Estratégia
CONCURSOS

Aul

TCE-RS (Auditor Público Externo - Tecnologia da Informação) Infraestrutura de TI

Professor: Diego Carvalho, Renato da Costa

Índice

XML

Conceitos Básicos

INCIDÊNCIA EM PROVA: ALTA

O e**X**tensible **M**arkup **L**anguage (XML) pode ser definido como uma metalinguagem de marcação extensível, especificada pela W3C, de propósito geral e que define um conjunto de regras para codificar documentos em um formato que seja legível tanto por humanos quanto por máquinas com o intuito principal de facilitar a representação, armazenamento, transporte e intercâmbio de dados entre sistemas de forma padronizada. *Calma! Nós vamos destrinchar tudo ponto por ponto...*

Em primeiro lugar, trata-se de uma metalinguagem! *O que seria uma metalinguagem?* É basicamente uma linguagem utilizada para descrever outras linguagens. Por exemplo: o idioma português é uma metalinguagem, visto que se trata de uma linguagem capaz de descrever outras linguagens, inclusive ela mesma. É possível definir em português como é a estrutura, gramática, sintaxe, ortografia, etc do próprio português.

O XML é considerado uma metalinguagem porque ele também é autodescritivo, isto é, ele é capaz de descrever a sua própria estrutura. Em segundo lugar, trata-se de uma metalinguagem de marcação. *O que significa isso?* Galera, as marcações (também chamadas de tags) são basicamente anotações (sinais e códigos) dentro de um documento que marcam o início e o fim de um elemento de dados que serão intercambiados – além de idealmente definir seus significados.

Notem por meio do exemplo apresentado a seguir que as marcações são aquelas destacadas em azul e elas apenas marcam o início e fim de elementos de dados – elas não fazem parte do conteúdo em si. Dessa forma, essa linguagem (assim como outras linguagens de marcação) permite representar os dados de tal forma que dados de conteúdo fiquem visualmente separados de dados sobre a estrutura do próprio documento. *Bacana?*

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

<carta>
  <de>Banca</de>
  <para>Aluno</para>
  <assunto>Você passou no concurso dos seus sonhos!</assunto>
  <corpo>Isso mesmo que você leu: você está sendo convocado para tomar posse!</corpo>
</carta>
```

Quem aí já ouviu falar em HTML (HyperText Markup Language)? É a linguagem de marcação padrão utilizada na construção de páginas web. *Você acessa páginas web?* Então você acessa uma página escrita (entre outras tecnologias) em HTML. Note que ela também é uma linguagem de marcação, porque ela também tem marcações. Por outro lado, todas as marcações do HTML são pré-definidas, isto é, já existe uma lista de marcações que podem ser definidas.

Já o XML é uma metalinguagem de marcação extensível! *Por que ela é extensível?* Porque as marcações não são pré-definidas – você pode criar suas próprias marcações (tags). Vejam a seguir um exemplo de HTML: observe que se trata também de uma linguagem de marcações, mas as marcações do HTML são pré-definidas. As marcações **<html>**, **<head>**, **<title>**, **<body>**, **<h1>**, **<p>** são pré-definidas na linguagem – já as marcações do XML podem ser criadas pelo usuário...

```
<!DOCTYPE html>
<html>
    <head>
        <title>Título de uma Página Web</title>
    </head>
    <body>
        <h1>Isto é um cabeçalho</h1>
        <p>Isto é um parágrafo</p>
    </body>
</html>
```

Bacana! Nós já entendemos porque XML é uma metalinguagem, porque é uma metalinguagem de marcação e porque é uma linguagem de marcação extensível. *E por que ela é especificada pela W3C?* W3C é a World Wide Web Consortium) – trata-se da principal organização de padronização da World Wide Web. *Por que é bom ser especificada pela W3C?* Porque novas tecnologias e aplicações tentam se adequar às especificações dessa organização. Vamos continuar...

XML é uma linguagem de propósito geral! Nós acabamos de ver que o HTML é uma linguagem que tem basicamente o propósito de definir a estrutura de conteúdos de páginas web, logo se trata de uma linguagem com um propósito bem definido. Já o XML é uma linguagem que tem o propósito de definir a estrutura de intercâmbio de dados de qualquer contexto ou domínio e de forma independente de tecnologia ou plataforma.

O XML define um conjunto de regras para codificar documentos em um formato que seja legível tanto por humanos quanto por máquinas. Essa parte é bacana! Observem o código a seguir...

