tiver barra recupera o valor que está entre as barras. No segundo bloco utiliza-se um comando *for* para procurar o filho do nó estabelecido pela primeira barra. No terceiro bloco se for definido o atributo *show* com valor *replace*, o código é retirado e é exibido o valor do nó especificado.

```
function xpointer(node, childSeq, show){
               index = childSeq.indexOf("/")
               index2 = childSeq.substring(index+1).indexOf("/")
               if(index > -1){
[1]                    if(index2 == -1){
                              index2 = childSeq.length
                       }
-------------------------------------------------------------------------------------------
                       iChild = childSeq.substring(index + 1,index2+1)
                       child = node.firstChild
                       for(i=1;i<iChild;i++){
                              child = child.nextSibling
                              if(child.nodevalue != undefined){
[2]                                   child = child.nextSibling
                              }
                       }
                       childSeq = childSeq.substring(index2)
                       xpointer(child, childSeq,show)
-------------------------------------------------------------------------------------------
               } else  if (show == "replace"){
                       if(!node.hasChildNodes){
                              xmlDIV.innerHTML = "<br>" + node.nodeValue
                       }
                       else {
[3]                           xmlDIV.innerHTML = "<br>" + node.firstChild.nodeValue

                       }
               } else {
                       if(!node.hasChildNodes){
                              xmlDIV.innerHTML += "<br>" + node.nodeValue
                       }
                       else {
                              xmlDIV.innerHTML += "<br>" + node.firstChild.nodeValue

                       }
               }
       }
```

Figura 5-35 Código da função xpointer (node, childSeq, show)

5.4.7. Embutindo documentos

Para possibilitar o suporte ao atributo *xlink:show= "embed",* foi definida uma função *embed (href)*, que tem o objetivo de implementar o valor *embed* para documentos XML adicionando mais um arquivo. Esta chama a função *validateDocument ("embed" + count)* preservando a interpretação atual. A Figura 5-36 apresenta o código desta função.

```
function embed(href) {

		ixp = href.indexOf("#xpointer")
		if(ixp == -1) {
			if(href.indexOf(".xml") >-1){
				count = 0
				while(filesDIV.innerHTML.indexOf("embed" + count) > 0) {
					count++;
				}
				id = "embed" + count
				filesDIV.innerHTML += "<XML ID=\"" + id + "\" SRC=\"" + href +
									"\"></XML>"
				validateDocument(id)
			} else {
				xmlDIV.innerHTML += "<br><br><img src=\"" + href + "\"><br><br>"
			}

		} else{
			xml = href.substring(0,ixp)
			childSeq = href.substring(ixp + 12, href.length-1 )
			count = 0
			while(filesDIV.innerHTML.indexOf("xpointer" + count) != -1) {
					count++;
			}
			id = "xpointer" + count
			filesDIV.innerHTML += "<XML ID=\"" + id + "\" SRC=\"" + href +
								"\"></XML>"
			xmldoc = document.all(id).XMLDocument
			if(xmldoc.parseError == false) {
				xpointer(xmldoc.documentElement, childSeq, "embed")
			} else if(confirm("Erro! Motivo: " + xmldoc.parseError.reason+ " Linha:
" + xmldoc.parseError.line +
				" Texto: " + xmldoc.parseError.srcText + "\n \nVocê deseja
selecionar outro documento?")){
				selectFile()
			}

		}
	}
```

Figura 5-36 Código da função embed (href)

Primeiramente é verificada a existência da string #xpointer num documento XML. Caso afirmativo procura-se um *id* ainda não usado para o *embed.* Se não tiver xpointer no documento XML, a função procura qual *embed* vai poder ser usado.

5.5. Vantagens do IXLink

Pode-se destacar como vantagens:

- Através de um código relativamente simples escrito em JavaScript acrescenta funcionalidades de XLink e XPointer ao browser Internet Explorer, onde ele pode ser executado;

- Seu código pode ser facilmente adaptado para utilização em aplicações Web.;
- Pode ser utilizado tanto online como offline;
- Pode-se verificar o funcionamento de alguns atributos e valores do XLink com XPointer.

5.6. Considerações Finais

Neste Capítulo foram descritos o projeto e a implementação do interpretador IXLink, que permite simular o funcionamento dos links simples, estendidos e parte do XPointer usando o identificador de fragmento de Seqüência de Filhos. Verificou-se também suas características, vantagens e concluiu-se que se pode adaptar o código de IXLink em outras aplicações Web. Concluiu-se também que esta é uma alternativa simples e eficiente para a verificação do funcionamento dos links XML, uma vez que traz um bom suporte aos links simples e estendidos.

