XML

Carlos Bazilio

Depto de Ciência e Tecnologia
Pólo Universitário de Rio das Ostras
Universidade Federal Fluminense

XML

- XML significa eXtensible Markup Language (linguagem de marcadores extensível)
- Foi projetada para armazenamento e transporte de dados

XML – Formato “Qualquer”

```
<?xml version="1.0" encoding="UTF-8"?>
<livros>
        <livro isbn=“0001”>
                <titulo>JavaServer Pages</titulo>
                <autor>Nick Todd</autor>
                <editora>Campus</editora>
                <assunto>JSP</assunto>
        </livro>
        <livro isbn=“0002”>
                <titulo>Meu pé de laranja lima</titulo>
                <editora>Vozes</editora>
                <autor>Brilhante</autor>
        </livro>
</livros>
```

XML – RSS

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<?xml-stylesheet type="text/css" href="rss.css"?>
<rss version="2.0">
        <channel>
                <title>O GLOBO » Ciência</title>
                <link><![CDATA[http://www.oglobo.com.br/]]></link>
                <description>Plantão de Notícias RSS do GLOBO - Ciência</description>
                <docs>http://blogs.law.harvard.edu/tech/rss/</docs>
                <copyright>Infoglobo Comunicações LTDA. - Agência O Globo</copyright>
                <generator>Este é um arquivo RSS. Para maiores informações sobre o formato e como
utilizá-lo, acesse http://www.oglobo.com.br/rss .</generator>
                <language>pt-br</language> <ttl>10</ttl>
                <image>
                        <url>http://oglobo.globo.com/rss/logo.png</url>
                        <title>O GLOBO</title>
                        <link><![CDATA[http://www.oglobo.com.br/]]></link>
                        <width>120</width>
                        <height>30</height>
                </image>
                <pubDate>Wed, 4 Feb 2009 10:17:00 GMT</pubDate>
                <lastBuildDate>Wed, 4 Feb 2009 10:17:00 GMT</lastBuildDate>
                <item>
                        <title>Nasa adia lançamento de Discovery ao menos por uma semana</title>
                        <link><![CDATA[http://oglobo.globo.com/ciencia/mat/2009/02/04/nasa-adia-
lancamento-de-discovery-ao-menos-por-uma-semana-754255306.asp]]></link>
                        <description>&lt;p&gt;...&lt;/p&gt;&lt;p&gt;&lt;a
href="http://oglobo.globo.com/ciencia/mat/2009/02/04/nasa-adia-lancamento-de-discovery-ao-menos-por-uma-semana-
754255306.asp"&gt;Leia mais&lt;/a&gt;&lt;/p&gt;
&lt;p&gt;&lt;a href="https://seguro.oglobo.com.br/assinatura/"&gt;Assine O GLOBO&lt;/a&gt; e receba todo o conteúdo
do jornal na sua casa&lt;/p&gt;
   </description>
                        <pubDate>Wed, 4 Feb 2009 10:17:00 GMT</pubDate>
                        <localDataHora>04-02-2009 08:17:00</localDataHora>
                </item>
                <item ... />
        </channel>
</rss>
```

XML

- Um arquivo XML é definido estritamente por:
  - Um arquivo em formato ASCII
  - Tags aninhadas hierarquicamente (número indeterminado de sub-elementos)
  - Um único elemento (<tag></tag>) raiz
  - Marcadores de início e fim de uma tag (diferente de html – por exemplo,
)
  - Um número indeterminado de instruções de processamento <? app ... ?> antes da raiz
    - Ex: <?xml ... ?>, <?xml-stylesheet ... ?>
    - Não faz parte do documento XML em si, mas serve para instruir aplicações específicas que utilizam o documento
  - Indeterminado número de atributos numa tag
    - <tag atr1="val1"... atrn="valn"></tag>

XML

- Conteúdos de elementos (tags), valores de atributos, podem ter qualquer valor, exceto alguns reservados à sintaxe XML:
  - Por exemplo: <, >, “
- Neste caso, há 2 possibilidades:
  - Substituirmos as ocorrências por entidades
    - < é o caracter <
    - > é o caracter >
    - ' é o caracter '
    - " é o caracter "
    - & é o caracter &
  - Inserir o texto em seções CDATA (trechos que não são processados pelo parser XML)
    - <![CDATA[...]]

XML

- Usos:
  - Trocas de informações entre aplicativos
  - Arquivos de configuração
  - Formatos de entrada e/ou saída de aplicativos
  - Etc

Esquema XML

- Documentos XSD (esquemas), assim como DTDs, definem uma gramática para documentos XML
- Como diferenças temos:
  - XSD é definido na própria linguagem XML
  - É extensível
  - Suporta tipos de dados
  - Suporta namespaces

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<livros>
        <livro isbn=“0001”>
                <titulo>JavaServer Pages</titulo>
                <autor>Nick Todd</autor>
                <editora>Campus</editora>
                <assunto>JSP</assunto>
        </livro>
        <livro isbn=“0002”>
                <titulo>Meu pé de laranja lima</titulo>
                <editora>Vozes</editora>
                <autor>Brilhante</autor>
        </livro>
</livros>
```

