XPath

Carlos A. Heuser

XPath

- Sintaxe para **referenciar partes de documentos XML**
- Expressa a navegação dentro de um documento XML através de **expressões de caminho**
- Utiliza uma **biblioteca de funções**
- Faz parte do padrão XSLT, mas é usado em outros padrões XML (por exemplo, Xquery)

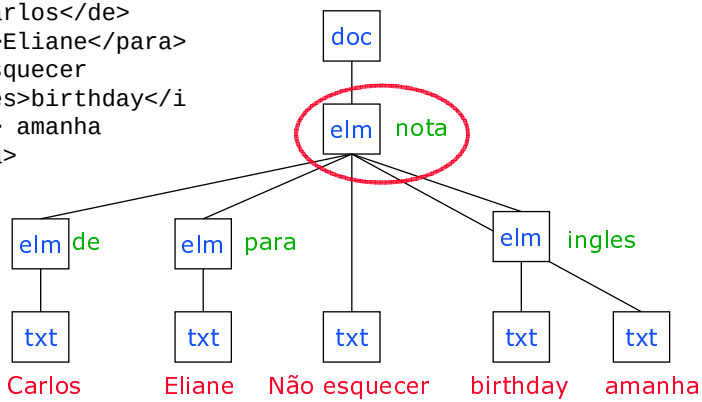
Xpath – expressões de caminho

- Expressões de caminho são semelhantes às expressões de caminho usadas para navegar dentro de diretórios de arquivos em *shells* de Windows ou Unix.
- Há dois tipos:
  - Expressão **absoluta**:
    - **Começa** com o símbolo "/"
    - Navegação **inicia na raiz** do documento XML
  - Expressão **relativa**:
    - **Não começa** com o símbolo "/"
    - É executada em relação a um **elemento contexto**
    - Navegação **inicia no** elemento **contexto**

Expressão Xpath absoluta - exemplo

/nota

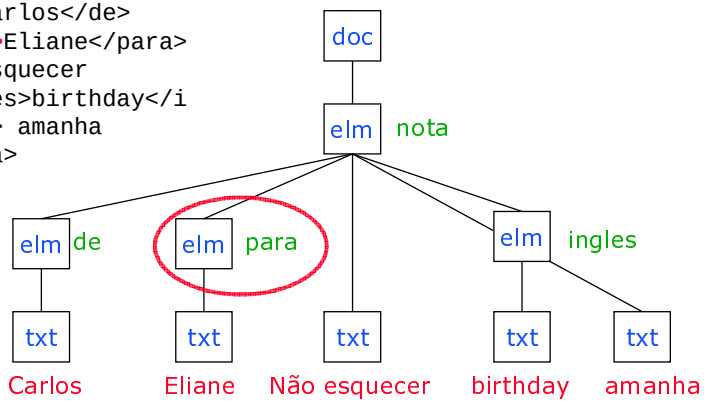
```
<nota>
<de>Carlos</de>
<para>Eliane</para>
Nao esquecer
<ingles>birthday</i
ngles> amanha
</nota>
```



Expressão Xpath absoluta - exemplo

/nota/para

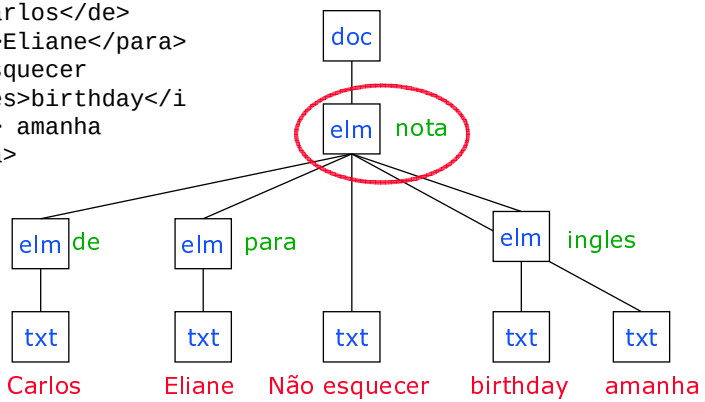
```
<nota>
<de>Carlos</de>
<para>Eliane</para>
Nao esquecer
<ingles>birthday</i
ngles> amanha
</nota>
```



Expressão Xpath absoluta - exemplo

/nota/para/..

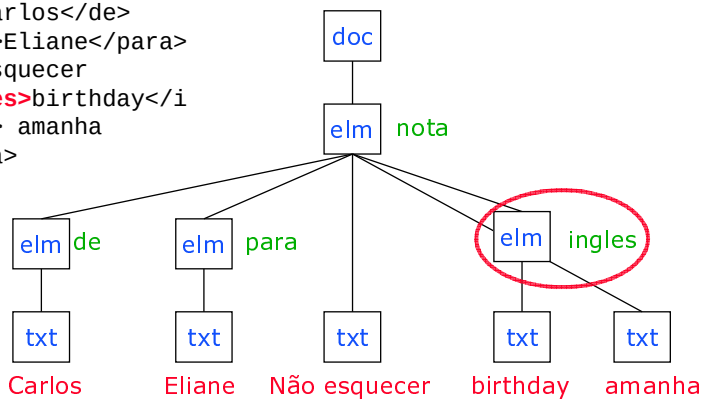
```
<nota>
<de>Carlos</de>
<para>Eliane</para>
Nao esquecer
<ingles>birthday</i
ngles> amanha
</nota>
```



