XML (III)

Linguagens de Anotação de Documentos

Transformações XML

Transformações XML

- O XML é uma “meta-linguagem” que permite estruturar informação arbitrária desde que a “árvore” esteja bem formada
- DTDs podem ser usados para definir uma camada adicional de validação, definindo as anotações válidas e qual o seu conteúdo
- Expressões XPath podem ser usadas para navegar documentos XML e para efetuar consultas básicas
- Por vezes é necessário *transformar* os dados para diferentes formatos para que possam ser processados por outras ferramentas ou noutros contextos

Transformações XML

- Uma transformação XML especifica como elementos duma árvore de *origem* são representados numa árvore de *destino*
- Permite que os dados do nosso domínio de aplicação sejam processados por outras ferramentas
- E.g.: transformação para o esquema HTML permite que os dados sejam apresentados no *browser*
- Em XML são geralmente definidas usando XSLT, uma linguagem também gerida pela W3C

eXtensible Stylesheet Language

- Ficheiros XML não têm estilo ou formatação predefinida, sendo simplesmente apresentados como uma “árvore”
- As linguagens XSL (*eXtensible Stylesheet Language*) foram propostas para definir estilos de documentos XML
- O XSLT (*XSL Transformations*) em particular permite transformar documentos XML noutros formatos para que possam ser apresentados
- A linguagem de consulta XPath faz também parte das linguagens XSL

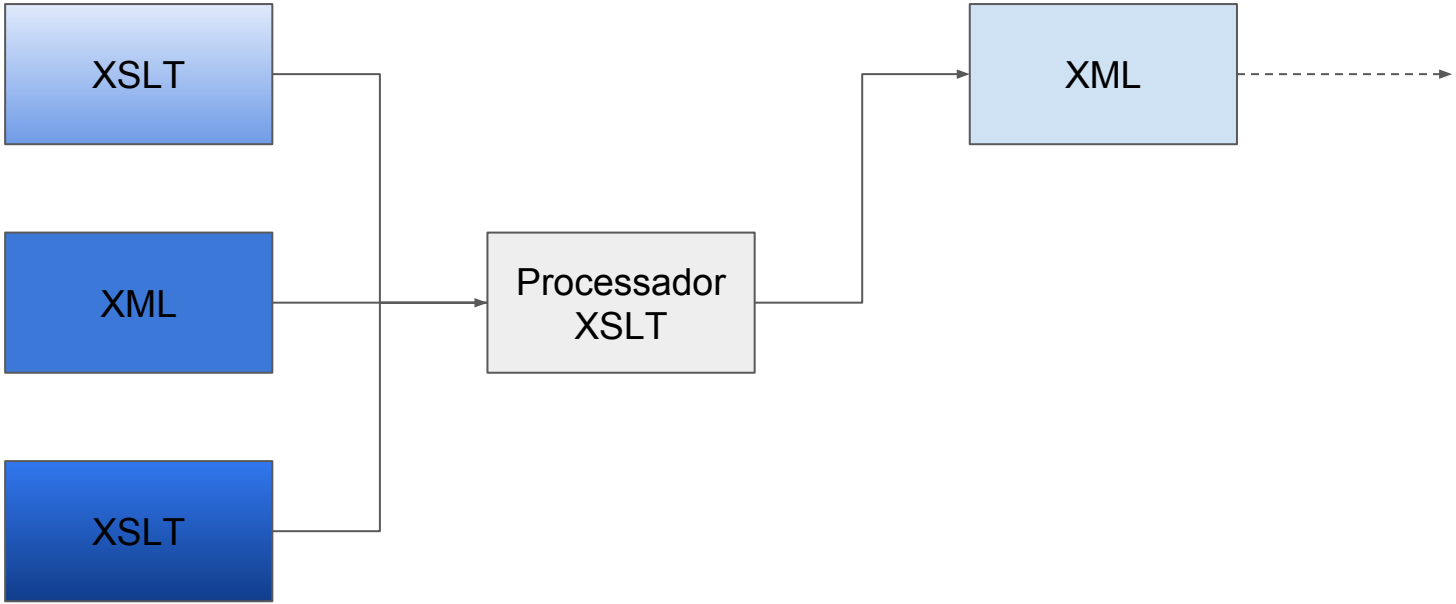
XSL Transformations

- Documentos XSLT definem regras para transformar elementos do XML de origem noutros elementos XML
- Para selecionar os elementos a serem transformados são usadas expressões XPath
- Outros operadores permitem também reorganizar ou tomar decisões alternativas dentro das transformações