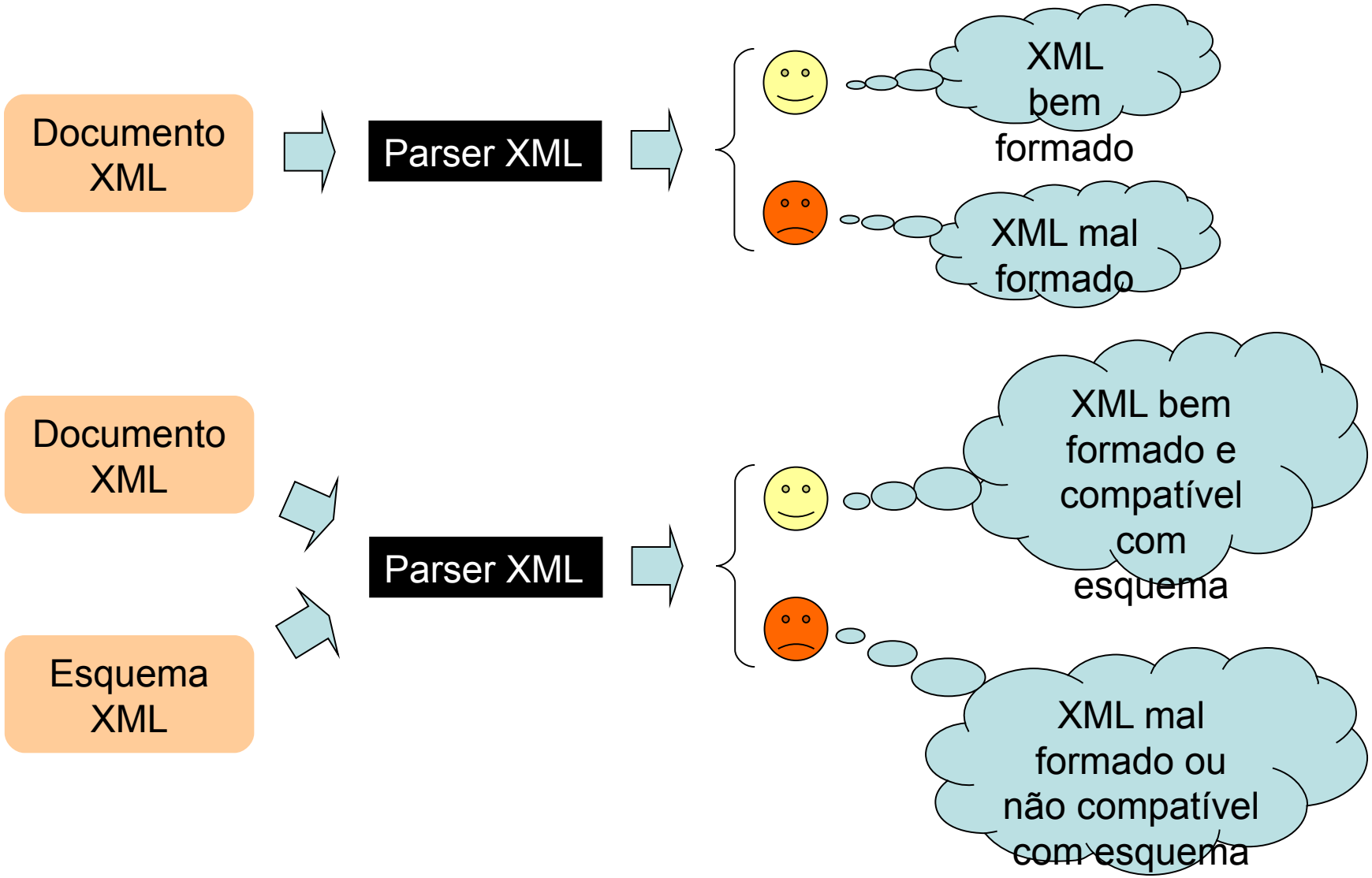
DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!--DTD generated by XMLSpy v2008
(http://www.altova.com)-->
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT livros ((livro+))>
<!ATTLIST livros
        xmlns:xsi CDATA #FIXED
"http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation CDATA
#IMPLIED>
<!ELEMENT livro ((titulo, ((editora, autor) | (autor, editora,
assunto))))>
<!ELEMENT editora (#PCDATA)>
<!ELEMENT autor (#PCDATA)>
<!ELEMENT assunto (#PCDATA)>
```

Esquema XML

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="livros">
    <xs:complexType>
      <xs:sequence>
      <xs:element ref="livro" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="livro">
    <xs:complexType>
      <xs:all>
      <xs:element name="titulo" type="xs:string"/>
      <xs:element name="autor" type="xs:string"/>
      <xs:element name="editora" type="xs:string"/>
      <xs:element name="assunto" type="xs:string" minOccurs="0"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Verificação de um XML



XML Namespace

- Um namespace é uma coleção de nomes (vocabulário)
- É útil para evitar conflito entre nomes de elementos
- É identificado por um prefixo (referência URL)

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="livro">
    <xs:complexType>
      <xs:all>
        <xs:element name="titulo" type="xs:string"/>
        ...
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Namespace que está sendo declarado

Referências ao namespace

XML Namespace

- Também podemos indicar qual é o namespace padrão de um documento omitindo o prefixo

```
<schema xmlns="http://www.w3.org/2001/XMLSchema">
<element name="livro">
    <complexType>
      <all>
        <element name="titulo" type="string"/>
        ...
      </all>
    </complexType>
  </element>
</schema>
```

XML Namespace

- Para os parsers, estas urls são utilizadas somente para distinguir os nomes dos elementos
- Entretanto, alguns fabricantes utilizam urls reais que contém descrições do formato xml
  - Por exemplo, http://www.w3.org/2001/XMLSchema
- Além do atributo xmlns, o elemento raiz schema possui um atributo chamado targetNamespace, o qual permite a definição de um namespace
- Ou seja, o uso deste atributo permite que os elementos definidos neste esquema possam ser referenciados por outros documentos XML

XML

- Definições: XML, Namespace, DTD, XSD
- Vantagens: documento texto, formato extensível, reuso de ferramentas de manipulação
- Desvantagens: documento verboso, manipulação manual ou através de bibliotecas
  - http://xmlsucks.org/

Aplicações: ANT, Xpath, XSLT, RSS, Web Services, SGBDs

Ant

- Ferramenta de *build* feita em Java
- Similar ao *make*, entretanto:
  - Sua entrada é feita utilizando arquivos XML
  - Possui a portabilidade inerente das aplicações baseadas em Java
- É utilizada ou aceita por várias IDE's (Eclipse, NetBeans, JBuilder, WebSphere, ...), para auxílio no processo de compilação
  - Localização da instalação no Eclipse:
    "<eclipse-dir>\plugins\org.apache.ant" ou similar
- Site principal: http://ant.apache.org/index.html

Ant – Arquivo de Entrada Básico – build.xml (1)