Expressão Xpath absoluta - exemplo

/nota/para/../ingles

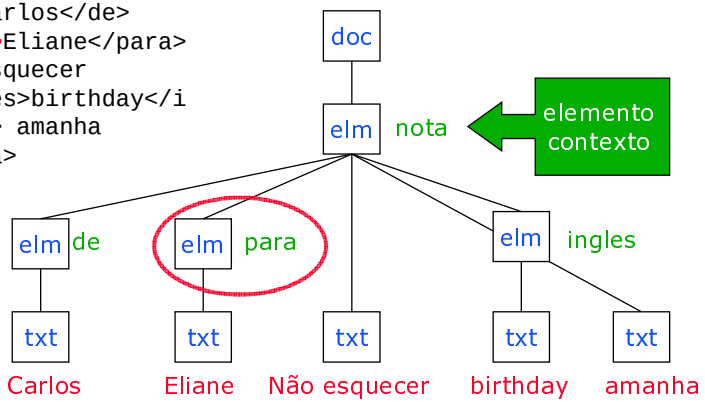
```
<nota>
<de>Carlos</de>
<para>Eliane</para>
Nao esquecer
<ingles>birthday</i
ngles> amanha
</nota>
```



Expressão Xpath relativa - exemplo

para

```
<nota>
<de>Carlos</de>
<para>Eliane</para>
Nao esquecer
<ingles>birthday</i
ngles> amanha
</nota>
```



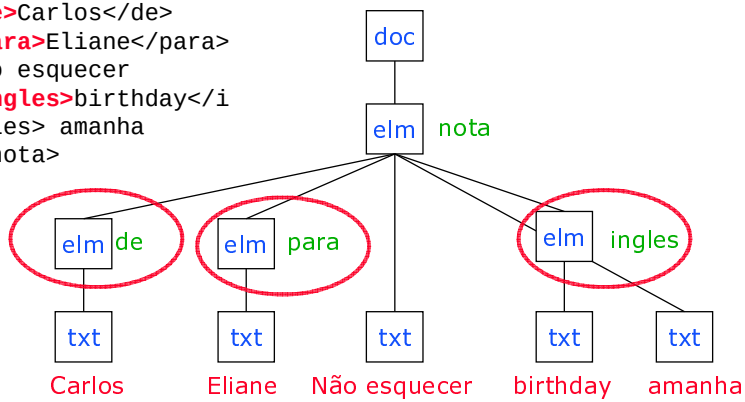
Seleção do nodo

- Para indicar qual os nodos a selecionar na expressão de caminho pode ser usado:
  - Nome do elemento
  - Símbolo "`*`" representando qualquer nome de elemento
  - Símbolo "`.`" representando o próprio nodo
  - Símbolo "`..`" representando o nodo ancestral do nodo em questão

Uso do "*" - exemplo

`/nota/*`

```
<nota>
<de>Carlos</de>
<para>Eliane</para>
Nao esquecer
<ingles>birthday</i
ngles> amanha
</nota>
```



Xpath – operadores de eixo (*axis operators*)

- A notação vista na realidade é uma notação sucinta
- A expressão
  `/nota/para`
  é uma notação sucinta para
  `/child::nota/child::para`
- `A forma geral de um termo de uma expressão de caminho é:`
  `operador_de_eixo``::`*`referência_a_nodo`*

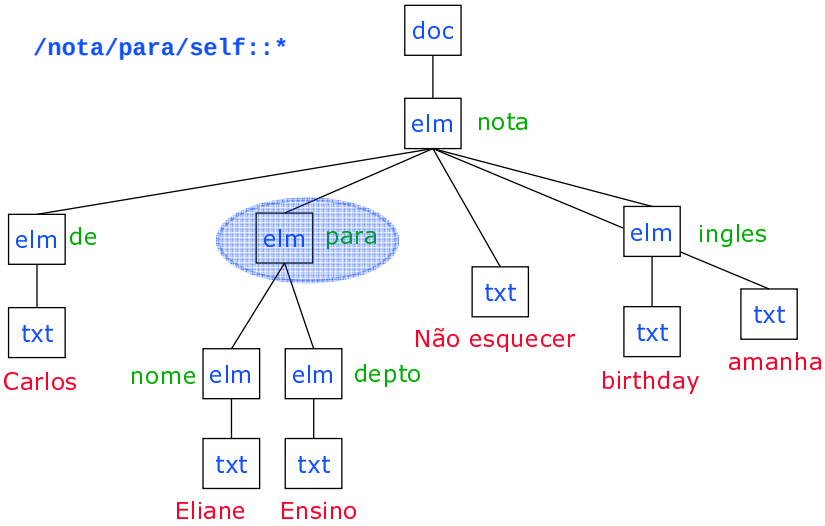
Operadores de eixo (1)

- `self::`
  - O próprio nodo
- `child::`
  - Nodos filhos (um nível da árvore)
- `parent::`
  - Nodo pai
- `preceding::`
  - Seleciona todos nodos que precedem o nodo em questão no documento (nodos ancestrais não são selecionados)
- `following::`
  - Seleciona todos nodos que sucedem o nodo em questão no documento (nodos ancestrais não são selecionados)

