



QUEM SOMOS

CURSOS

LIVROS

FÓRUM

SIMULADOS

BLOG

MATERIAIS

FALE CONOSCO



XML



PROFESSOR MARCELO PACOTE – MARCELOPACOTE@DOMINANDOTI.COM.BR

WWW.DOMINANDOTI.COM.BR



ACESSE NOSSO SITE EM WWW.DOMINANDOTI.COM.BR

QUEM SOMOS

CURSOS

LIVROS

FÓRUM

SIMULADOS

BLOG

MATERIAIS

FALE CONOSCO

Cursos Turmas em Brasília, na sua cidade, e cursos online
Livros Edições publicadas, lançamentos e promoções
Fórum Interação direta entre estudantes e com os professores

Simulados Questões inéditas, ranking de notas e correções em vídeo

Blog Dicas e macetes de estudo, indicações de bibliografia, etc.

Materiais Versões atualizadas de notas de aula e listas de exercícios



Curta o Dominando TI no **facebook**
e receba nossas dicas sobre concursos!



ESTRATÉGIA DO CURSO

- Esquema de apresentação
 - Referencial teórico → exercícios
 - Questões de concursos anteriores do Cespe, especialmente de 2013 e 2012.
 - Quando pertinente, questões eventuais de outras bancas.
- Exercícios complementares



SOBRE O PROFESSOR

- Marcelo Pacote
 - Auditor Federal de Controle Externo (TCU/2005)
 - Mestre em Informática pela UnB
 - 14 anos de experiência em desenvolvimento de *software*
 - Certificações: SOACP, PMP, CTFL, CSM, SCJA, SCJP, SCJD, SCWCD, SCBCD, SCEA (I), RUPF, IRUP, ITIL Foundation, Oracle SQL Expert



AGENDA – DESENVOLVIMENTO DE SISTEMAS

- AULA 1: XML
- AULA 2: Web Services
- AULA 3: SOA + XSLT
- AULA 4: Java OO
- AULA 5: Java EE +Java (recursos avançados)
- AULA 6: Servlets + Ajax
- AULA 7: Padrões de Projeto
- AULA 8: JSF + Hibernate
- AULA 9: Raciocínio lógico (lógica de programação).



XML – eXtensible Markup Language

- Voltado para a troca de informações
 - Formato para descrever dados estruturados
 - Transporte e armazenamento de dados
- É independente de software e hardware
- Baseada em SGML



XML – eXtensible Markup Language

- Permite adicionar contexto e significado à informação transmitida.
- Mantido pelo W3C.



HTML x XML

XML

```
<contato>  
  <nome>Senor Abravanel</nome>  
  <email>silvio@sbt.com.br</email>  
  <telefone>  
    <ddd>11</ddd>  
    <numero>8116-9977</numero>  
  </telefone>  
</contato>
```

Significado

- XML não é um melhoramento de HTML e sim uma mudança de conceito.
- Ao contrário de HTML, o conjunto de tags não é pré-definido

HTML

```
<h1>Senor Abravanel</h1>  
<h2>silvio@sbt.com.br</h2>  
<p>  
  <b>11</b>  
  <i>8116-9977</i>  
</p>
```

Apresentação



Com relação a linguagens de programa, julgue os itens que se seguem.

() O XML foi projetado para transportar e armazenar dados, enquanto o HTML foi projetado para mostrar dados com foco na sua aparência. Os *tags* predefinidos do XML são similares aos do HTML, mas com aplicações diferentes.



POR QUE XML É IMPORTANTE?

- Não depende de aplicação, linguagem ou formato para ser processado.
 - padrão “de fato”, mantido pelo W3C.
 - baseado em texto, com suporte a unicode
- Padroniza a troca de informações entre sistemas e aplicações.
 - Provê maior agilidade para a troca de dados.