```
<script type='text/javascript'>
$(document).ready(function()
{$(&#39;img#closed&#39;).click(function(){$(&#39;#bl_banner&#39;).hide(90);});});
document.addEventListener("DOMContentLoaded", function(){
   var popup = document.getElementById("popup");
   var ls = localStorage.getItem("popup");
   var data = new Date();
   var data_atual = data.valueOf();
   var data24 = data.setHours(data.getHours()+24);
   if(ls < data_atual){
      popup.style.display = "block";
      localStorage.setItem("popup", data24);
   }
});
```

Vocês conseguem entendê-lo? Difícil! *Por que?* Porque esse código foi escrito em uma linguagem chamada JavaScript. Só quem consegue entendê-lo são programadores e, principalmente, aqueles que dominam essa linguagem. Já o XML é uma linguagem que pode ser facilmente lida tanto por

máquinas quanto por humanos, conforme vimos no exemplo da página anterior. *Aquele vocês conseguiram ler e entender, correto?* Pois é...

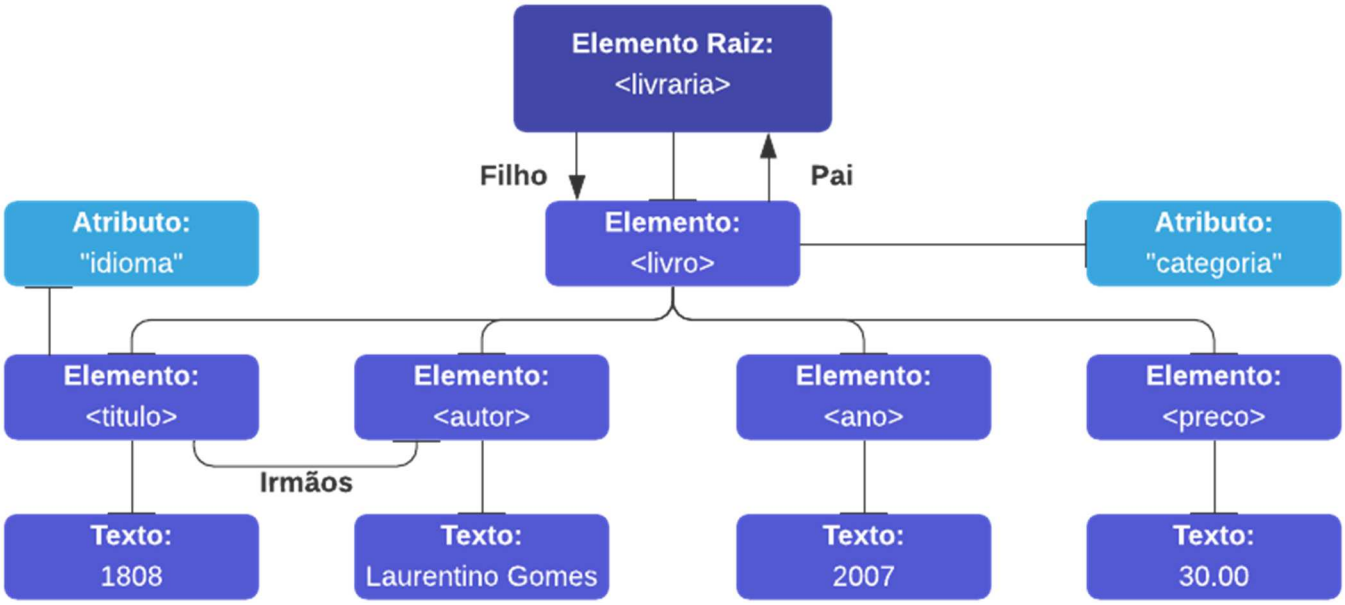
Você bate o olho e consegue identificar que há um emissor de dados, um receptor de dados, um assunto e um corpo contendo o conteúdo em si – é um documento autodescritível. Note que o XML não **executa** nada – ele apenas representa, armazena e facilita o transporte de dados. Quem executa instruções que informam o que o computador deve fazer são as linguagens de programação. XML é uma linguagem de marcação e, não, uma linguagem de programação.

Aliás, essa é a última parte da definição: XML tem o intuito principal de facilitar o intercâmbio de dados entre sistemas de forma padronizada. Trata-se de uma linguagem que busca facilitar o armazenamento, transporte e representação dos dados. Pronto! Terminamos de destrinchar cada ponto da nossa definição. No entanto, ainda não terminamos: vamos falar agora de mais algumas outras características dessa linguagem:

CARACTERÍSTICAS	DESCRIÇÃO
XML SEPARA DADOS DA APRESENTAÇÃO	XML não mantém nenhuma informação sobre como os dados serão exibidos, logo um mesmo documento XML pode ser utilizado em vários cenários de apresentação diferentes.