Capítulo 6 - Conclusões e Trabalhos Futuros

Neste capítulo são apresentadas as considerações finais, principais contribuições trazidas com os estudos realizados, bem como as perspectivas de trabalhos futuros.

6.1. Considerações Finais

O estudo ao longo desse trabalho tratou da pesquisa de como utilizar a linguagem de links, mais precisamente o XLink, em documentos XML.

Levando em conta a inexistência de suporte dos links XML nos principais browsers, teve-se como desafio implementar uma solução baseada na especificação de XLink (desenvolvida pela W3C), que auxiliasse na compreensão do funcionamento dos links.

A solução foi criada na forma de um interpretador, implementado em JavaScript, capaz de processar a sintaxe do XLink com alguns dos seus respectivos atributos e valores e pode ser executado no browser Internet Explorer, um browser padrão e bastante popular.

Esse interpretador, IXLink, traz suporte a uma boa parte dos links simples e estendidos do XLink, bem como à parte do XPointer usando Sequência de Filho (child sequence).

Algumas dificuldades foram encontradas no decorrer da implementação com relação a alguns aspectos, por exemplo:

- Aspecto Compreensão

  Subjetividade da especificação. Certos atributos e valores não foram implementados por falta de uma compreensão global da especificação.

- Aspecto Limitação da Linguagem de Implementação

  O atributo *show="new"* não funciona entre documentos XML, ou seja, estar dentro de um documento XML e abrir uma nova janela para outro documento XML. Pois JavaScript não aceita passagem de parâmetros ao abrir o arquivo HTML.

  Não foi possível implementar o *atributo title* posicionando o ponteiro do mouse sobre a referência do link, pois JavaScript não oferece recursos para esse tipo de comportamento. Portanto, foi implementado de outra forma que também está de acordo com a especificação.

O IXLink permite a manipulação de documentos XML contendo XLink e/ou XPointer. Esse interpretador oferece facilidade para a navegação entre os recursos envolvidos em um link XML, incluindo sofisticações como a escolha do destino a ser seguido na navegação por links multidirecionais.

Vários exemplos foram testados no IXLink, através dos quais pode-se verificar certas características do XLink, como links bidirecionais, com múltiplos destinos, com um número arbitrário de participantes, que indicam conteúdo embutido ou substituição do conteúdo, com endereçamento a qualquer região de um documento, entre outras.

Com IXLink pode-se verificar também as funcionalidades e aplicabilidade do XLink e o quanto ele poderá mudar a estória de hypertexto na Internet.

Constatou-se que os links simples procedem de forma semelhante aos links HTML. É um link unidirecional [Harold 2001]. E um link estendido é composto de conexões entre um conjunto de recursos que podem ser locais ou remotos [DeRose 2001].

Acredita-se que não se demorará muito para que os links XML se tornem o verdadeiro padrão de relacionamentos entre recursos, trazendo várias melhorias e funcionalidades aos links existentes atualmente.

O XLink e o XPointer oferecem um novo universo de possibilidades para a especificação de referências entre os dados, separando os relacionamentos propriamente ditos do comportamento esperado na travessia dos links [Maler 2001].

O uso do IXLink facilita o entendimento do funcionamento do XLink e XPointer, incluindo a capacidade de criar links inteligentes sem necessidade de recorrer a código JavaScript e possibilitando criação de sites na Web mais dinâmicos.

Assim, enquanto o mercado industrial não concebe de vez a utilização do XLink, sugere-se o interpretador IXLink como uma boa opção para aplicações simples, embora atualmente restrito ao browser Internet Explorer.

6.2. Principais Contribuições

Este trabalho teve como principais contribuições:

- Desenvolvimento de um interpretador de links para documentos XML, executado no Internet Explorer;
- Reusabilidade do código em outras aplicações Web;
- Apresentação do funcionamento dos links XML (XLink e XPointer) a partir de exemplos práticos;
- Utilização do IXLink como uma aplicação didática;
- O material apresentado neste trabalho pode vir a auxiliar em pesquisas futuras.

6.3. Trabalhos Futuros

Dentre as perspectivas de trabalhos futuros, pode-se destacar algumas possibilidades interessantes:

- Incorporar no interpretador IXLink uma base de dados de links (linkbase), já que o linkbase facilita o gerenciamento e manutenção dos links entre recursos;
- Incorporar também novas funções do XPointer como, por exemplo, ponto (point) e intervalo (range), uma vez que essas funções permitem que se aponte para locais mais específicos dentro de um documento XML;
- Fazer as alterações necessárias para o código ser interpretado em outros browsers;
- Mapear o código de JavaScript para a Linguagem Java.