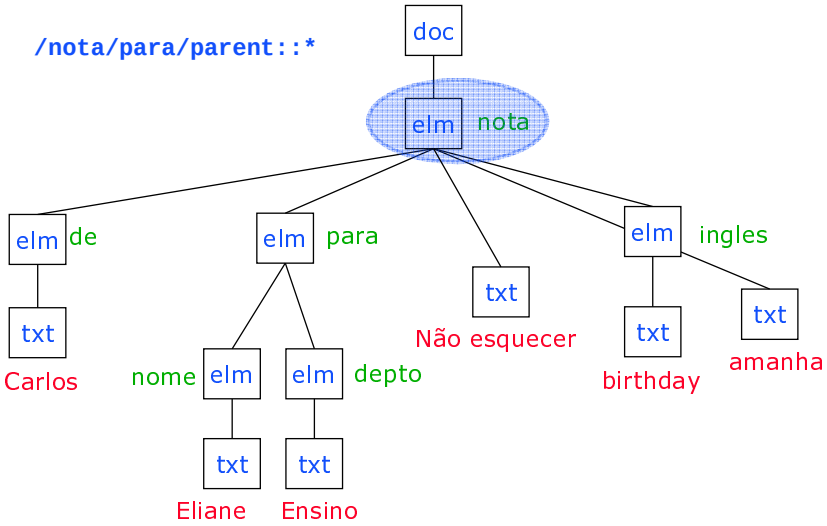
Operadores de eixo (2)

- `ancestor::`
  - Nodos ancestrais
- `descendant::`
  - Nodos descendentes
- `preceding-sibling::`
  - Seleciona todos nodos que precedem o nodo em questão no documento e que estão no mesmo nível hierárquico na árvore (nodos ancestrais não são selecionados)
- `following-sibling::`
  - Seleciona todos nodos que sucedem o nodo em questão no documento e que estão no mesmo nível hierárquico na árvore (nodos ancestrais não são selecionados)