```
<?xml version="1.0"?>
<project name="Basico">

  <target name="primeira">
    <mkdir dir="teste" />
  </target>

  <target name="segunda">
    ...
  </target>

</project>
```

Tag principal

Cada projeto é composto de um conjunto de tarefas

Ant – Testando o exemplo

- Em alguma linha de comando posicione no diretório: *“<eclipse-dir>\plugins\org.apache.ant”*
- Digite o seguinte comando:

  ant –f <caminho-build-xml> primeira

- O parâmetro –f indica o caminho do arquivo “build.xml” quando este não está no diretório corrente do Ant
- “primeira” é a tarefa escolhida para ser executada

ANT – Testando o exemplo

- Outra opção é colocar o caminho do Ant no PATH do sistema

Ant – Arquivo de Entrada Básico – build.xml (2)

```
<?xml version="1.0"?>
<project name="Basico" default="segunda">
  <target name="primeira">
    <mkdir dir="teste" />
  </target>
  <target name="segunda" depends="primeira">
    <mkdir dir="teste2" />
  </target>
</project>
```

Tarefa default

Cria dependência entre tarefas

Ant – Exemplo “real”

```
<?xml version="1.0"?>
<project name="Navegador" default="gera-jar" basedir=".">
  <property name="saida" value="jar-navegador" />
  <property name="classes"
value="WorkspaceEclipse\ContaBancaria\br\com\sirius\curso\exemplos" />

  <target name="compilar" description="compilação das classes java">
    <mkdir dir="${saida}" />
    <javac srcdir="${classes}" destdir="${saida}" debug="on"
    optimize="off" deprecation="on" />
  </target>

  <target name="gera-jar" depends="compilar" description="geração do jar">
    <jar destfile="Nav.jar">
      <zipfileset dir="${saida}"/>
      <manifest>
        <attribute name="Main-Class" value="br.com.sirius.curso.exemplos.Principal" />
      </manifest>
    </jar>
    <delete dir="${saida}"/>
  </target>
</project>
```

XPath

- Linguagem para referência a conjunto de nós XML
- Exemplos:
  - /livros/livro/titulo
    - Recupera os títulos de todos os livros
  - /livros/*/titulo
    - Recupera os títulos abaixo de livros que possuem um elemento intermediários (títulos com profundidade 2 à partir do elemento livros
  - /livros/livro[1]
    - Recupera os dados do primeiro livro
  - //livro[autor = 'Kurose']
    - Recupera os livros, em qualquer ponto do documento xml, cujo autor se chama 'Kurose'
  - count(/livros/livro)
    - Retorna a quantidade de livros
  - /livros/livro/@isbn
    - Retorna o valor do atributo isbn dos livros
  - /livros/livro[@qtd-estoque > 10]/titulo
    - Retorna os títulos dos livros cujo atributo qtd-estoque é maior que 10

XPath

- http://www.w3.org/TR/xpath20/
- Esta linguagem possui também um conjunto de funções, como a função count()
- Estas funções manipulam strings, sequências, datas e horas, valores numéricos, funções de agregação, etc.
- Uma lista destas funções pode ser encontrada em http://www.w3schools.com/xpath/xpath_functions.asp

XPath - Exemplos

- **para** matches any *para* element.
- * matches any element.
- **chapter|appendix** matches any *chapter* element and any *appendix* element.
- **olist/entry** matches any *entry* element with an *olist* parent.
- **appendix//para** matches any *para* element with an *appendix* ancestor element.
- **attribute(*, xs:date)** matches any attribute annotated as being of type *xs:date*.
- */* matches a *document node*.
- **document-node()** matches a document node.
- **text()** matches any *text node*.

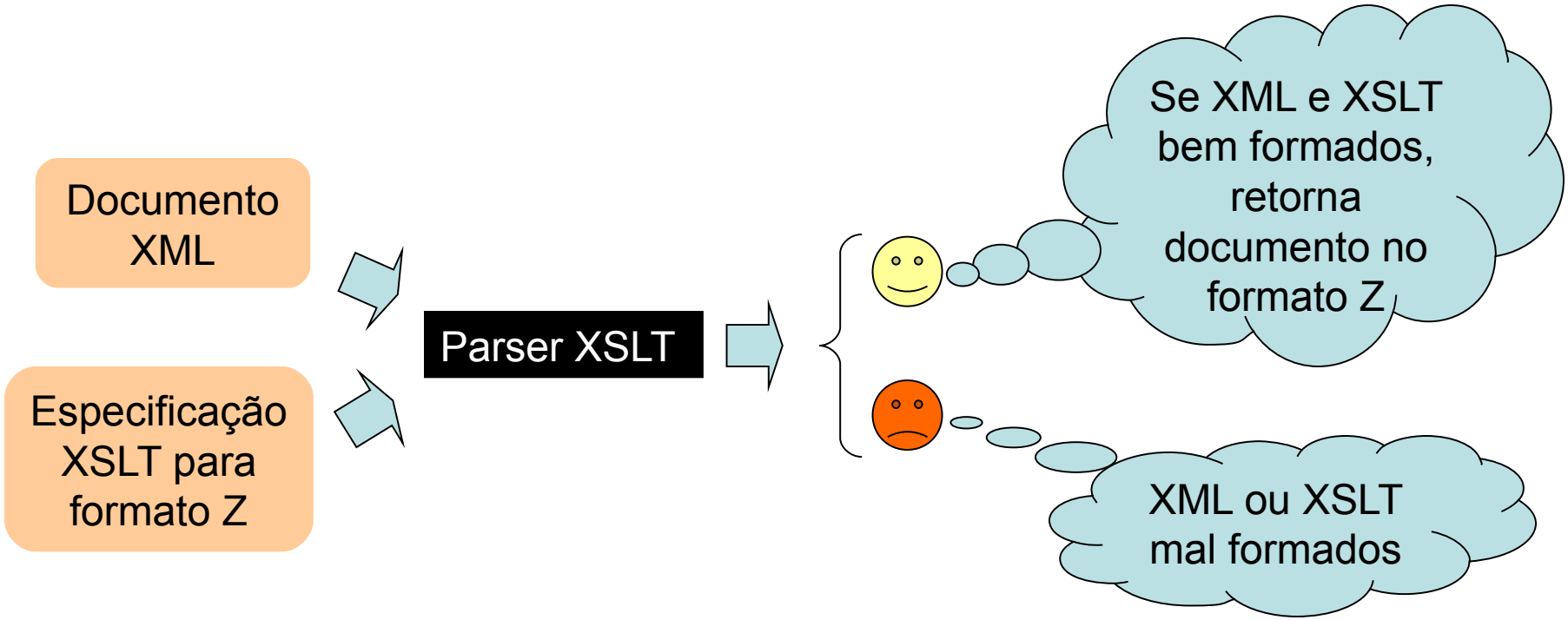
XPath - Exemplos

- **node()** matches any node other than an attribute node, namespace node, or document node.
- **para[1]** matches any *para* element that is the *first para child* element of its parent. It also matches a parentless para element.
- **//para** matches any *para* element that has a parent node.
- **bullet[position() mod 2 = 0]** matches any *bullet* element that is an even-numbered bullet child of its parent.
- **div[@class="appendix"]//p** matches any *p* element with a div ancestor element that has a *class attribute* with value appendix.
- **@class** matches any class attribute (*not* any element that has a class attribute).
- @* matches any attribute node.

XSLT

- XSLT significa *Extensible Stylesheet Language Transformation* (*http://www.w3.org/TR/xslt20/*)
- XSLT permite a transformação de um documento XML num documento em algum outro formato
- Também é definida utilizando o formato XML
- Utiliza a linguagem XPath para fazer referência aos elementos de um documento XML

XSLT


Documento XML
Especificação XSLT para formato Z
Parser XSLT
Se XML e XSLT bem formados, retorna documento no formato Z
XML ou XSLT mal formados

Especificação XSLT

- Uma especificação em XSLT se inicia com a seguinte declaração:

  <xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform“>

  </xsl:stylesheet>

- Uma especificação consiste de uma série de regras que são chamadas de *templates* <xsl:template></xsl:template>
- Um template contém ações a serem tomadas quando um tipo de nó especificado é encontrado ou um nome é fornecido

  <xsl:template match=“...” name=“...”>

  </xsl:template>

Especificação XSLT

- Há pelo menos 2 estilos de especificações XSLT: declarativo e não-declarativo
- Na declarativo, definimos que ações devem ser tomadas para cada tipo de nó encontrado