PARTES DE UM DOCUMENTO XML

elemento raiz declaração XML

```
<?xml version="1.0" encoding="iso-8859-1"?>
<usuario>
  <foto href="/imagens/logoduke2.gif" />
  <nome>Duke Java</nome>
  <endereco>Rua Imaginária, 20 – 4º andar – Deodoro da Fonseca – RN</endereco>
  <email>dukejv@script.com.br</email>
  <telefone tipo="residencial">
    <ddd>84</ddd>
    <numero>32562589</numero>
  </telefone>
</usuario>
```

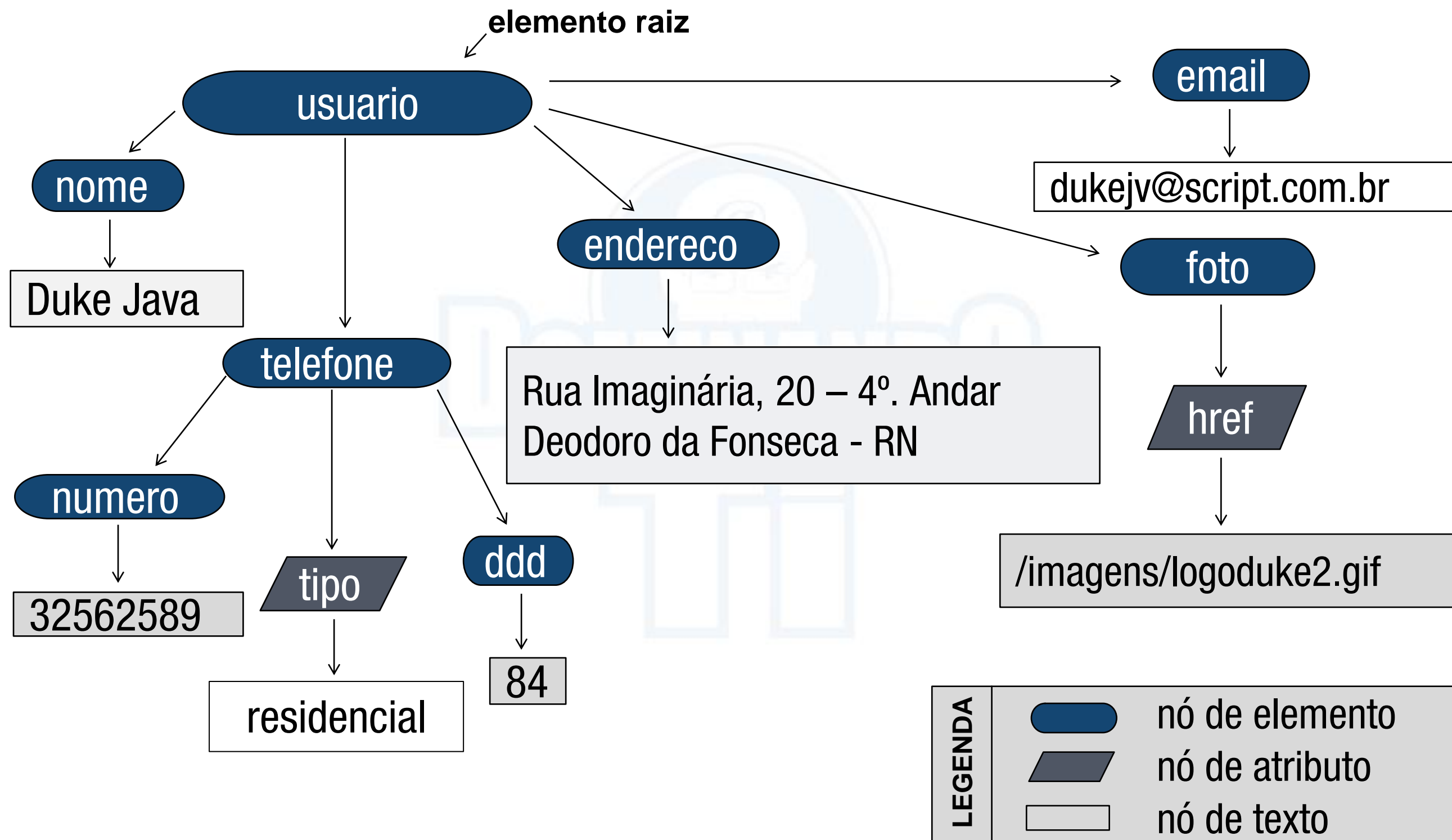
atributos

elementos

The diagram illustrates the components of an XML document. It shows a root element <usuario> containing several child elements. Arrows point from labels to specific parts of the XML code: 'elemento raiz' points to <?xml>, 'declaração XML' points to the version and encoding attributes, 'atributos' points to the href attribute of the <foto> element and the tipo attribute of the <telefone> element, and 'elementos' points to the <nome>, <endereco>, <email>, <ddd>, and <numero> elements.