Referências Bibliográficas

[Arciniegas 2000] ARCINIEGAS. F. A. **What is XLink?** Data de Publicação: Set. 2000. Disponível em: <http://www.xml.com/pub/a/2000/09/xlink/index.html?page=1>. Acesso em: mai.2001.

[Bender 2001] BENDER, M. **Desenvolvendo sites com XML**. Advanced Grafica e Editora, 2001.

[Bray 2000] BRAY, T. and PAOLI, J. and SPERBERG-McQUEEN and MALER, E. **Extensible Markup Language (XML) 1.0 (Second Edition) W3C Recommendation.** Disponível em: <http://www.w3.org/TR/REC-xml>. Acesso em: 23 mar. 2001.

[Castro 2001] CASTRO, E. **XML para a World Wide Web**. Ed.Campus,2001

[Clark 1999] CLARK, J. and DeROSE, S**. XML Path Language (XPath)** Version 1.0 – W3C Recommendation. Nov. 1999. Disponível em: < http://www.w3.org/TR/xpath>. Acesso em: out. 2002.

[DeRose 2001] DeROSE, S.; MALER, E.; ORCHARD, D. **XML Linking Language (XLink) Version 1.0. W3C Recommendation** 27 June 2001. Disponível em: <http://www.w3.org/TR/XLink/> Acesso em: 13 ago. 2001.

[DeRose 2001a] DeROSE, S. and MALER, E. and MARSH, J. **XML Pointer Language (XPointer)W3C Working Draft** Disponível em: <http://www.w3.org/TR/xptr/>. Acesso em: mai. 2002.

[DuCharme 2002] DuCHARME, B. **XLink: Who Care?** March 21, 2002; Disponível em: <http://www.xml.com/pub/a/2002/03/13/xlink.html> Acesso em: 15 abr. 2002.

[Duhig 2001] DUHIG, A. **Separating Links from Content using XML, XLink and XPointer.** Internationales Congress Centrum (ICC). 21-25 May 2001.Berlin,Germany. Disponível em:<http://www.gca.org/papers/ xmleurope2001/papers/html/s16-2.html>. Acesso em: out 2002.

[Esteves 2001] ESTEVES, A & SANTOS. L & GUIMARÃES, P. **O XML no contexto da Biblioteca Digital**. Maio de 2001.Disponível em <http://www.bibliosoft.pt/projxlink/ > Acesso em: 20 nov. 2002.

[Furgeri 2001] FURGERI, S. **Ensino Didático da Linguagem XML**: aprenda a criar padrões e documentos inteligentes com a XML. Editora Erica, São Paulo: 2001.

[Harold 2001] HAROLD, E. R. **Chapter 19 of the XML Bible, Second Edition:Xlinks.** Nov. 2001. Disponível em: <http://www.ibiblio.org/xml/books/bible2/chapters/ch19.html> Acesso em: mar. 2002.

[Harold 2001a] HAROLD, E. R. **Chapter 20 of the XML Bible, Second Edition:XPointer.** Nov. 2001. Disponível em: <http://www.ibiblio.org/xml/books/bible2/chapters/ch20.html> Acesso em: mar. 2002.

[Harold 2001b] HAROLD, E.R and MEANS, W.S. **XML in a Nutshell. Chapter 9 – XPath.** Data de Publicação**:** Jan. 2001. Disponível em: <http://www.oreilly.com/catalog/xmlnut/chapter/ch09.html> Acesso em: out. 2002.

[Heitlinger 2001] HEITLINGER, P. **O Guia Prático da XML**. Porto-Lisboa/ Portugal: Editora Centro Atlântico, 2001.

[Holzner 2001] HOLZNER, S. **Desvendando XML**. Ed. Campus, Rio de Janeiro, 2001.

[Hongchi 2002] HONGCHI S. **XML Hypertext Linking with XLink and XPointer**. Última atualização: mar. 2002. Disponível em: <http://www.cecs.missouri.edu/~shi/teaching/cecs383/lectures/23.html>. Acesso em: set. 2002.

[Jirat 2000] JIRAT, J. **XLink Reference**. Copyright (c) 2000. Disponível em: <http://zvon.org/xxl/xlink/Output/xlink_refs.html> Acesso em: jun. 2002.