```
<xsl:template match="livro">
        <tr>
                <td><xsl:value-of select="titulo"/></td>
                <td><xsl:value-of select="autor"/></td>
        </tr>
</xsl:template>
```

Especificação XSLT

- Na especificação declarativa, quando o valor do atributo match é “/” (raíz do documento em XPath), este template é chamado pelo parser XSLT no início da transformação
- Caso o atributo match seja de algum outro tipo, sua chamada é feita pelo parser após alguma chamada ao elemento

<xsl:apply-templates />

- No modo declarativo é importante que todos os tipos de nós existentes sejam tratados; do contrário, estes podem ser exibidos como texto da maneira como aparecem no documento XML

Especificação XSLT

- Na especificação não-declarativa, cada template é explicitamente chamado através do elemento

  <xsl:call-template name=“”>

  </xsl:call-template>

- Na chamada explícita de um template, o template acionado conterá o contexto (posição na árvore XML) onde onde o elemento *call-template* ocorrer

Especificação XSLT

```
<xsl:for-each select="//livro[editora = 'Campus']">
        <xsl:call-template name="tratalivro"/>
</xsl:for-each>
```

```
<xsl:template name="tratalivro">
        <tr>
                <td><xsl:value-of select="titulo"/></td>
                <td><xsl:value-of select="autor"/></td>
        </tr>
</xsl:template>
```

- O template “tratalivro” terá como contexto cada livro encontrado no documento XML (cuja editora é Campus)

Especificação XSLT

- Alguns elementos usados com freqüência em especificações XSLT:
  - <xsl:value-of select="..."/> → imprime o resultado de uma consulta XPath (valor do atributo *select*) na saída
  - <xsl:for-each select="..."> ... </xsl:for-each> → permite a iteração por um conjunto de nós selecionados por uma consulta XPath (valor do atributo *select*)
  - <xsl:if test="..."> ... </xsl:if> → condicional, cujo atributo *test* é uma expressão booleana que pode envolver valores do documento XML

Especificação XSLT

<xsl:choose>
  <xsl:when test=“ ... "> ... </xsl:when>
  <xsl:when test=“ ... "> ... </xsl:when>
  <xsl:otherwise> ... </xsl:otherwise>
</xsl:choose> → Análogo ao switch

- <xsl:sort select=“”/> → permite que o XML seja percorrido de forma ordenada por algum elemento do documento XML

```
<xsl:for-each select="//livro[editora = 'Campus']">
        <xsl:sort select=“autor"/>
        <xsl:call-template name="tratalivro"/>
</xsl:for-each>
```

Especificação XSLT

- <xsl:variable name="..." select="..."/> → guarda o valor de uma consulta XPath (valor do atributo *select*) numa variável com nome *name*
- <xsl:include href="..." /> → faz inclusão de um arquivo de transformação em outro, permitindo, por exemplo, que este último chame um template do primeiro
- <xsl:for-each-group select="..." group-by="..." /> → itera sobre grupos, onde o atributo *select* indica os elementos algo para iteração e o atributo *group-by* indica o campo chave para agrupamento

Especificação XSLT

```
<xsl:for-each-group select="livros/livro" group-by="editora">
<tr>
        <td><xsl:value-of select="current-grouping-key()"/>
        </td>
        <td><xsl:value-of select="sum(current-group()/preco)
div count(current-group())"/>
        </td>
</tr>
</xsl:for-each-group>
```

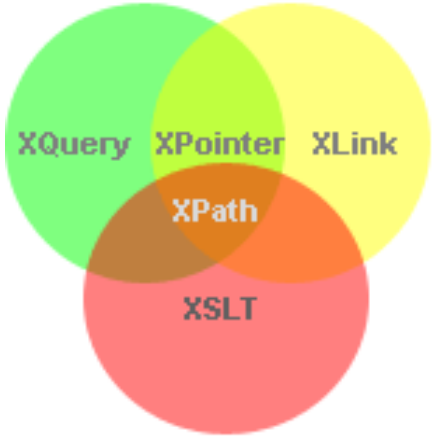
XSLT e Java