ÁRVORE XML



(ANAC/2012/CESPE/ANALISTA ADM./85) - ADAPTADA

() Um arquivo XML possui atributos e elementos. No exemplo 1, que se segue, sexo é atributo e, no exemplo 2, sexo é elemento, provendo, em ambos os exemplos, a mesma informação.

Exemplo 1:

```
<peessoa sexo="F">  
  <nome>Dhara</nome>  
  <sobrenome>Silva</sobrenome>  
</peessoa>
```

Exemplo 2:

```
<peessoa>  
  <sexo>F</sexo>  
  <nome>Dhara</nome>  
  <sobrenome>Silva</sobrenome>  
</peessoa>
```



DOCUMENTOS XML BEM FORMADOS

- Para que possa ser manipulado como uma árvore, um documento XML precisa ser bem formado.
- Regras
 - Ter um, e apenas um, elemento raiz
 - Todos os elementos devem ter uma tag de fechamento
 - Elementos deve estar corretamente aninhados
 - Valores dos atributos devem estar entre aspas ou apóstrofes. Atributos não devem se repetir em um mesmo elemento.
 - XML diferencia caracteres maiúsculos de minúsculos (case sensitive)
- O próprio navegador pode ser utilizado para verificar as regras.
 - Há validadores online (http://w3schools.com/xml/xml_validator.asp)



() A expressão XML mostrada abaixo está correta.

```
<?xml version="1.0"=??>
```

```
<registro>
```

```
  <nome idade=29>carlos</nome>
```

```
  <sobrenome>barbosa</sobrenome>
```

```
</registro>
```



(TRE-MS/2013/CESPE/ANALISTA JUDICIÁRIO/44) (1/2)

```
1 <!xml version=1.0 encoding=ISO-8859-1!>
2 <Concessionaria>
3 <carro>
4 <marca>Hyundai</marca>
5 <nome>sonata</nome>
6 <motor potencia=2.0>
7 </carro>
8 <carro>
9 <marca>Chevrolet</marca>
10 <nome>Camaro</Nome>
11 <motor potencia=4.0>
12 <concessionaria>
```

Acima, está mostrado o conteúdo total de um arquivo XML, em que os números à esquerda indicam apenas as linhas em que informações são apresentadas e não fazem parte do conteúdo do arquivo. Tendo como referência essas informações, assinale a opção correta, à luz dos padrões XML.



(TRE-MS/2013/CESPE/ANALISTA JUDICIÁRIO/44) (2/2)

- A) Há erro nas linhas 4 e 9, pois não se pode repetir uma tag, no caso, *<marca>*.
- B) Há erro nas linhas 6 e 11, pois valores de atributos devem ficar entre aspas.
- C) Por não haver outra tag igual até o final do documento, não é necessário fechar a tag *<carro>* introduzida na linha 8.
- D) A sintaxe para a descrição da versão e da codificação estão corretamente definidas na linha 1.
- E) As *tags* em XML podem ser maiúsculas ou minúsculas sem distinção, tal como utilizado nas linhas 2 e 12.



() Em um documento XML bem formado, os elementos pertinentes podem estar entrelaçados, desde que estejam aninhados.



XML NAMESPACES - MOTIVAÇÃO

```
<table>
  <tr>
    <td>ABIN</td>
    <td>TCU</td>
  </tr>
</table>
```


```
<table>
  <name>mesa madeira</name>
  <width>80</width>
  <length>120</length>
</table>
```



XML NAMESPACES

- Limita o escopo de elementos
 - Forma simples e direta de diferenciar nomes usados em documentos XML. Evita conflitos de nomes.
- Consiste da associação de um identificador a cada elemento/atributo da linguagem, que pode ser herdado (namespace default) ou atribuído explicitamente através de um prefixo.