XML FREQUENTEMENTE COMPLEMENTA O HTML	XML é comumente utilizado para armazenar e transportar dados enquanto HTML é utilizado para formatação e exibição dos mesmos dados – ambos em arquivos separados e tratados independentemente.
XML SUPORTE A UNICODE	XML oferece suporte a Unicode, o que permite a comunicação de quase todas as informações em qualquer linguagem humana escrita.
XML SE ADAPTA A AVANÇOS TECNOLÓGICOS	XML pode se adaptar às novas tecnologias por causa de sua natureza independente de plataforma ou tecnologia. Logo, é uma ótima opção para armazenamento de dados por um longo período.
XML TRATA DADOS EM UMA ESTRUTURA DE ÁRVORE	XML mantém uma estrutura hierárquica de elementos – tanto que o primeiro elemento é sempre o elemento raiz. Isso facilita a representação de dados hierárquicos.
XML É UM FORMATO QUE PODE SER VALIDADO	XML permite fornecer um segundo documento XML – chamado XSD – para descrever exatamente como o arquivo de dados deve ser estruturado, facilitando seu processamento.
XML PERMITE CRIAR OUTRAS LINGUAGENS	XML é uma metalinguagem extensível, logo permite criar outras linguagens. Atualmente, existem linguagens baseadas em XML como WSDL, RSS e XHTML.
XML PERMITE BUSCAS EFICIENTES	Como elementos podem ser unicamente "etiquetados" por meio de tags, isso facilita buscas de dados dentro de documentos.

Estrutura de Árvore

INCIDÊNCIA EM PROVA: BAIXA



Documentos XML são formados como uma árvore de elementos! A imagem acima representa uma estrutura hierárquica de livros que é representada em XML conforme apresenta o código a seguir:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<livraria>
  <livro categoria="historia">
    <titulo idioma="pt">1808</titulo>
    <autor>Laurentino Gomes</autor>
    <ano>2007</ano>
    <preco>30.00</preco>
  </livro>
  <livro categoria="infantil">
    <titulo idioma="en">Harry Potter</titulo>
    <autor>J K. Rowling</autor>
    <ano>2005</ano>
    <preco>29.99</preco>
  </livro>
  <livro categoria="economia">
    <titulo idioma="pt">A Liberdade</titulo>
    <autor>John Stuart Mill</autor>
    <ano>1859</ano>
    <preco>39.95</preco>
  </livro>
</livraria>
```