```
import java.io.*;
import javax.xml.transform.*;
import javax.xml.transform.stream.*;
public class TransformacaoExemplo {
public static void main(String[] args) {
        // Create transformer factory
         TransformerFactory factory = TransformerFactory.newInstance();
        // Use the factory to create a template containing the xsl file
         Templates template;
        try {
                template = factory.newTemplates(new StreamSource(
new FileInputStream("rss.xslt")));
                // Use the template to create a transformer
                Transformer xformer = template.newTransformer();
                // Prepare the input and output files
                Source source = new StreamSource(new
FileInputStream("plantaociencia.xml"));
                Result result = new StreamResult(new
FileOutputStream("plantaociencia.html"));
                // Apply the transformation
                xformer.transform(source, result);
} catch ...
}
```

XQuery

- Linguagem projetada para consultar dados XML
- XQuery está para XML assim como SQL está para um BD
- Não segue a sintaxe de XML
- Alternativa para o uso de XSLT
- Também se baseia na linguagem

XPath 2.0



- Uso com Java: http://www.ibm.com/developerworks/library/x-xjavaxquery/index.html

Ex. de Processamento XSLT - RSS

```
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
  <xsl:output method="html"/>
  <xsl:template match="rss">
    <html>
      <head><title>
        <xsl:value-of select="channel/title"/>
      </title></head>
      <xsl:apply-templates/>
    </html>
  </xsl:template>
  <xsl:template match="channel/title">
    <h1><a href="{../link}"><xsl:apply-templates/></a></h1>
  </xsl:template>
  <xsl:template match="title">
    <h2><a href="{../link}"><xsl:apply-templates/></a></h2>
  </xsl:template>
  <xsl:template match="description">
    <p><xsl:apply-templates/></p>
  </xsl:template>
  <xsl:template match="pubDate | link | language"/>
</xsl:stylesheet>
```

Ex. de Processamento XQuery - RSS

UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