- Exemplo


<cadastro xmlns:firma="01.234.567/0001-99">

<nome>Maria Joaquina</nome>

<firma:nome>Maria Doces Ltda.</firma:nome>

<email>doces@maria.com.br</email>

</cadastro>

Este elemento <nome> pertence a outro namespace



DECLARAÇÃO DE NAMESPACES

- `xmlns="identificador"` (namespace default)
 - Associa o identificador com todos os subelementos que não possuem prefixo. Ex.: `<nome>`
- `xmlns:prefixo="identificador"`
 - Associa identificador com subelementos e atributos cujo nome local é precedido do prefixo. Ex.: `<prefixo:nome>`
- O escopo da declaração inclui o elemento que contém o atributo `xmlns` e todos os seu descendentes.
- O prefixo é arbitrário e só existe dentro do documento.



NAMESPACES

está associada a este prefixo Esta URI

```
<ct:usuario xmlns:ct="01.234.567/0001-89/dominandoti">  
  <ct:nome>Duke Java</ct:nome>  
  <ct:endereco>  
    Rua imaginária, 20 – Centro – 19920-321 – Deodoro da Fonseca – RN  
  </ct:endereco>  
  <ct:email>dukejv@script.com.br</ct:email>  
  <ct:telefone tipo="residencial">  
    <ct:ddd>84</ct:ddd>  
    <ct:numero>32562589</ct:numero>  
  </ct:telefone>  
</ct:usuario>
```



() Considere que um líder de equipe solicite a um programador que comente o trecho de código de uma página de programação server-side apresentada a seguir:

```
<html xmlns="http://www.w3.org/1999/xhtml"  
xmlns:h="http://java.sun.com/jsf/html"  
xmlns:f="http://java.sun.com/jsf/core"  
xmlns:ui="http://java.sun.com/jsf/facelets"  
xmlns:ez="http://java.sun.com/jsf/composite/ezcomp">
```

Nessa situação, se o programador disser que a página importa exatamente cinco namespaces XML, sendo que o namespace default possui o URI (universal resource identifier) `http://www.w3.org/1999/xhtml`, esse comentário estará correto.



XML VÁLIDO

- Um XML é dito válido quando for bem formado e estiver de acordo com a gramática que define sua estrutura.
- A gramática é definida em um esquema:
 - elementos fazem parte de um vocabulário limitado
 - certos atributos têm valores e tipos definidos
 - elementos são organizados de acordo com uma estrutura hierárquica
- DTD e XML Schema são linguagens para descrição de esquemas XML.
- Um XML Schema está para um modelo de dados de banco de dados assim como um documento XML está para um registro deste banco.



DTD vs. XML Schema

- Soluções padrão do W3C

DTD

```
<!ELEMENT contato (nome, email, telefone)>  
<!ATTLIST contato codigo NMTOKEN #REQUIRED>
```

- Simples mas não é XML
- Não suporta namespaces
- Limitado quanto a tipos de dados

XML Schema

```
<xsd:schema xmlns:xsd=".../XMLSchema">  
  <xsd:element name="contato">  
    <xsd:complexType>  
      <xsd:attribute name="codigo" use="required">
```

- É XML, porém mais complexo
- Suporta namespaces
- Permite definição de tipos



(SERPRO/2013/CESPE/SUP./92) - ADAPTADA

() Para ser considerado válido, um documento XML precisa estar em conformidade com um DTD (document type definition) ou com um XML Schema.



() DTD (document type definition) e XSD (XML schema description) são dois formatos de interoperabilidade de dados usados no escopo do padrão XML, e, de modo geral, um documento DTD é semanticamente menos expressivo que seu equivalente XSD.



DOCUMENTOS VÁLIDOS

- Válidos em relação a um XML Schema contém:
 - Declaração de pelo menos um namespace de aplicação no documento
 - Declaração de namespace padrão da instância XML Schema
 - Atributo schemaLocation (do namespace padrão) associando o(s) namespace(s) de aplicação a um (ou mais) documento(s) XML Schema