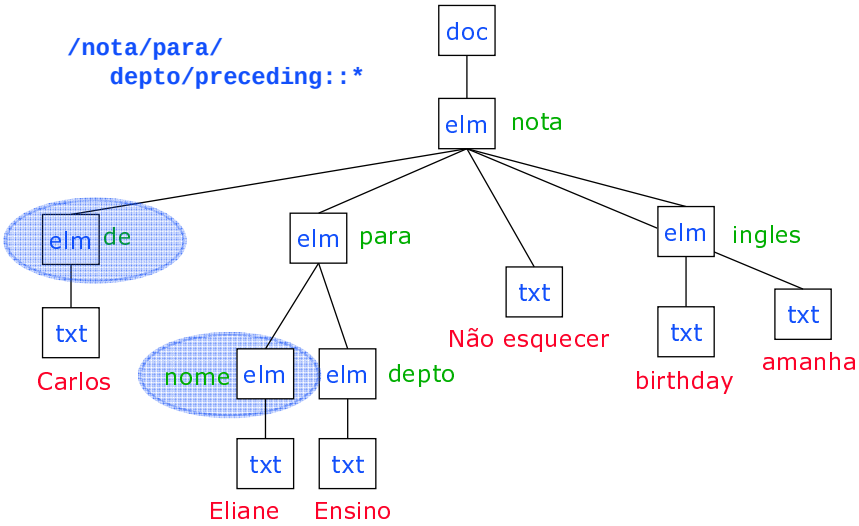
Exemplo de uso de operador de eixo



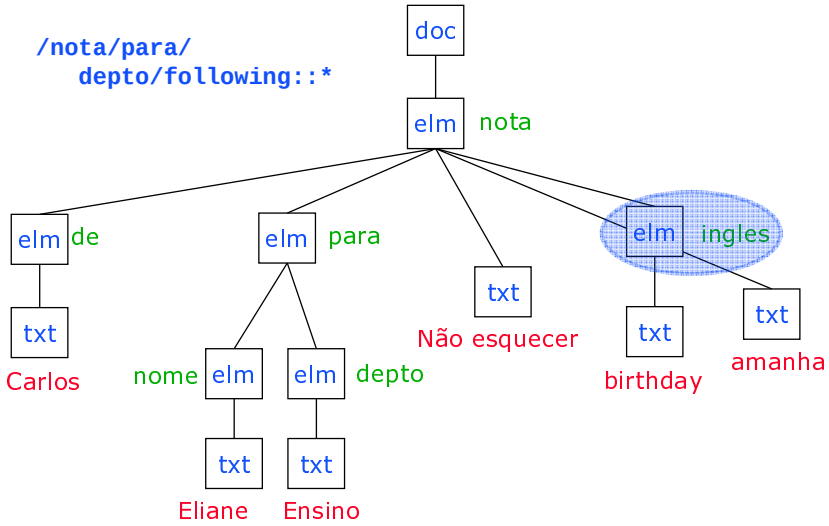
Exemplo de uso de operador de eixo



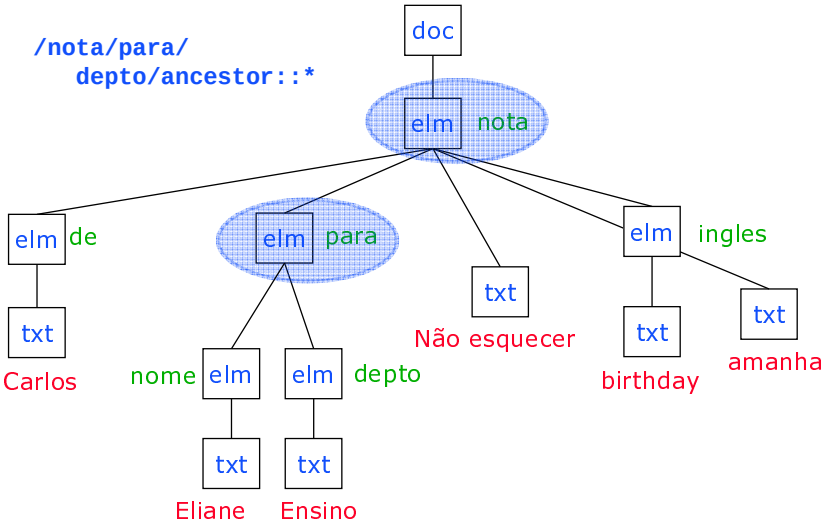
Exemplo de uso de operador de eixo



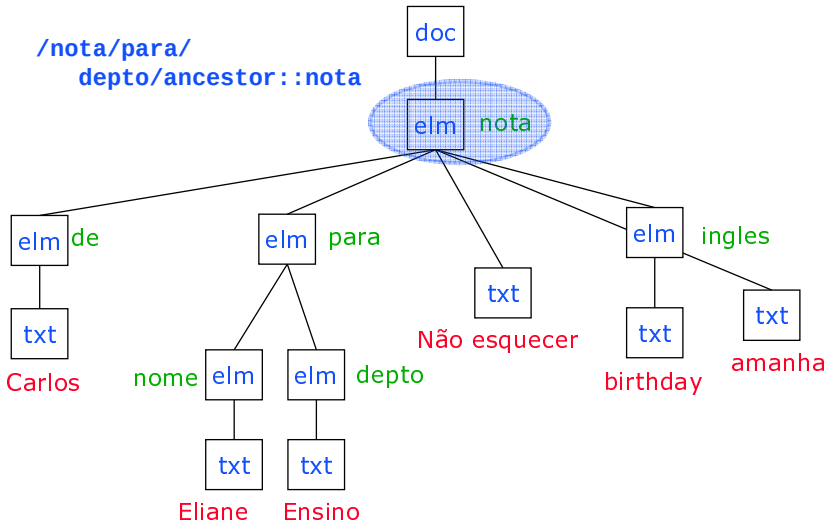
Exemplo de uso de operador de eixo



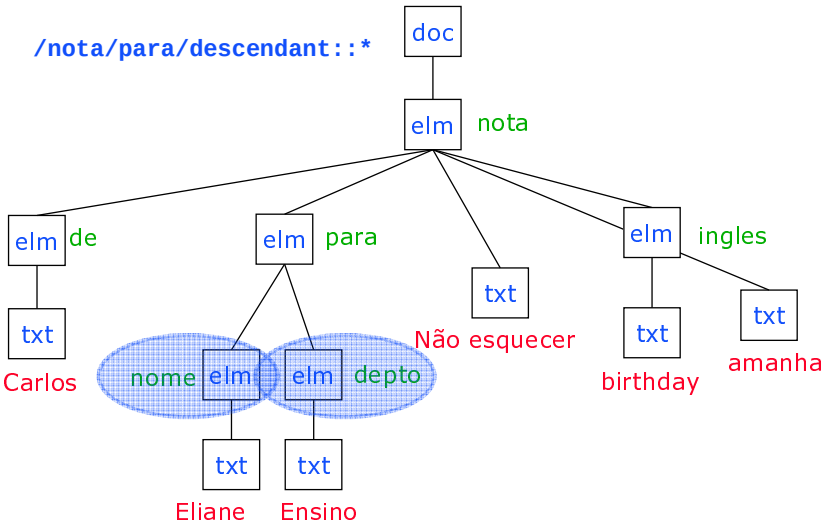
Exemplo de uso de operador de eixo



Exemplo de uso de operador de eixo

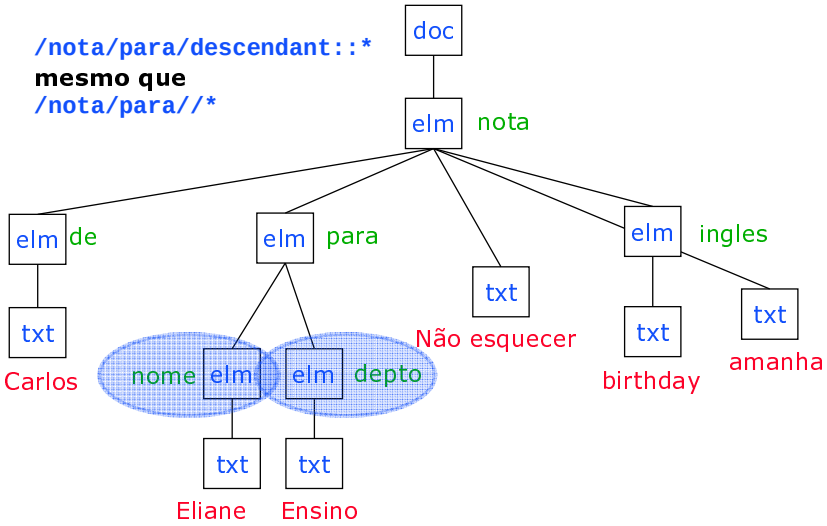


Exemplo de uso de operador de eixo



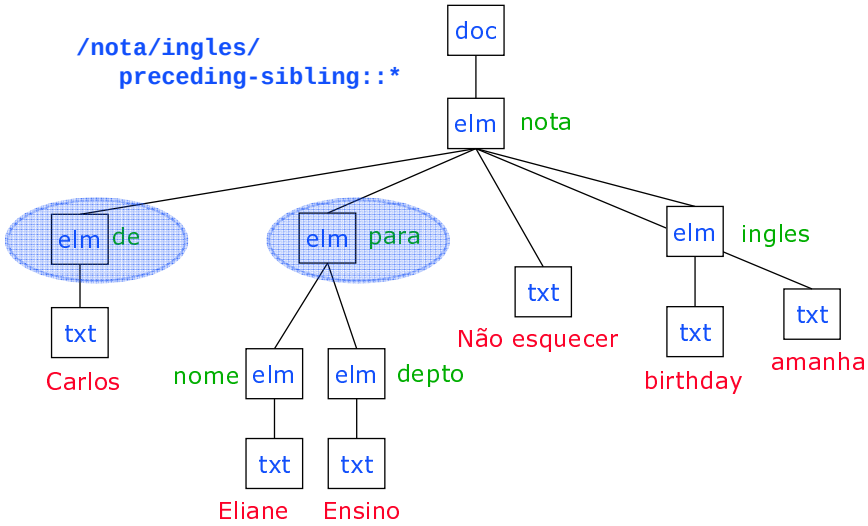
Exemplo de uso de operador de eixo