Observem que uma árvore sempre começa em um elemento raiz e se ramifica da raiz para os elementos filhos, lembrando que todos os elementos podem ter sub-elementos (elementos filhos). Note que os termos pai, filho e irmãos são utilizados para descrever relacionamentos entre elementos. Pais possuem filhos, filhos possuem pais, irmãos são filhos localizados no mesmo nível. E todos os elementos podem ter conteúdos (Ex: Harry Potter) e atributos (categoria=”infantil”).

```
<raiz>
  <filho>
    <sub-filho>...</sub-filho>
  </filho>
</raiz>
```

Diego, por que você pulou a primeira linha do código apresentado na página anterior? Ah, sim! Essa primeira linha (assim como tudo que aparece antes do elemento-raiz) se chama **prolog** e esse componente é responsável por definir a versão do documento, o tipo de codificação de caracteres, instruções de processamento, entre outros. Na verdade, todas essas configurações são opcionais, mas – se existirem – devem ser necessariamente a primeira coisa em um documento.

A declaração do prolog deve ser a primeira coisa em um documento, não pode haver nem um espaço em branco antes. Vamos entendê-la melhor:

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

Essa linha indica que o documento está utilizando a versão de especificação 1.0 do XML e que os caracteres do arquivo utilizado a codificação UTF-8. *Que codificação é essa, professor?* Galera, documentos podem conter caracteres internacionais como ç, ø, æ, å, è, ß. Para evitar erros, recomenda-se especificar o tipo de codificação de caracteres utilizado. UTF-8 é o tipo de codificação de caracteres padrão do XML (assim como do HTML, SQL, etc). Vamos seguir...

```
<livraria>
  <livro categoria="historia">
    <titulo idioma="pt">1808</titulo>
    <autor>Laurentino Gomes</autor>
    <ano>2007</ano>
    <preco>30.00</preco>
  </livro>
  ...
```

A próxima linha após o prolog contém o elemento raiz do documento: `<livraria>`; a linha seguinte começa com um elemento `<livro>`; o elemento livro contém quatro elementos filho: `<titulo>`, `<autor>`, `<ano>`, `<preco>`; e, por fim, a próxima linha finaliza o elemento <livro> por meio de uma tag de fechamento `</livro>`. *Conseguiram perceber por que o XML é considerado uma linguagem autodescritiva capaz de ser lida tanto por humanos como por máquinas?* Pois é, vamos continuar...

Sintaxe XML

INCIDÊNCIA EM PROVA: MÉDIA

Elementos

Um elemento é considerado tudo que se encontra entre a tag inicial e uma tag final, incluindo a própria tag do elemento. Ele pode conter outros elementos, textos e atributos – ou também ser vazio. Observem no código seguinte que os elementos `<titulo>`, `<autor>`, `<ano>` e `<preco>` possuem textos; já os elementos `<livro>` e `<titulo>` possuem atributos; e os elementos `<livraria>` e `<livro>` possuem outros elementos.

```
<livraria>
  <livro categoria="historia">
    <titulo idioma="pt">1808</titulo>
    <autor>Laurentino Gomes</autor>
    <ano>2007</ano>
    <preco>30.00</preco>
  </livro>
</livraria>
```

Um elemento pode ser escrito das duas maneiras apresentadas abaixo – lembrando que todo elemento deve ter uma tag de abertura e de fechamento. *Professor, o prolog apresentado na página anterior não possui nenhuma tag de fechamento!* Sim, bem observado! Isso ocorre por o prolog não é considerado como parte do documento XML em si. Logo, ele não invalida a regra que acabamos de mencionar. *Fechado?*

```
<livro>

</livro>
```

```
<livro

/>
```

É possível existir elementos vazios, isto é, aqueles que não possuem nenhum conteúdo (apesar de poder possuir atributos). Eles podem ser representados da seguinte forma:

```
<elemento-vazio></elemento-vazio>
```

É possível também utilizar uma tag auto-fechada, que – na verdade – produz resultados idênticos ao apresentado na linha anterior:

```
<elemento-vazio/>
```

Por fim, existem algumas regras para nomes de elementos: nomes de elementos devem começar com uma letra ou underscore (_); nomes de elementos não podem começar com "xml" ou suas variações de maiúsculas e minúsculas; nomes de elementos podem conter letras, dígitos, hífen,

underscore e ponto; nomes de elementos não podem conter espaços. Todo nome pode ser utilizado – não há palavras reservadas (exceto "xml" e suas variações).

Atributos

Atributos são informações adicionais sobre um elemento. Eles vêm dentro da tag de início de um elemento entre aspas (simples ou duplas) e em um formato nome=valor:

```
<pessoa genero="feminino">
<pessoa genero='feminino>
```

Se o próprio valor do atributo contiver aspas duplas, você poderá usar aspas simples (apóstrofos) ou símbolos de escape (que veremos em detalhes mais à frente). Vejamos...

```
<lutador nome='Anderson "Spyder" Silva'>
<lutador nome="Anderson &quot;Spyder&quot; Silva">
```