<bilhete

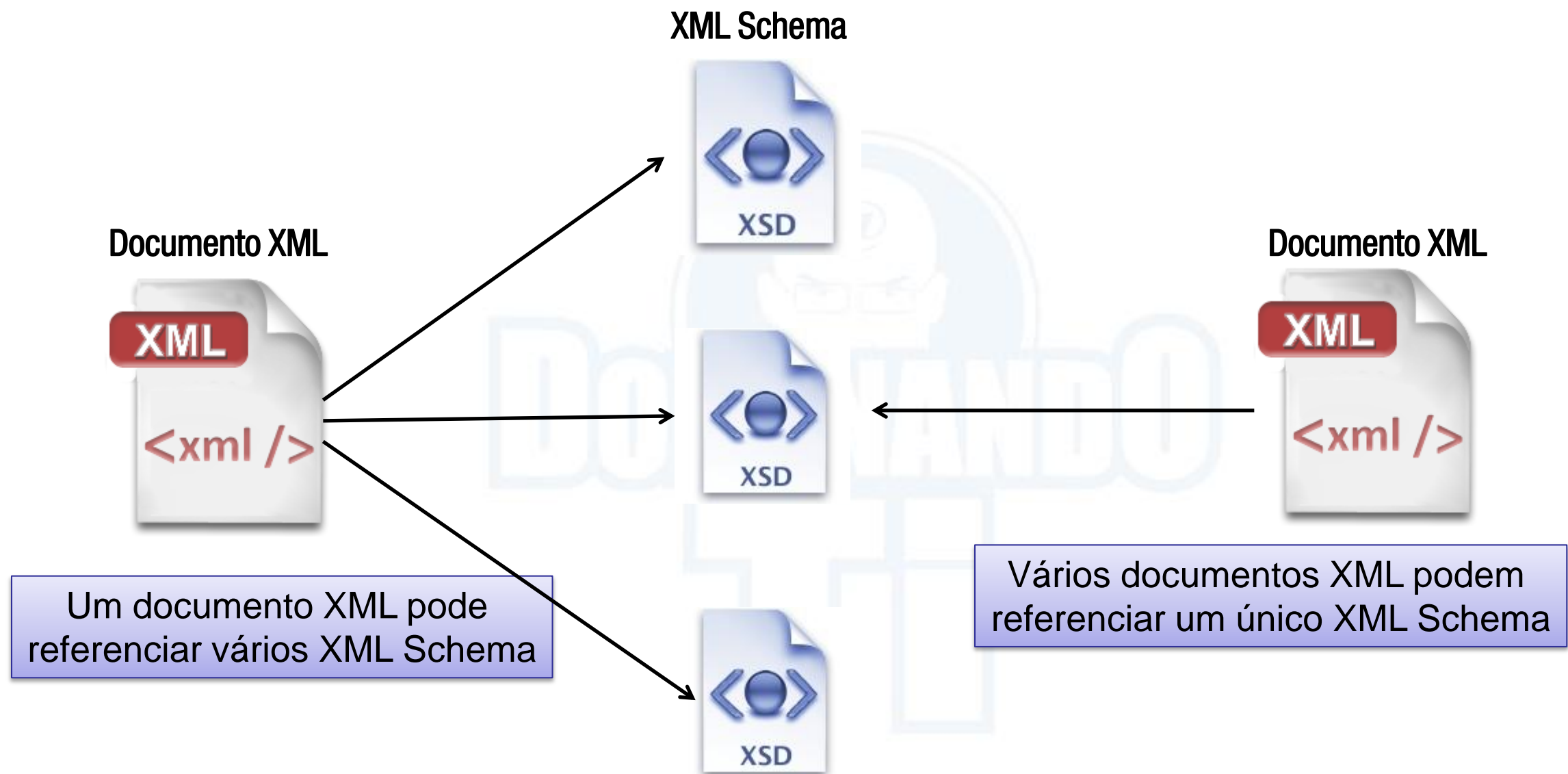
xmlns="urn:123456789"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="urn:123456789 bilhete.xsd">



XML Schema: DEFINIÇÕES



XML SCHEMA: TIPOS

- **Tipos simples** representam tipos de dados básicos como texto, números, tokens, booleanos
 - Fazem parte do namespace do XML Schema (requerem prefixo associado ao identificador do namespace), por exemplo: `xs:int`, `xs:string`
- **Tipos complexos** representam estruturas do documento como entidades, atributos, etc.