`/nota/para/descendant::*`
mesmo que
`/nota/para//*`



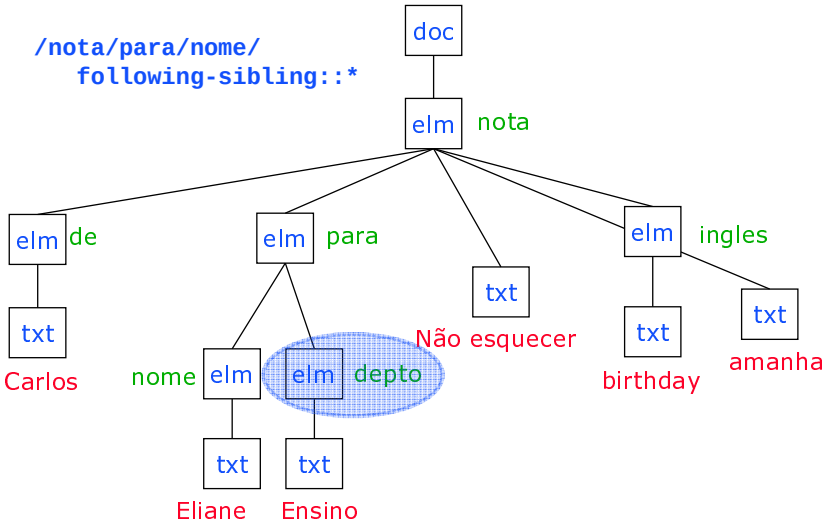
Exemplo de uso de operador de eixo

`/nota/ingles/`
`preceding-sibling::*`

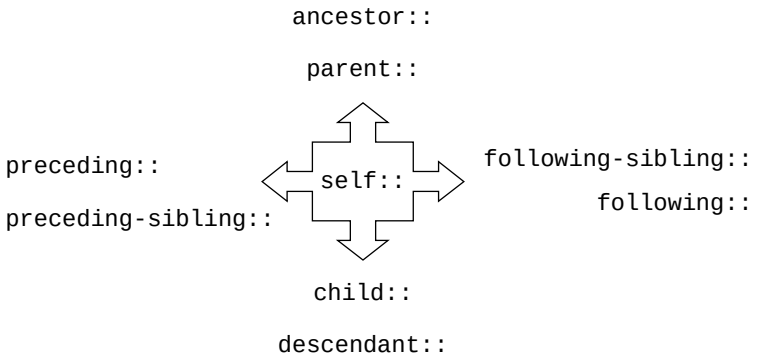


Exemplo de uso de operador de eixo

`/nota/para/nome/`
`following-sibling::*`



Operadores de eixo



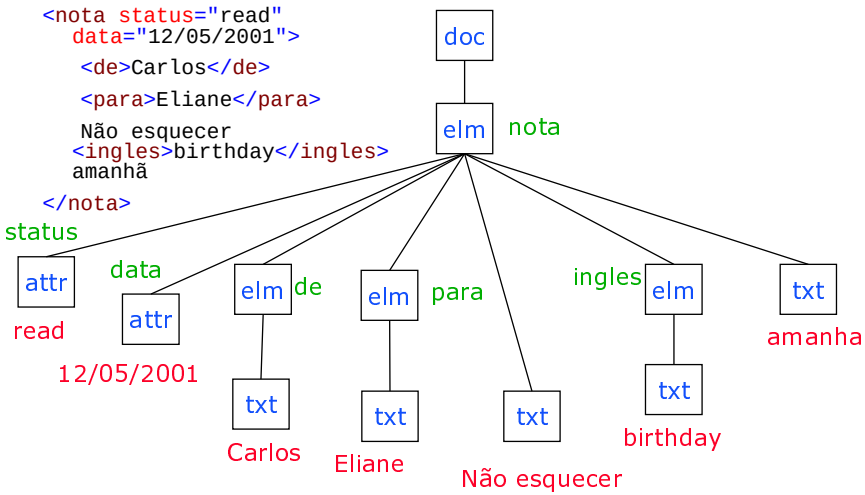
Acessando atributos e nodos de texto

- Nos exemplos vistos até aqui, os nodos referenciados sempre são **elementos**
- Existem formas de acessar também os demais tipos de nodos da árvore, como:
  - Atributos
  - Nodos de texto