```
xquery version "1.0";
<result>
 {
  for $i in doc("plantaociencia.xml")/rss/channel/item
  where starts-with($i/pubDate/text(),"Wed")
  return
    <friday>
      { $i/title/text() }
    </friday>
 }
 {
  for $i in doc("plantaociencia.xml")/rss/channel/item
  where contains($i/title/text(),"XMI")
  return
    <xmi>
      { $i/title/text() }
    </xmi>
 }
</result>
```

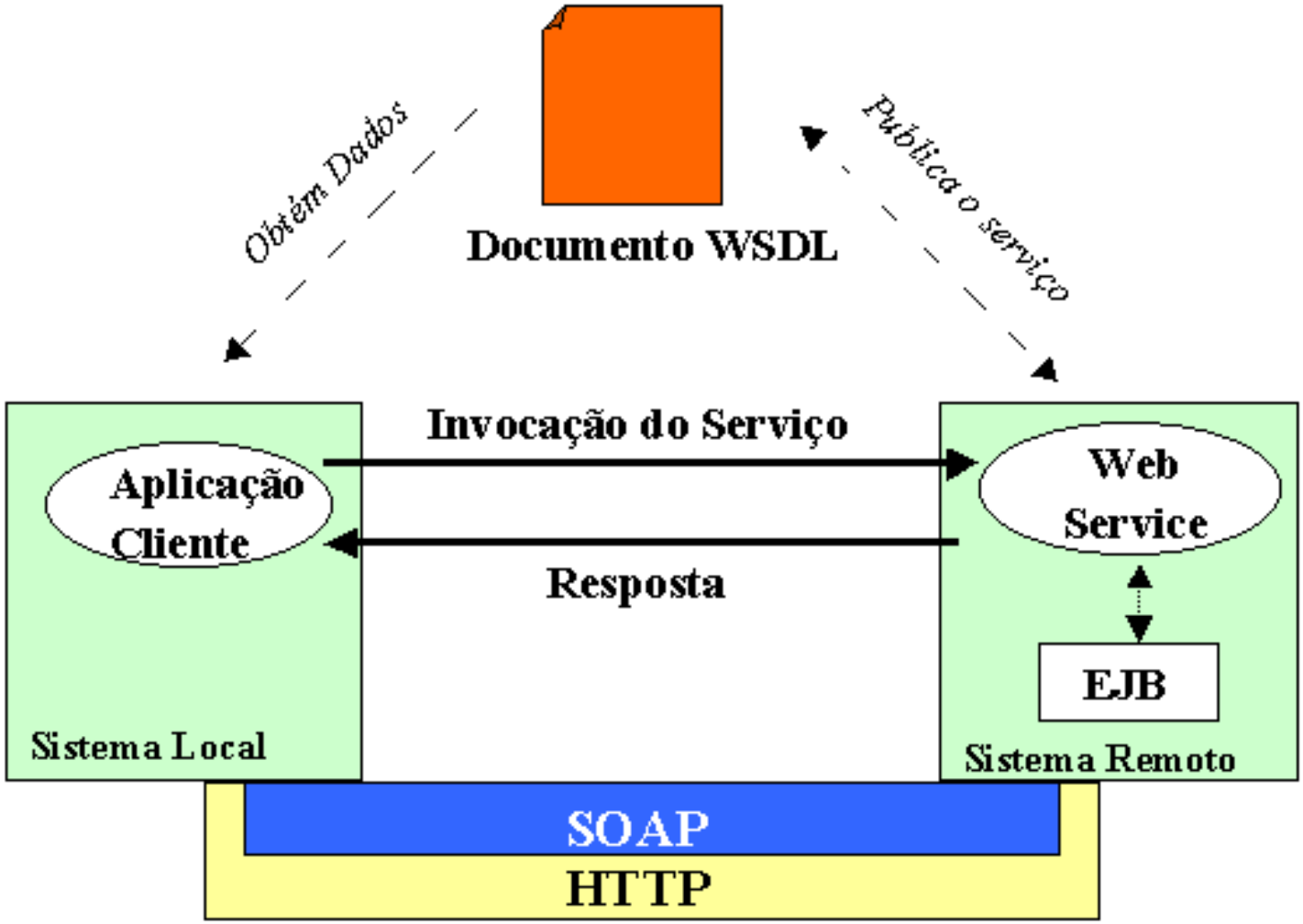
Diferenças entre XSLT e XQuery

- São igualmente capazes, principalmente por se basearem em XPath
- XSLT foi projetada pensando-se em trabalhar com todo o xml; além disso, seu parser possui um comportamento padrão, o qual é iterar por todo o documento
- XSLT não é fortemente tipada
- XQuery, por sua vez, foi projetada pensando em manipular seções de um xml
- XQuery não possui comportamento padrão
- XQuery é fortemente tipada e não é baseada na sintaxe XML
- http://www.ibm.com/developerworks/xml/library/x-wxxm34.html

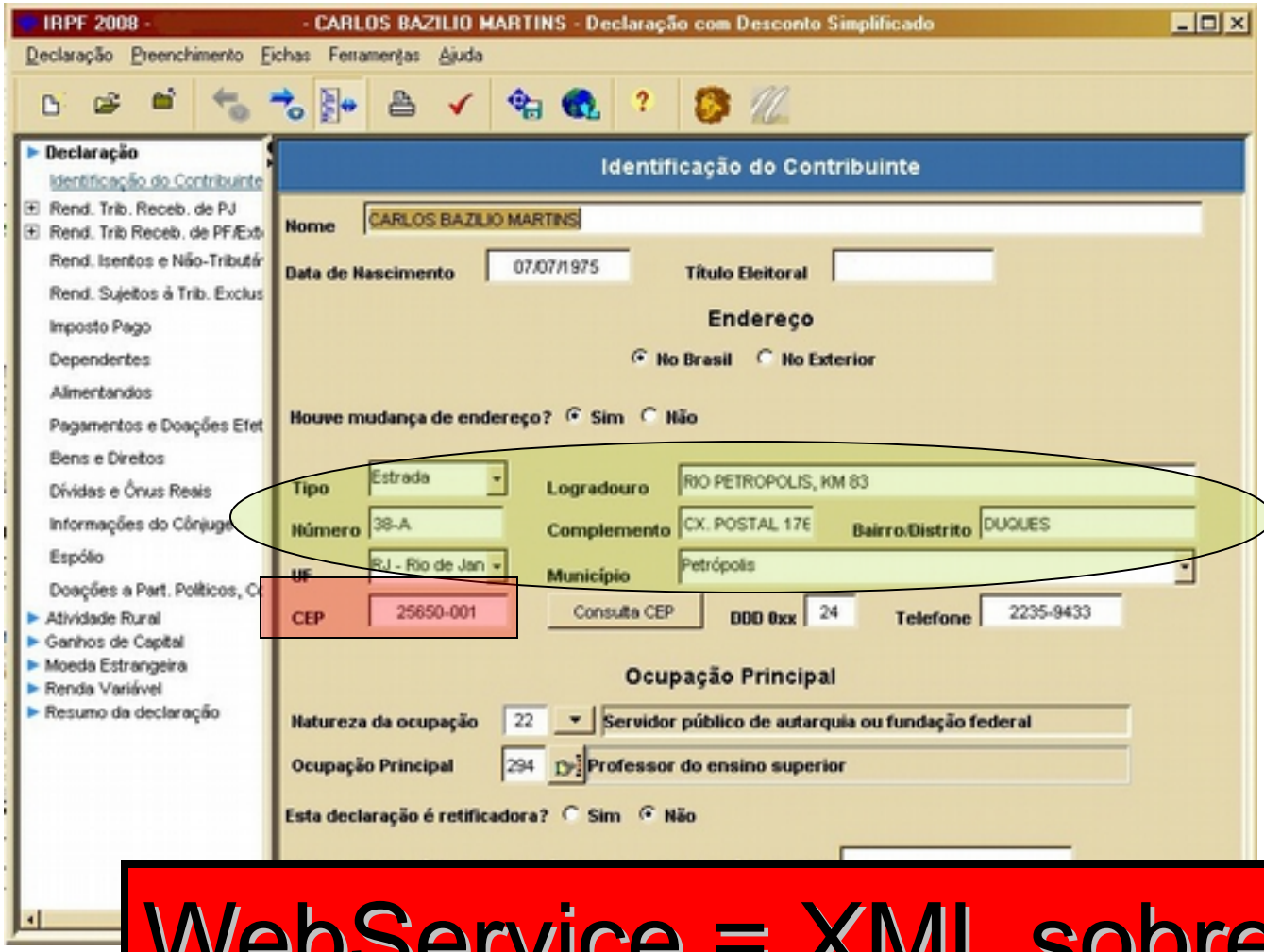
Web Services

- Proposta:
  - Integração de sistemas através do uso de XML sobre HTTP
- Tecnologias:
  - XML
  - WSDL (Web Service Description Language)
  - SOAP (Simple Object Access Protocol)
- Vantagens:
  - Integração com baixo acoplamento
  - Independe das linguagens de implementação