XML SCHEMA

- **Tipo Simples**

- não pode conter outros elementos, apenas valor textual

```
<xs:element name="sobrenome" type="xs:string"/>
```

```
<xs:element name="idade" type="xs:integer"/>
```

```
<xs:element name="dataNasc" type="xs:date"/>
```

```
<sobrenome>Zouza</sobrenome>
```

```
<idade>26</idade>
```

```
<dataNasc>1984-11-21</dataNasc>
```



XML SCHEMA – TIPOS COMPLEXOS

<xs:element name="empregado">

<xs:complexType>

<xs:sequence>

Indicador de ordem

<xs:element name="nome" type="xs:string"/>

<xs:element name="sobrenome" type="xs:string"

maxOccurs="10" minOccurs="0" />

Indicador de ocorrência

</xs:sequence>

</xs:complexType>

</xs:element>

<empregado>

<nome>Marcelo</nome>

<sobrenome>Pacote</sobrenome>

</empregado>



Com relação aos conceitos e aplicações de gestão eletrônica de documentos, julgue os itens a seguir.

() Em um documento XML, não é possível definir regras de tipos de dados para elementos e atributos, uma vez que o documento é um arquivo do tipo texto.



XML SCHEMA

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.xml.com"
xmlns="http://www.xml.com" elementFormDefault="unqualified">
<xs:element name="nota">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="de" type="xs:string" />
      <xs:element name="para" type="xs:string" />
      <xs:element name="titulo" type="xs:string" />
      <xs:element name="descricao" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```



() Uma especificação em XML Schema é sempre iniciada com tag `<schema>` e concluída com tag `</schema>`. Todas as declarações de elementos devem ser inseridas entre as duas tags, bem como a definição de atributos e tipos deve ser inserida no corpo do programa.



(SENADO/2008/FGV/AN. SISTEMAS/45 A 47) - ADAPTADO

- Para as três questões a seguir, sejam o documento XML e seu correspondente XML Schema:



```
<?xml version="1.0"?>
<OrdemdeCompra xmlns="http://xyz.org/oc.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xyz.org oc.xsd" datacompra="20-12-2000">
  <enddestino pais="BRASIL">
    <nome>Luis Potata</nome>
    <rua>Rua Torta 423</rua>
    <cidade>Cintra</cidade>
    <estado>SP</estado>
    <cep>90952023</cep>
  </enddestino>
  <endpagamento pais="BRASIL">
    <nome>Julia Pombal</nome>
    <rua>Rua Silvano 30</rua>
    <cidade>Pirara</cidade>
    <estado>PA</estado>
    <cep>76889043</cep>
  </endpagamento>
  <comentario>Esta compra é urgente!</comentario>
  (...)
</OrdemdeCompra>
```



```

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://xyz.org/oc.xsd" xmlns="http://xyz.org/oc.xsd"
elementFormDefault="qualified">
  <xs:element name="OrdemdeCompra" type="TipoOrdemdeCompra"/>
  <xs:element name="comentario" type="xs:string"/>
  <xs:complexType name="TipoOrdemdeCompra">
    <xs:sequence>
      <xs:element name="enddestino" type="endereco"/>
      <xs:element name="endpagamento" type="endereco"/>
      <xs:element ref="comentario" minOccurs="0"/>
      <xs:element name="itens" type="Itens"/>
    </xs:sequence>
    <xs:attribute name="datacompra" type="xs:date"/>
  </xs:complexType>
  <xs:complexType name="endereco">
    <xs:sequence>
      <xs:element name="nome" type="xs:string"/>
      <xs:element name="rua" type="xs:string"/>
      <xs:element name="cidade" type="xs:string"/>
      <xs:element name="estado" type="xs:string"/>
      <xs:element name="cep" type="xs:decimal"/>
    </xs:sequence>
    <xs:attribute name="pais" type="xs:NMTOKEN" fixed="BRASIL"/>
  </xs:complexType>
  (...)
</xs:schema>

```



É correto afirmar que, no documento XML:

- a) o elemento *rua* poderia anteceder o elemento *nome* em *enddestino*.
- b) o elemento *comentario* pode aparecer mais de uma vez.
- c) *http://xyz.org/oc.xsd* é o namespace padrão.
- d) o atributo *datacompra* não deveria estar dentro da tag de abertura do elemento *OrdemdeCompra*.
- e) os elementos *enddestino* e *endpagamento* não podem ter um atributo com mesmo nome.