Agora notem os dois exemplos a seguir: no primeiro, gênero é um atributo; no segundo, é um elemento. Ambos fornecem a mesma informação – não há regras para usar um ou outro!

```
<pessoa genero="feminino">
  <primeiro-nome>Ana</primeiro-nome>
  <ultimo-nome>de Amsterdam</ultimo-nome>
</pessoa>
```

```
<pessoa>
  <genero>feminino</genero>
  <primeiro-nome>Ana</primeiro-nome>
  <ultimo-nome>de Amsterdam</ultimo-nome>
</pessoa>
```

A única ressalva é que atributos são bem mais limitados que elementos, uma vez que não podem conter múltiplos valores e não podem ser dispostos em uma hierarquia. Além disso, é fundamental destacar um elemento pode conter vários atributos, mas atributos não podem conter múltiplos valores ou estruturas hierárquicas – diferentemente dos elementos. Ademais, dentro de um mesmo elemento, dois atributos jamais podem ter o mesmo nome.

Namespaces

Namespaces são recursos que permitem evitar conflitos de nomes de elementos. Nós vimos que esses nomes são definidos pelo criador do documento, por outro lado frequentemente isso pode resultar em conflitos ao tentar misturar documentos de aplicações diferentes que utilizam o mesmo nome. Vejam no exemplo seguinte dois documentos que utilizam o mesmo nome de elemento, mas um trata do maior clube de futebol do planeta e o outro trata de um bairro do Rio de Janeiro!

```
<flamengo>
  <esporte>Futebol</esporte>
  <mascote>Urubu</mascote>
```

```
</flamengo>
```

```
<flamengo>
  <populacao>50.640</populacao>
  <distrito>Zona Sul</distrito>
</flamengo>
```

Se esses documentos interagirem de alguma forma, pode ocorrer um conflito de nomes. Para evitar esse conflito, podemos utilizar prefixos:

```
<raiz>
  <a:flamengo>
    <a:esporte>Futebol</a:esporte>
    <a:mascote>Urubu</a:mascote>
  </a:flamengo>
  <b:flamengo>
    <b:populacao>50.640</b:populacao>
    <b:distrito>Zona Sul</b:distrito>
  </b:flamengo>
</raiz>
```

Ao utilizar um prefixo, um **namespace** deve ser definido. Esse namespace é definido pelo atributo **xmlns** na tag de abertura de um elemento. A declaração do namespace segue a seguinte sintaxe:

```
xmlns:prefix="URI"
```

Vamos ver como é que fica aplicado ao exemplo anterior:

```
<raiz>

<a:flamengo xmlns:a="http://www.flamengo.com.br">
  <a:esporte>Futebol</a:esporte>
  <a:mascote>Urubu</a:mascote>
</a:flamengo>

<b:flamengo xmlns:b="https://prefeitura.rio">
  <b:populacao>50.640</b:populacao>
  <b:distrito>Zona Sul</b:distrito>
</b:flamengo>

</raiz>
```

No exemplo acima, o atributo xmlns no primeiro elemento **<flamengo>** dá um namespace para **a:**; já o atributo xmlns no segundo elemento **<flamengo>** dá um namespace para **b:**. Lembrando que, quando um namespace é definido para um elemento, todos os elementos filhos com o mesmo prefixo são associados ao mesmo namespace. Além disso, os namespaces também podem ser declarados no elemento raiz do documento.

```
<raiz xmlns:a="http://www.flamengo.com.br"
xmlns:b="https://prefeitura.rio">

<a:flamengo>
  <a:esporte>Futebol</a:esporte>
```

```
  <a:mascote>Urubu</a:mascote>
</a:flamengo>

<b:flamengo>
  <b:populacao>50.640</b:populacao>
  <b:distrito>Zona Sul</b:distrito>
</b:flamengo>

</raiz>
```

Professor, fiquei meio perdido! O namespace é um site? Não, o namespace é um nome, mas esse nome precisa ser um identificador único. Logo, é comum a utilização de uma URI (Uniform Resource Identifier) para evitar de ter dois identificadores de namespace com o mesmo nome, mas é completamente possível colocar qualquer identificador desde que ele seja único (Ex: CPF, RG, etc). *Entendido, galera?* Vamos seguir...