Web Services



Exemplo de Integração usando XML



XML (CEP)

CORREIOS

XML (Dados Postais)

WebService = XML sobre HTTP

XML e SGBDs

- A proliferação do formato XML estimulou o surgimento de SGBDs com suporte a XML
- Entendemos como suporte a capacidade do SGBD de:
  - gerar e receber dados no formato XML
  - entender algumas derivações do XML, como XQuery
  - abdicar de trabalhar com o modelo relacional para trabalhar com XML (modelo em árvore); este o caso dos chamados NXD (*Native XML Databases*)
- Site sobre o assunto: http://www.rpbourret.com/xml/XMLAndDatabases.htr

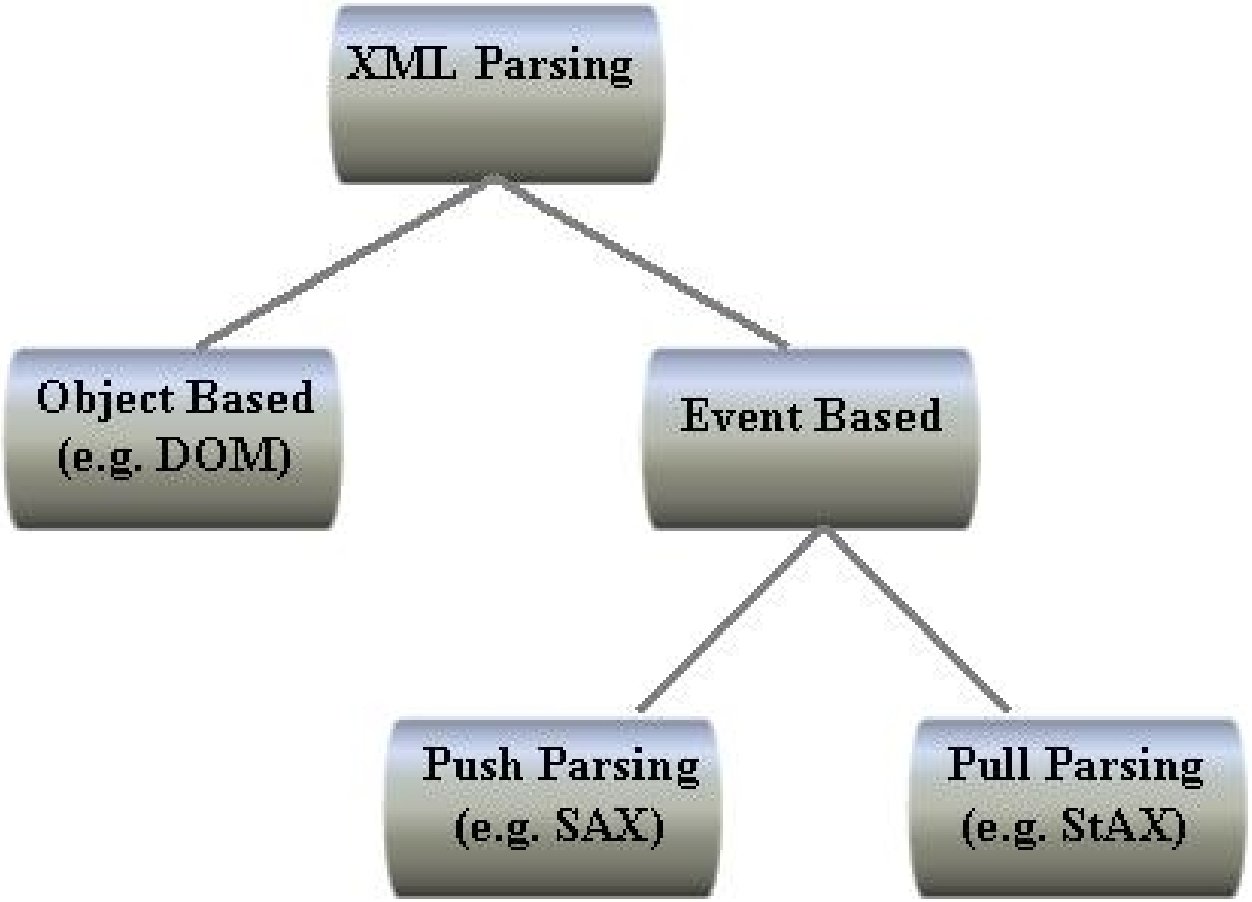
XML e SGBDs

- Alguns produtos:
  - eXist (http://exist.sourceforge.net/): SGBD de código aberto inteiramente baseado no modelo XML
  - DB2 pureXML (http://www-01.ibm.com/software/data/db2/xml/): SGBD relacional e baseado no modelo XML
  - DB2 Express-C (http://www-01.ibm.com/software/data/db2/express/): Versão limitada e gratuita do pureXML
  - Tamino (http://www.softwareag.com/Corporate/products/wm/tamino): Um dos primeiros SGBDs lançados baseados em XML

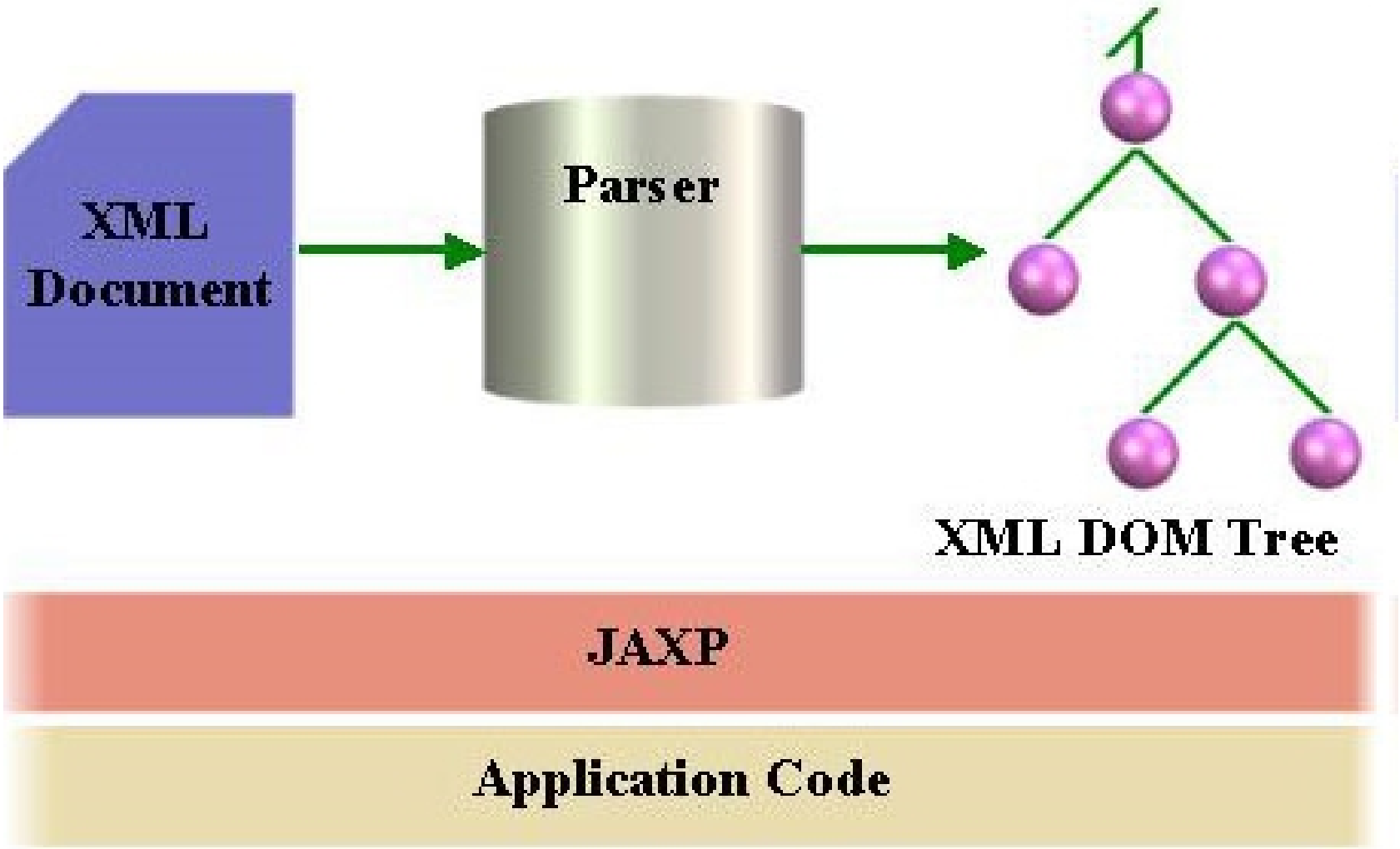
Manipulando XML em Java

- JAXP (*Java API for XML Processing*) é a API mais comumente utilizada para processamento de documentos XML