Em relação ao documento XML Schema é correto afirmar que:

- a) se *minOccurs="0"* fosse removido da declaração do elemento *comentario*, então o documento XML deveria ter pelo menos uma ocorrência desses elementos.
- b) o elemento *comentário* é um tipo simples por não ter um atributo *type* associado.
- c) os elementos *nome*, *rua*, *cidade*, *estado* e *cep* não poderiam ser declarados diretamente como subelementos dos elementos *endpagamento* e *enddestino* em lugar da declaração através do *type endereço*.



- d) a declaração `<xs:attribute name="datacompra" type="xs:date"/>` deveria anteceder a declaração do tipo complexo `TipoOrdemdeCompra`.
- e) no documento XML, os elementos em que *minOccurs="0"* não podem ter qualquer ocorrência.



Avalie as afirmativas a seguir:

- I. A declaração *elementFormDefault="qualified"* torna obrigatório o uso de tipos complexos no esquema.
- II. A declaração *targetNamespace="http://xyz.org/oc.xsd"* indica que os elementos e tipos de dados usados no XML Schema (schema, element, complexType, sequence etc.) vêm do namespace *http://xyz.org/oc.xsd*.
- III. Um elemento do tipo complexo pode se basear em um tipo complexo existente e ainda adicionar alguns elementos.



(SENADO/2008/FGV/AN. SISTEMAS/47) (2/2)

- a) se somente as afirmativas I e II estiverem corretas.
- b) se somente as afirmativas II e III estiverem corretas.
- c) se somente a afirmativa III estiver correta.
- d) se somente a afirmativa II estiver correta.
- e) se somente as afirmativas I e III estiverem corretas.



Acerca do XML, julgue os itens a seguir.

() Em XML, as tags definem elementos de dados e o texto fornece o dado real representado no documento.

() A sintaxe básica para um elemento XML pode ser corretamente representada pela instrução a seguir.

`<nome_do_elemento>Texto</nome_do_elemento>`

() Um documento XML pode conter definições para o elemento raiz e para os elementos filhos, podendo também conter elementos vazios.



(TRE-BA/2010/CESPE/ANALISTA JUD./84 A 89) (2/2)

() A instrução a seguir está sintaticamente correta e permite o uso de algarismos romanos para codificação de números.