Por fim, é importante mencionar que – quando inserimos um namespace no próprio elemento – podemos evitar de usar os prefixos nos elementos filhos:

```
<flamengo xmlns:a="http://www.flamengo.com.br">
  <esporte>Futebol</esporte>
  <mascote>Urubu</mascote>
</flamengo>

<flamengo xmlns:b="https://prefeitura.rio">
  <populacao>50.640</populacao>
  <distrito>Zona Sul</distrito>
</flamengo>
```

Comentários

A sintaxe para escrever comentários é extremamente simples, só lembrando que não se pode utilizar dois traços no meio do comentário para não confundir o processador do documento:

```
<!-- Isso é um comentário válido -->
<!-- Isso é um comentário -- inválido -->
```

```
<flamengo xmlns:a="http://www.flamengo.com.br">
  <esporte>Futebol</esporte>
  <mascote>Urubu</mascote>
</flamengo>

<!—Flamengo é o maior time de futebol do mundo-->

<flamengo xmlns:b="https://prefeitura.rio">
  <populacao>50.640</populacao>
  <distrito>Zona Sul</distrito>
</flamengo>
```

Espaços em Branco

Em contraste com HTML, XML não trunca ou elimina múltiplos espaços em branco em documentos (Ex: espaços, tabs, quebra de linha). Dessa forma, se você deixar espaços em branco nos seus dados, eles vão permanecer no documento. Vejam os exemplos apresentados no código a seguir e note que temos três registros diferentes. Caso isso fosse um Documento HTML, essa linguagem truncaria (eliminaria) os espaços em branco.

```
<nome>DiegoCarvalho</nome>
<nome>Diego Carvalho</nome>
<nome>Diego                         Carvalho</nome>
```

Caracteres Especiais

Nós vimos que os símbolos de menor (<) e maior (>) possuem significados especiais e são utilizados para indicar o início ou fim de uma tag. Se você inserir um caractere como "<" dentro de um elemento (conforme é apresentado na linha de código apresenta a seguir), isso gerará um erro porque ocasionará uma ambiguidade para o processador[1] do documento que achará que se trata do início de um novo elemento.

```
<concurseiro>salario < 30000</concurseiro>
```

Para evitar esse tipo de problema, existem cinco entidades de escape pré-definidos que ajudam a não confundir o processador do documento:

MENOR QUE	MAIOR QUE	E COMERCIAL	APÓSTROFO	ASPAS
<	>	&	'	"
&lt	&gt	&amp	&apos	&quot

É importante destacar que os únicos elementos terminantemente proibidos são < e &. De todo modo, é uma boa prática substituir os outros três pelos seus símbolos correspondentes.

[1] O documento XML é processado por um *parser*, isto é, uma ferramenta que quebra o código em partes menores para verificar sua estrutura gramatical e sintática.

Validação de XML

INCIDÊNCIA EM PROVA: MÉDIA

Dizemos que um documento XML bem formado é aquele que obedece categoricamente às suas regras de sintaxe. Nós já vimos todas as regras, mas vamos relembrá-las:

REGRA 1	Documentos XML devem possuir um único elemento-raiz.
REGRA 2	Todos os elementos devem conter uma *tag* de fechamento.
REGRA 3	Elementos devem estar corretamente aninhados.
REGRA 4	Atributos devem possuir valor entre aspas simples ou duplas.
REGRA 5	Nomes de tags e atributos são Case-Sensitive.

Legal! Aprendemos o que define um Documento XML bem formado! Agora é o momento de ver o que define um documento XML válido! *São coisas diferentes, professor?* Sim, são conceitos complementares. Todo documento XML válido é bem formado, mas nem todo documento bem formado é válido. O que define a validade de um documento XML não é a sua sintaxe e, sim, se ele é consistente com o seu esquema. *Como é, professor?* Vamos explicar melhor...

Um esquema (ou arquivo de definição) é um modelo que descreve como devem ser os elementos, atributos, dados, entre outros. Um documento XML será considerado válido se ele for bem formado e... obedece às regras de seu esquema. *Vamos abstrair um pouco?* Imaginem o seguinte: eu quero construir um carro! Logo, para que seja um carro bem formado, é preciso que tenha um único volante, quatro rodas, um sistema elétrico e hidráulico, entre outros!