Atributos e nodos texto – documento exemplo

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<nota status="read" data="12/05/2001">
   <de>Carlos</de>
   <para>Eliane</para>
   Não esquecer <ingles>birthday</ingles> amanhã
</nota>
```

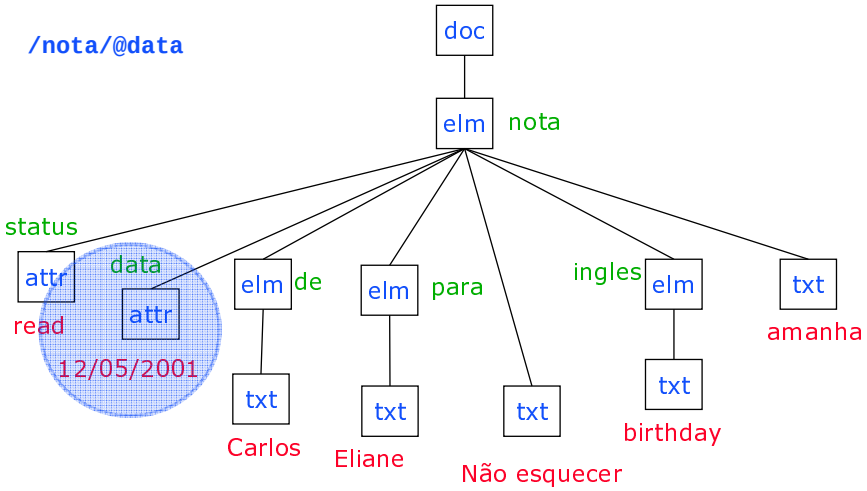

Atributos e nodos texto – documento exemplo



Acessando atributos

- Para acessar atributos usa-se o operador de eixo `attribute::`
- Exemplo:
  `/nota/attribute::status`
- Notação sucinta:
  `/nota/@status`

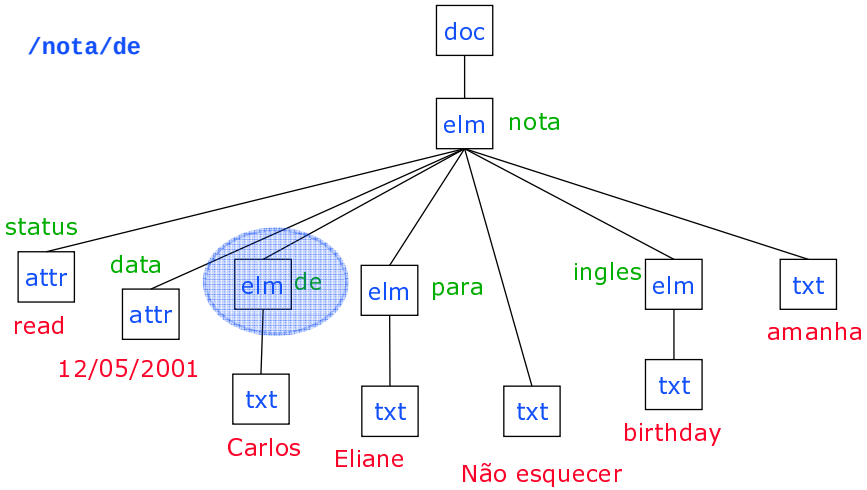
Atributos – exemplo



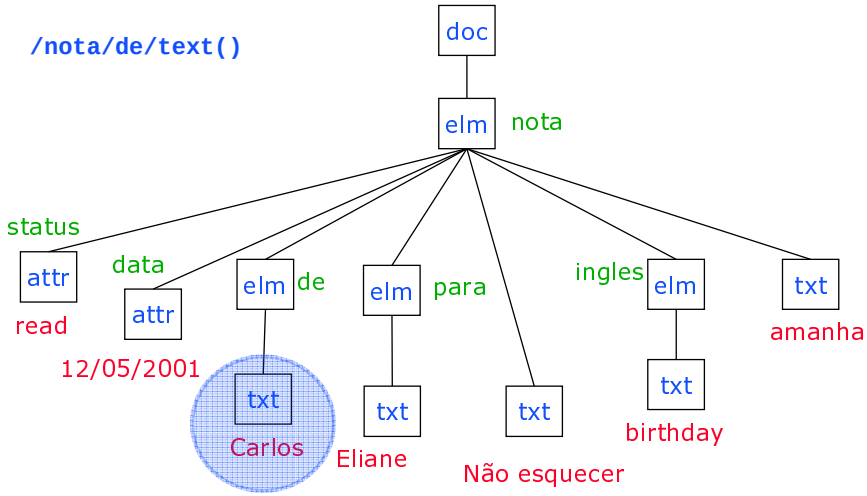
Acessando nodos texto

- Para acessar o texto contido em um elemento, usa-se a função `text()`
- Observar que há diferença entre acessar um **elemento** e acessar seu **conteúdo**.

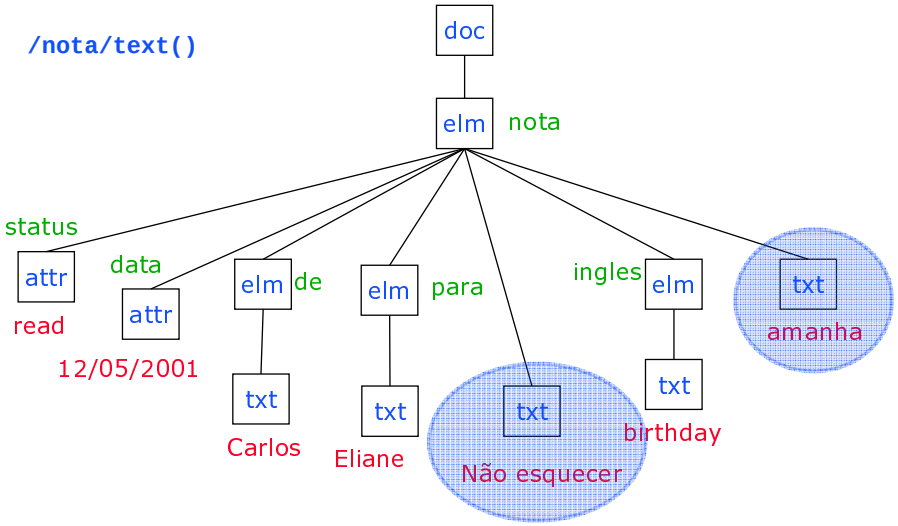
Acessando um elemento



Acessando o texto no elemento



Acessando o texto no elemento



Verificação de esquema

- Xpath não faz validação de nomes contra nenhum esquema.
- Por exemplo, considerando o documento das transparências anteriores a expressão:

```
/nota/origem
```

não retorna nada, já que não há este elemento (**origem**) no documento.

Predicados

- Dentro de um conjunto de nodos (*node-set*) é possível selecionar nodos específicos através de predicados.
- O predicado é escrito entre colchetes, após a expressão que seleciona um conjunto de nodos.
- Exemplo:

```
/livros/livro[titulo="XML"]/autor[2]
```


Predicados

- Dentro de um conjunto de nodos (*node-set*) é possível selecionar nodos específicos através de predicados.
- O predicado é escrito entre colchetes, após a expressão que seleciona um conjunto de nodos.
- Exemplo:

```
/livros/livro[titulo="XML"]/autor[2]
```

Seleciona nodos dentro dos filhos de **livros** que se chamam **livro**

Predicados

- Dentro de um conjunto de nodos (*node-set*) é possível selecionar nodos específicos através de predicados.
- O predicado é escrito entre colchetes, após a expressão que seleciona um conjunto de nodos.
- Exemplo:

```
/livros/livro[titulo="XML"]/autor[2]
```

Seleciona nodos **autor** dentro de um nodo **livro**

Predicados – documento exemplo

```
<bibliografia>
 <referencia id="b1">
   <obra>Data on the Web:
             From Relations to Semistructured Data and XML</obra>
   <autor>
     <nome>Serge Abiteboul</nome><endereco/>
   </autor>
   <autor>
     <nome>Peter Buneman</nome><endereco/>
   </autor>
   <autor>
     <nome>Dan Suciu</nome>
      <instituicao>University of Pennsylvania</instituicao>
     <instituicao>ATT - Bell labs</instituicao>
     <endereco/>
   </autor>
   <ano>1999</ano>
 </referencia>
```


Predicados – documento exemplo

```
<referencia id="b2">
    <obra>Union Types for Semistructured Data</obra>
    <autor><nome>Peter Buneman</nome><endereco/></autor>
    <autor><nome>Benjamin Pierce</nome><endereco/></autor>
    <ano>1999</ano>
    <local>University of Pennsylvania</local>
</referencia>
  <referencia id="b5">
    <obra>Optimizing regular path expressions using graph
  Schemas</obra>
    <autor><nome>Mary F. Fernandez</nome><endereco/></autor>
    <autor>
      <nome>Dan suciu</nome>
      <endereco><rua>Rua das Flores</rua></endereco>
    </autor>
    <ano>1998</ano>
    <local>IEEE Computer Society</local>
  </referencia>
</bibliografia>
```


Predicados

- Teste da **existência** de um elemento
- Buscar os autores que têm o elemento **instituição**

```
/bibliografia/referencia/autor[instituicao]
```

- Buscar as referências cujos autores que têm o elemento **instituição**

```
/bibliografia/referencia[autor/instituicao]
```

Predicado – existência de sub elemento

- Buscar os autores que têm o elemento **instituição**

```
/bibliografia/referencia/autor[instituicao]
```

- Buscar as referências que têm ao menos uma autor que tem o elemento **instituição**

```
/bibliografia/referencia[autor/instituicao]
```

Expressão XPath relativa

- **Semântica**:
  - selecionar o nodo para o qual a expressão XPath relativa retorna ao menos um nodo.

Predicado – existência de sub elemento

- Predicados podem ser usados em qualquer ponto da consulta para restingir o conjunto de nodos selecionados

- Buscar os nomes dos autores das referências que têm ao menos um autor que tem o elemento **instituição**

```
/bibliografia/
   referencia[autor/instituicao]/
   autor/
   nome
```


Predicado – teste de conteúdo

- Buscar as referências de 1999

```
/bibliografia/referencia[ano="1999"]
```

- Buscar as referências com atributo **id** igual a "b1"

```
/bibliografia/referencia[@id="b1"]
```


Predicado – testes de posição

- Função **position()** retorna a **posição** de um elemento no seu contexto

- Acessar todas referências exceto a primeira:

```
/bibliografia/referencia[position()>1]
```

- Acessar o primeiro autor de todas referência exceto a primeira

```
/bibliografia/
   referencia[position()>1]/
   autor[position()=1]
```

Função position() – notação sucinta

- Obter o primeiro autor de cada referência

```
/bibliografia/referencia/autor[position()=1]
```

- Notação sucinta

```
/bibliografia/referencia/autor[1]
```


Posição – outras funções

- Posição do último elemento - função **last()**

```
/bibliografia/referencia/autor[position()=last()]
```

ou

```
/bibliografia/referencia/autor[last()]
```

- Outro exemplo:

```
/bibliografia/referencia/
      autor[not(position()=last())]
```

- Contagem de elementos em um node-set – função **count()**

```
/bibliografia/referencia[count(autor)>2]
```


Função boolean()

- Função **boolean()** avalia uma expressão e retorna um valor boleano. Se o argumento for:
  - Número: Retorna TRUE se for diferente de zero

```
boolean(3) --> TRUE
boolean(0) --> FALSE
```

  - Conjunto de nodos: Retorna TRUE se houver ao menos uma entrada

```
boolean(title) --> TRUE
boolean() --> FALSE
```

  - String: Retorna TRUE se a string possuir ao menos um caracter

```
boolean("algum texto") --> TRUE
boolean("") --> FALSE
```


Tratamento de *strings* – função contains()

- Função

contains(*nodo,string*)

  - Verifica se o nodo contém o string

- Obter as autores cujo nome contém "P"