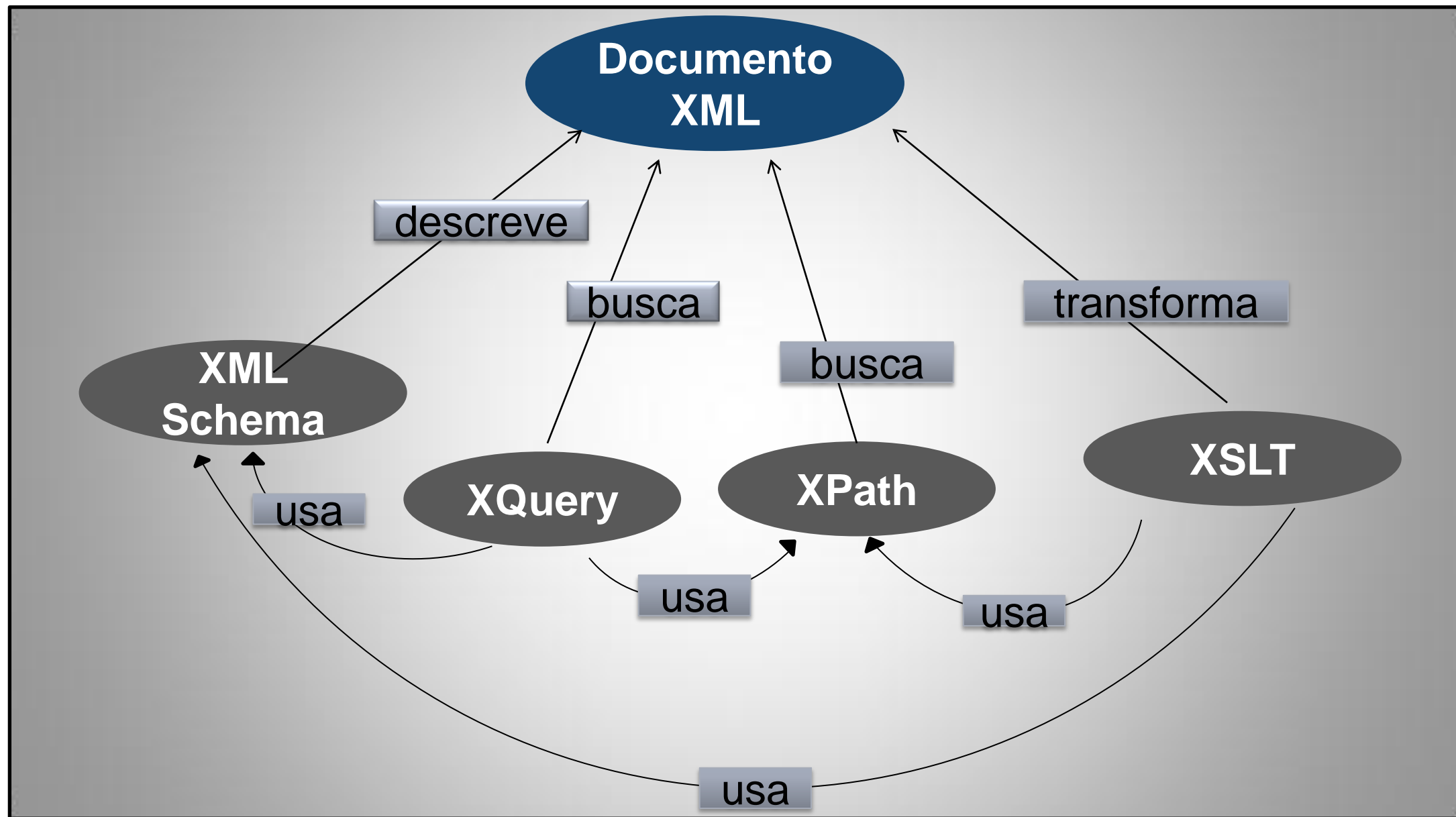
<?xml version="1.0" encoding="ISO-8859-1"?>

() As marcações XML não fazem distinção entre letras minúsculas e maiúsculas.

() Um documento XML sempre deve ter um elemento principal, também conhecido como root tag.



TECNOLOGIAS XML



Fonte: SOA Systems Inc. (www.arcitura.com)

- Outras tecnologias da família: XLink, Xpointer, XHTML, SVG



FORMAS DE PROCESSAMENTO XML

- Via APIs de programação (independentes de linguagem e plataforma)
 - SAX (*Simple API for XML*): leitura sequencial. Ideal para extração de dados.
 - visão baseada em eventos.
 - DOM (*Document Object Model*): leitura completa. Ideal para manipulação (inserção, reordenação, alteração, remoção de nós); consome mais memória.
 - visão baseada em árvore.



() Na linguagem XML, todo atributo é parte de um elemento, todo elemento é raiz ou filho de uma raiz, a construção de uma árvore pode empregar o modelo DOM, uma transformação pode ser direcionada por um documento XSLT, e quando se deseja consumir pouca memória no processamento de XML pode-se empregar um parser do tipo SAX.



GABARITO

(TRT-BA/2008/Cespe/An. Judiciário/66) errado
(ANAC/2012/Cespe/Analista Adm./85) c
(MPU/2013/Cespe/Desv./104) e
(TRE-MS/2013/Cespe/Analista Judiciário/44) b
(STF/2013/Cespe/Tec./104) e
(TCU/2010/CESPE/aufc/136) certo
(Serpro/2013/Cespe/Sup./92) certo
(Basa/2012/Cespe/Prod. e Infra/111) certo
(SERPRO/2013/Cespe/Analista/116) e
(TRT/2013/Cespe/Téc. Judiciário/94) e
(Senado/2008/FGV/an. sistemas/45) c
(Senado/2008/FGV/an. sistemas/46) a
(Senado/2008/FGV/an. sistemas/47) c
(TRE-BA/2010/CESPE/Analista Jud./84 a 89) ccceec
(Inmetro/2009/Cespe/An. – Desv. Sistemas/75) certo