Nós podemos dizer que qualquer veículo que contenha essas características será considerado um carro bem formado. *Fechado?* Mas eu não quero um carro qualquer... eu quero uma Ferrari Califórnia! Logo, para que esse carro seja validado como uma Ferrari Califórnia, é preciso que tenha um motor 4.3L de 460 cv, torque de 77 kgfm, 900 kg, 8 cilindros, transmissão de sete velocidades, suspensão esportiva, design como o da imagem ao lado!

Qualquer objeto que seja considerado um carro bem formado e que contenha essas características que eu defini de forma específica será considerado uma Ferrari California. Note também que nem todo carro é uma Ferrari Califórnia, mas toda Ferrari Califórnia é um carro. Um outro exemplo que eu gosto de ensinar é a relação entre uma prova discursiva e o padrão de resposta. Muitas vezes, as bancas disponibilizam um padrão de resposta esperada para as provas discursivas.

Nesse contexto, imagine que você fez uma discursiva com o português correto, ortografia impecável, pontuação adequada, sintaxe satisfatória e orações bem construídas – sua prova discursiva estaria bem formada. Agora imaginem que, apesar de tudo isso, seu texto não tem nada a ver com o que o examinador esperava como resposta. Nesse caso, sua prova discursiva está bem formada, mas não é válida. *Entenderam agora?*

Agora vamos voltar para o mundo real: *é obrigatória a utilização de um esquema?* Não! Nós mostramos vários exemplos que não precisavam de esquema nenhum. Em geral, quando se trabalha com documentos pequenos ou se deseja apenas realizar alguns testes, criar um esquema pode ser uma perda de tempo. Já em outros contextos, um esquema permite a verificação da estrutura e regras de preenchimento de um arquivo contendo dados no formato XML.

Dessa forma, ele serve para facilitar a vida do desenvolvedor, ajuda a padronizar documentos para diversos usuários, auxilia o compartilhamento de dados, definem quais tags são válidas e fornece uma boa referência de estrutura para todos que queiram interagir e utilizar os dados. Existem dois tipos principais de arquivos de definição: DTD e XSD! Vamos começar falando do primeiro porque ele é o mais antigo.

O DTD (Document Type Definition) define a estrutura e os elementos/atributos legais permitidos dentro de um documento XML. Trata-se de um conjunto de regras que define quais tipos de dados e entidades farão parte de um documento XML. Estas regras serão utilizadas para que o analisador sintático verifique se o documento é válido ou não. O DTD pode estar definida dentro do próprio arquivo XML ou em um arquivo à parte com extensão .dtd. Vejamos um exemplo anterior:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE carta SYSTEM "Carta.dtd">

<carta>
  <de>Banca</de>
  <para>Aluno</para>
  <assunto>Você passou no concurso dos seus sonhos!</assunto>
  <corpo>Isso mesmo que você leu: você está sendo convocado para tomar posse!</corpo>
</carta>
```

Note que agora a segunda linha contém uma declaração DOCTYPE. *O que isso significa, professor?* Trata-se de uma instrução que faz uma referência a um arquivo DTD externo (no caso, Carta.dtd).