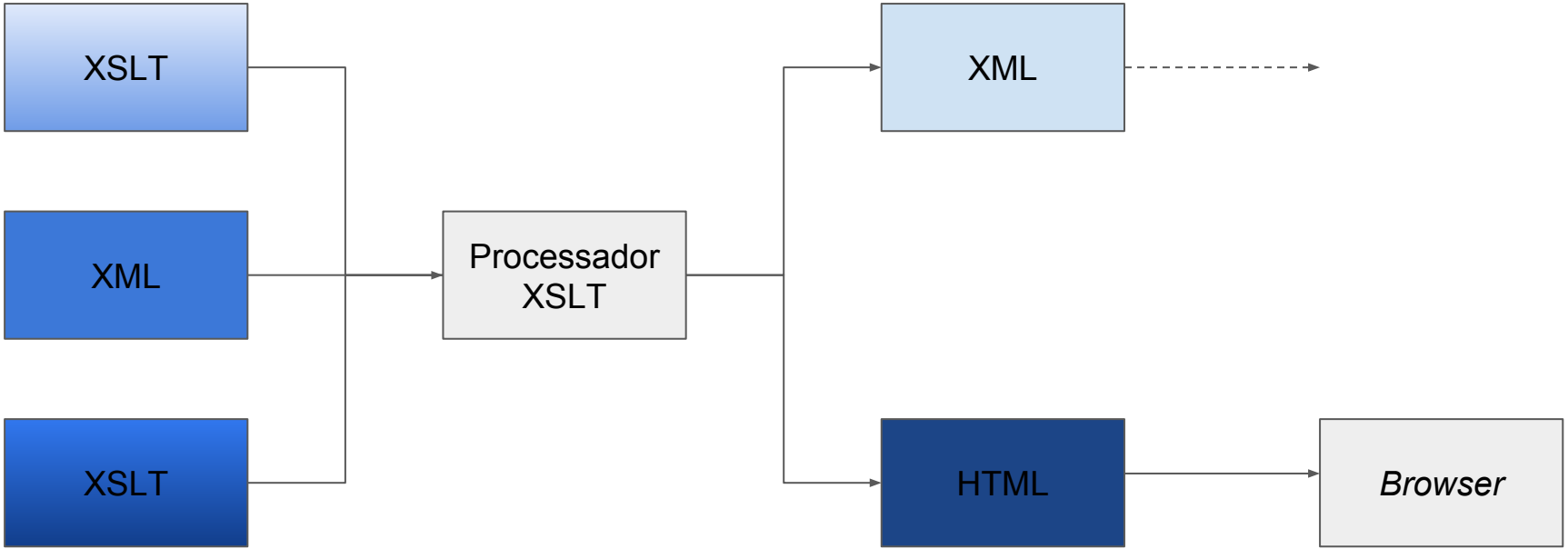
XSL Transformations

- Transformações XSLT são elas próprias documentos XML, definidos de acordo com o esquema respectivo esquema
- São geralmente definidas num documento independente para permitir a separação de preocupações (tal como o HTML / CSS)
- A aplicação efetiva duma transformação XSLT a um documento XML é feita por processadores de XSLT

XSL Transformations



XSL Transformations



Documentos XSLT

- Um ficheiro XSLT tem que ser declarado como tal (`xsl:stylesheet` ou `xsl:transform`) na sua raíz

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

- O atributo `xmlns:xsl` permite a utilização dos elementos XSLT dentro do elemento com o *namepace xsl*
- Um documento XML pode depois referenciar o ficheiro XSLT para definir o seu estilo

```
<?xml-stylesheet type="text/xsl" href="ficheiro.xsl"?>
```

Templates XSLT

- As regras de transformação em XSLT são definidas através de *templates*
- O domínio de atuação de cada *template* é definido através de “correspondências” (*matches*)
- Estas são definidas usando expressões XPath sobre o XML de origem
- O conteúdo de cada *template* é criado para cada elemento do XML de origem selecionado pela expressão XPath

Templates XSLT

- Declarados como:

```
<xsl:template match="expressão xpath">
    ...
    conteúdo a criar
    ...
</xsl:template>
```

Exemplo XSLT

```
<lad>
    <descricao>Registo dos grupos</descricao>
    <grupos ano="15/16">
        <grupo id="g1">
            <aluno>
                <nome>Aluno A</nome>
                <numero>a1</numero>
                <nota>10</nota>
            </aluno>
            ...
```

Exemplo XSLT

```
<?xml-stylesheet type="text/xsl" href="grupos.xsl"?>
<lad>
    <descricao>Registo dos grupos</descricao>
    <grupos ano="15/16">
        <grupo id="g1">
            <aluno>
                <nome>Aluno A</nome>
                <numero>a1</numero>
                <nota>10</nota>
            </aluno>
            ...
```

Exemplo XSLT

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
        <html>
        <body>
        Converti o XML para HTML.
        </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