- Entenda como *processamento* a possibilidade de leitura e geração de xml, busca por elementos, transformação, etc
- Este processamento não é implementado efetivamente pela API; ou seja, a JAXP funciona como um *wrapper* para processadores XML de terceiros (não está amarrada a nenhum fabricante)
- Entretanto, há limitações quanto ao suporte atual (próximo slide)

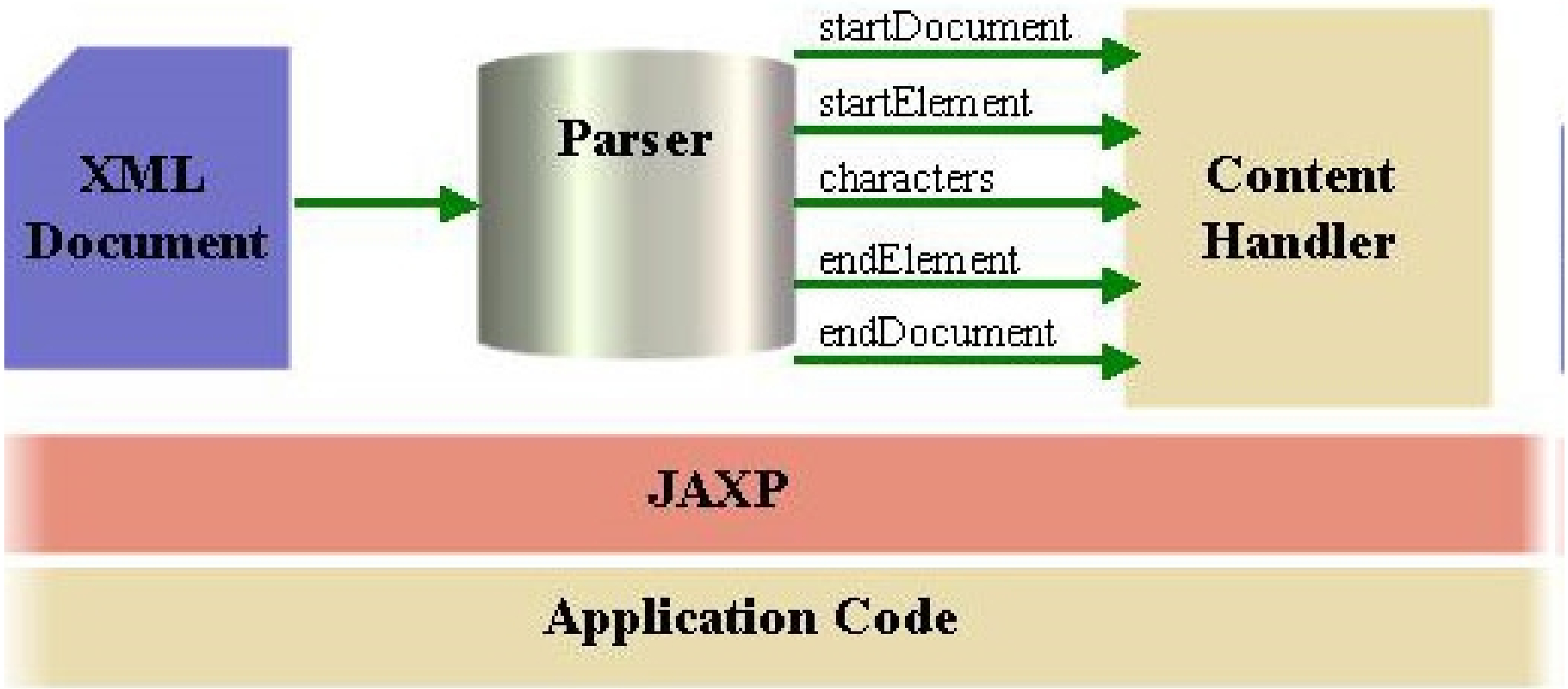
Manipulando XML em Java



Manipulando XML em Java
DOM



Manipulando XML em Java
SAX



Outras APIs para Uso de XML em Java

- JDOM (http://www.jdom.org/)
- dom4j (http://www.dom4j.org/)
- XOM (http://www.xom.nu/)
- XStream (http://xstream.codehaus.org/)
  - Difere dos anteriores por permitir a geração de documentos XML à partir de estruturas de classe e vice-versa

Referências

- http://www.w3schools.com/
  - Site com tutoriais on-line rápidos e com muita qualidade
- http://del.icio.us/carlosbazilio/xml
  - Meus favoritos sobre o assunto
- http://www.xml.com/
  - Site com diversas informações sobre a linguagem
- http://www.w3.org/
  - Site do consórcio W3C
- http://www.xml.com/pub/a/2005/07/06/jaxp.html
  - Artigo sobre processamento de XML utilizando Java
- http://www.cafeconleche.org/books/xmljava/
  - Livro online de XML usando Java