```
/bibliografia/
   referencia/
   autor[contains(nome,"P")]
```

Tratamento de *strings* – função `contains()`

- Considerar o seguinte fragmento de documento:

```
<secao>Esta secao apresenta...
      <paragrafo>O modelo relacional ...</paragrafo>
      <paragrafo>Como já mencionado, ...</paragrafo>
</secao>
```

- Observar a diferença entre:

```
secao[contains(.,"relacional")]
```

e

```
secao[contains(text(),"relacional")]
```


Tratamento de *strings* – função `contains()`

- Considerar o seguinte fragmento de documento:

```
<secao>Esta secao apresenta...
      <paragrafo>O modelo relacional ...</paragrafo>
      <paragrafo>Como já mencionado, ...</paragrafo>
</secao>
```

- Observar a diferença entre:

Testa o conteúdo de secao e de todos seus sub-elementos

```
secao[contains(.,"relacional")]
```

e

```
secao[contains(text(),"relacional")]
```


Tratamento de *strings* – função `contains()`

- Considerar o seguinte fragmento de documento:

```
<secao>Esta secao apresenta...
      <paragrafo>O modelo relacional ...</paragrafo>
      <paragrafo>Como já mencionado, ...</paragrafo>
</secao>
```

- Observar a diferença entre:

```
secao[contains(.,"relacional")]
```

e

```
secao[contains(text(),"relacional")]
```

Testa o conteúdo do primeiro texto em secao

Tratamento de strings – função `starts-with()`

- Função **`starts-with()`** testa o texto no começo de um string.

- Exemplo:
  - Selecionar elementos `titulo` que iniciem com a palavra "Introdução"

```
titulo[starts-with(.,"Introdução")]
```

```
<titulo>Introdução a JSP</titulo>

<titulo>_Introdução a JSP</titulo>
```

Não funciona!

Tratamento de strings – função normalize-space()

- Função **normalize-space()**
  - remove espaços no início e final
  - reduz vários espaços em branco entre palavras para um único caractere espaço

```
titulo[
   contains(normalize-space(.),"Introdução a JSP")
  ]
```

```
 <titulo>  Introdução        a        JSP  </titulo>
```


Tratamento de strings – função concat()

- Função concat() concatena strings. Pode ter um ou mais parâmetros:

```
Concat(.,text1, texto2,..., textn)
```

  - Exemplo: Retornar a seção que fale da autora do livro

```
//secao[contains(text(),concat (.,/../autor/nome[text()],
  /../autor/sobrenome[text()]))]
<livro>
   <autor>
      <nome>Ana</nome>
      <sobrenome>Silva</sobrenome>
   </autor>
   <secao> A autora Ana Silva ...</secao>
   <secao> ...  </secao>
</livro>
```


Tratamento de strings – função translate()

- Função translate() converte caracteres de acordo com um esquema de mapeamento.
  - Uso: comparações *case-insensitive*
  - Parâmetros: string para converter, caracteres para modificar no texto fonte, e valores a serem colocados

```
paragrafo[contains(
   translate(normalize-space(.),
      "abcdefghijklmnopqrstuvwxyz",
      "ABCDEFGHIJKLMNOPQRSTUVWXYZ"),
  "ELEMENTO")]
```

```
<paragrafo>EM XML, UM ELEMENTO É ...</paragrafo>
<paragrafo>Um documento XML deve possuir um elemento raiz
  ...</paragrafo>
```

- os dois elementos paragrafo são recuperados

Tratamento de strings – tamanho de string

- Função **string-length()** determina o número de caracteres em um string.

```
paragrafo[string-length(.)=15]
```

```
<paragrafo> quinze letras </paragrafo>
<paragrafo>123456789012345</paragrafo>
```

Tratamento de strings – função substring-before()

- Função **substring-before()** extrai texto do início do string.
  - Parâmetros:
    - o string de onde será extraído o texto,
    - os caracteres que terminam o prefixo a serem extraídos

```
titulo[contains(.,substring-before(.,"*"))]

<titulo>Contrução dos arquivos *.gif</titulo>
```

- Função **substring-after()** extrai texto do final da string

```
titulo[contains(.,substring-after(.,"*"))]
<titulo>Contrução dos arquivos *.gif</titulo>
```


Tratamento de strings – função substring()

- Função **substring()** extrai qualquer fragmento de um string
  - Parâmetros:
    - o string fonte,
    - posição de deslocamento,
    - número de caracteres a extrair

```
paragrafo[substring(.,13,5)="XPath"]

<paragrafo>A linguagem XPath</paragrafo>
<paragrafo>Este parágrafo com XPath não é
  recuperado</paragrafo>
```


Tratamento de números – várias funções

- Função **number()**
  - Converte o valor do argumento para números.
- Função **round()**
  - Converte reais para o inteiro mais próximo equivalente
- Função **floor()**
  - Converte reais para o inteiro inferior mais próximo equivalente
- Função **ceiling()**
  - Converte reais para o inteiro superior mais próximo equivalente
- Função **mod()**
  - Fornece o resto de uma divisão inteira

Filtros múltiplos

- São usados quando um teste de posição e um outro tipo de teste precisam ser combinados.
  - Selecionar nomes de companhias, e então extrair o terceiro nome da lista

    name[company][3]
  - Selecionar o terceiro nome, fornecendo o nome da companhia (só seleciona se for uma companhia!)

    name[3][company]

União de node-sets

- O resultado de várias expressões XPath pode ser unido

```
/nota[de|para]
```

Recuperar elementos **nota** que tenham um filho **de** ou um filho **para**

```
/nota/de|/nota/para
```

Recuperar os elementos **/nota/de** unido com os elementos **/nota/para**