Extração de Valores

- Depois de definir as regras de aplicação dos *templates*, é preciso definir que conteúdo vai ser criado a partir dos elementos selecionados
- Esse conteúdo é extraído com elementos (vazios) `value-of` sobre os elementos XML selecionados pela expressão XPath `select`

```
<xsl:value-of select="expressão xpath"/>
```

- Os caminhos são calculados a partir do caminho atual

Extração de Valores

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
        <html>
        <body>
        Converti o <xsl:value-of select="lad/descricao"/> para HTML.
        </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

Iteração de Elementos

- Quando um elemento contém vários elementos aninhados, geralmente queremos transformar cada um deles *individualmente*
- Para isso podemos usar o operador `for-each` que itera sobre elementos selecionados pela expressão XPath `select`

```
<xsl:for-each select="expressão xpath"/>
```

Iteração de Elementos

```
<xsl:template match="/">
    <html>
    <body>
    Converti o <xsl:value-of select="lad/descricao"/> para HTML.
    <xsl:for-each select="grupos[@ano='16/17']/grupo/aluno">
        <xsl:value-of select="nome"/>
    </xsl:for-each>
    ...
```

Ordenação

- Facilmente ordenados com elementos (vazios) `sort` de acordo com elementos selecionados pela expressão XPath `select`

```
<xsl:sort select="expressão xpath"/>
```

- O atributo `sort` define se a ordem é ascendente ou descendente (predefinido é ascendente)

Ordenação

```
<xsl:template match="/">
    <html>
    <body>
    Converti o <xsl:value-of select="lad/descricao"/> para HTML.
    <xsl:for-each select="grupos[@ano='16/17']/grupo/aluno">
        <xsl:sort select="numero"/>
        <xsl:value-of select="nome"/>
    </xsl:for-each>
    ...
```

Condicionais

- É comum querermos gerar elementos diferentes dependendo do conteúdo dos elementos de origem
- Para tal é necessário definir a *condição* e as ações *alternativas* para os diferentes casos
- Em XSLT elementos `if` e `choose` permitem esse tipo de comportamento

Condicionais

- Apenas aplica a regra se certa condição for verdadeira

```
<xsl:if test="expressão">
    ...
    conteúdo a criar
    ...
</xsl:if>
```

Condicionais

- Testa várias condições, caso contrário usa caso predefinido

```
<xsl:choose>
    <xsl:when test="expressão">
    ... conteúdo a criar ...
    </xsl:when>
    <xsl:otherwise>
    ... conteúdo a criar ...
    </xsl:otherwise>
</xsl:choose>
```

Condicionais

```
<xsl:template match="/">
    <html>
    <body>
    Converti o <xsl:value-of select="lad/descricao"/> para HTML.
    <xsl:for-each select="grupos[@ano='16/17']/grupo/aluno">
        <xsl:sort select="numero"/>
        <xsl:if test="nota &gt; 9">
            <xsl:value-of select="nome"/>
        </xsl:if>
    </xsl:for-each>
    ...
```

Aplicação de *Templates*

- Para transformações complexas é importante partir os *templates* em regras mais pequenas
- Elementos `apply-templates` dizem ao processador para procurar *templates* apropriados para os filhos do elemento atual selecionados por `select`

```
<xsl:apply-templates select="expressão xpath"/>
```

Aplicação de *Templates*

```
<xsl:template match="/">
    <html>
    <body>
    Converti o <xsl:value-of select="lad/descricao"/> para HTML.
    <table>
    <xsl:for-each select="grupos[@ano='16/17']/grupo">
        <xsl:apply-templates select="aluno[nota]">
            <xsl:sort select="numero"/>
        </xsl:apply-templates>
    </xsl:for-each>
    </table>
    ...
```

Aplicação de *Templates*

```
<xsl:template match="aluno">
    <tr>
    <xsl:choose>
        <xsl:when test="nota &lt; 10">
            <td><xsl:value-of select="nome"/></td>
        </xsl:when>
        <xsl:otherwise>
            <td style="color:red;"><xsl:value-of select="nome"/></td>
        </xsl:otherwise>
    </xsl:choose>
    </tr>
</xsl:template>
```

Take-home Lesson

- O XML é extremamente flexível e aplicado em contextos variados, pelo que é comum ter que se traduzir documentos entre esquemas
- Além disso, não tem um estilo de apresentação predefinido, pelo que uma tradução para o esquema HTML permite facilmente apresentar dados XML
- O XSLT foi desenvolvido precisamente para definir transformações entre árvores de XML origem e árvores XML destino
- Permite se navegar os documentos XML usam-se expressões XPath e outros operadores para iterar elementos e tomar decisões condicionais

Tutorial

https://www.w3schools.com/xml/xsl_intro.asp