[Kelly 1998] KELLY, B. **What Are...XLink and XPointer?** Institutional Web Management workshop held at Newcastle University on 15-17 Sep. 1998. Disponível em: <http://www.ariadne.ac.uk/issue 16/ what-is/intro.html> Acesso em 06 jun 2002.

[Kraus 2001] KRAUS, M. **(XLink) Hyperlinks: Linkbase Processing and Document Grouping.** Atualizada em: Jun. 2001. Disponível em: < http://www.pms.informatik.uni-muenchen.de/lehre/projekt-diplom-arbeit/linkbase-documentgroup.html>. Acesso em: mai.2002.

[Light 1999] LIGHT, R. **Iniciando em Xml**. São Paulo:Ed. Makron Books,1999.

[Lowe 2001] LOWE, D. and WILDE, E. **Improving Web linking using XLink**. Open Publish 2001. Disponível em: <www.binarything.com/binarything/ openpublish/DavidLowe1.pdf > Acesso em: jul 2002.

[Maler 1998] MALER, E. **XLink and XPointer Overview** . Data de Publicação: Apr. 1998. Disponível em: <http://www.oasis-open.org/cover/xlinkMaler980402.html >. Acesso em: mai. 2001.

[Maler 1998a] MALER, E. and DeROSE, S. **XLink Design Principles.** World Wide Web Consortium Note 3-March-1998. Disponível em: <http://www.w3.org/TR/1998/NOTE-xlink-principles-19980303>. Acesso em: out. 2001.

[Maler 1998b] MALER, E. and and DeROSE, S. **XPointer Design Principles.** World Wide Web Consortium Note 3-March-1998. Disponível em: <http://www.w3.org/TR/1998/NOTE-xlink-principles-19980303>. Acesso em: out. 2001.

[Maler 2001] MALER, E. **XML Linking: State Of The Art**. Disponível em: <http://wwws.sun.com/software/xml/developers/xlink/. Acesso em: set. 2001>

[Marchal 2000] MARCHAL, B**. XML by Exemple**. Ed.Ebras Berkley, 2000.

[Martin 2001] MARTIN, D. et al**. Professional XML**. Rio de Janeiro: Ed.Ciência Moderna, 2001.

[Mclaughlin 2002] MCLAUGHLIN, B. **How to use XLink with XML**. Feb. 2002; Disponível em: <http://www-106.ibm.com/developerworks/xml/library/x-xlink/index.html>. Acesso em: abr. 2002.

[Moller 2001] Anders e Michael I. **XLink, XPointer and XPath**. Dec. 2001. Disponível em <http://www.brics.dk/~amoeller/XML/linking/index.html>.Acesso em: out. 2002.

[Pitts-Moultis 2000] PITTS-MOULTIS, N. and KIRK, C**. XML Black Book Solução e Poder**. São Paulo: Makron Books, 2000.

[Punin 2002] PUNIN, J. **Creation of XML Documents**. Jan. 2002. Disponível em: <http://www.cs.rpi.edu/~puninj/XMLJ/classes/class9/Overview.html>. Acesso em: nov. 2002.

[Ray 2001] RAY, E. T. **Aprendendo XML**. Rio de Janeiro: Campus, 2001.

[St. Laurent 2001] ST. LAURENT, Simon. **XPointer Referencing points and ranges inside of documents.** Jan. 2001.Disponível em:. <http://www.simonstl.com/articles/xptr/xptr.html>. Acesso em: nov. 2002.

[St. Laurent 2002] ST. LAURENT, S. **XML Linking Language (XLink): Creating Powerful, Flexible Hypertext Structures.** Disponível em: <http://www.vbxml.com/xml/articles/xlink>. Acesso em: nov. 2002.

[W3CSchools 2002] W3CSCHOOLS. **XPath Tutorial** Disponível em: <http://www.w3schools.com/xpath/default.asp>. Acesso em: out. 2002.

[Wilde 2002] WILDE, E. and LOWE, D. **XPath, XLink, XPointer, and XML – A Pratical guide to Web Hyperlink and Transclusion**. Jun. 2002 Disponível em: http://www.aw.com/samplechapter/0201703440.pdf. Acesso em: jan. 2003.

[Will 2002] WILL, M. V. **XLink & XPointer**. Jan. 2002. Disponível em: <http://www.codenotes.com/do/articles/article?articleID=154>. Acesso em: mai. 2002.

[Young 2000] YOUNG, M**. XML Step by Step**. Redmond: Microsoft Press, 2000.