```
<!DOCTYPE carta
[
<!ELEMENT carta (de,para,assunto,corpo)>
<!ELEMENT de (#PCDATA)>
<!ELEMENT para (#PCDATA)>
<!ELEMENT assunto (#PCDATA)>
<!ELEMENT corpo (#PCDATA)>
]>
```

Esse arquivo pode ser interpretado da seguinte maneira:

- **!DOCTYPE carta** - define que carta é o elemento raiz do documento;
- **!ELEMENT carta** - define que o elemento carta deve conter esses quatro elementos em ordem;
- **!ELEMENT de** - define o elemento "de" como sendo do tipo #PCDATA;
- **!ELEMENT para** - define o elemento "para" como sendo do tipo #PCDATA;
- **!ELEMENT assunto** - define o elemento "assunto" como sendo do tipo #PCDATA;
- **!ELEMENT corpo** - define o elemento "corpo" como sendo do tipo #PCDATA.

Professor, o que é esse #PCDATA? É um código que indica que determinado elemento contém dados que devem ser processados pelo validador. Existe também o #CDATA, que indica que determinado elemento contém dados que não devem ser processados pelo validador (Ex: caracteres especiais). Por meio do meu arquivo DTD, eu consigo analisar se meu Documento XML é válido. Eu não coloquei no exemplo, mas é possível representar números de ocorrências de um elemento.

Por meio de alguns símbolos: o sinal de mais (+) indica que se trata de um elemento obrigatório que pode aparecer uma ou mais vezes; o sinal de asterisco (*) indica que se trata de um elemento opcional que pode aparecer zero ou mais vezes; o sinal de ponto de interrogação (?) indica que se trata de um elemento opcional que pode aparecer zero ou uma vez; e se não houver símbolo algum, indica que um elemento deve aparecer uma e apenas uma vez. *E o XML Schema?*

Assim como o DTD, o XML Schema Definition (XSD) descreve a estrutura de um documento XML. No entanto, trata-se de uma ferramenta mais poderosa por ser capaz de suportar a criação de namespaces, a definição de novos tipos de dados, a definição de restrições, a conversão de dados, entre outras características que não vamos detalhar. Em contraste com o DTD, ele é escrito em XML – não sendo necessário aprender outra linguagem. Vamos ver um exemplo:

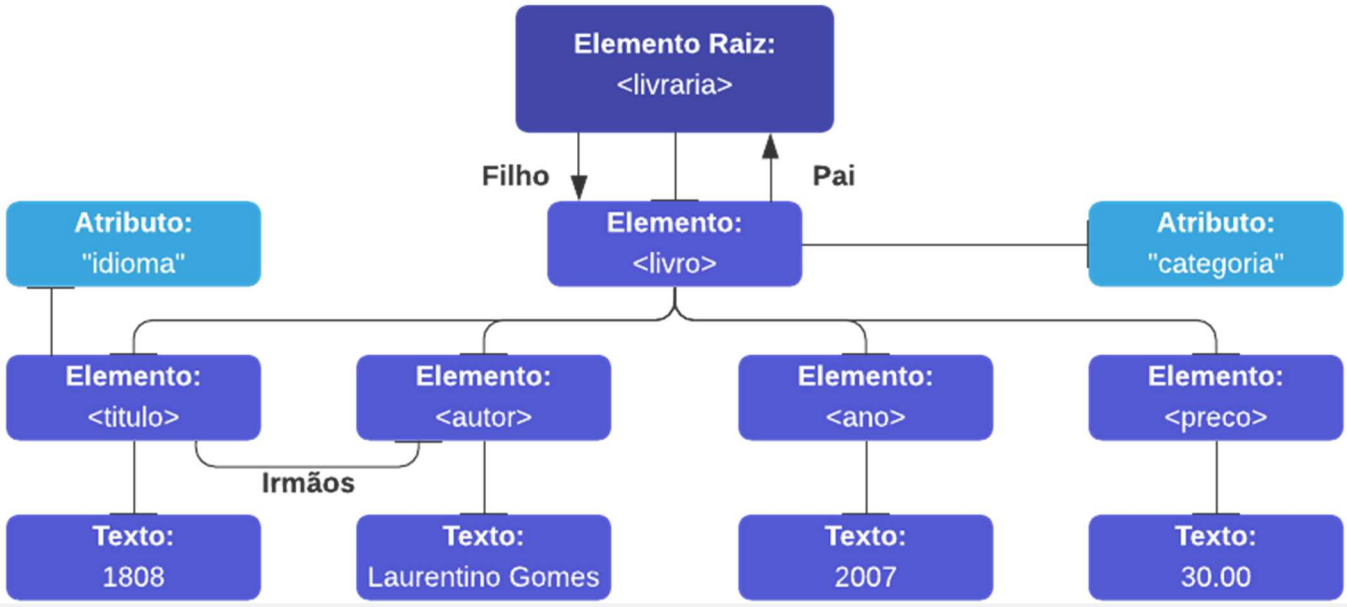
```
<xs:element name="carta">

<xs:complexType>
  <xs:sequence>
    <xs:element name="de" type="xs:string"/>
    <xs:element name="para" type="xs:string"/>
    <xs:element name="assunto" type="xs:string"/>
    <xs:element name="corpo" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

</xs:element>
```

Esse arquivo pode ser interpretado da seguinte maneira:

- **<xs:element name="carta">** define um elemento chamado "carta";
- **<xs:complexType>** "carta" é um elemento do tipo complexo;
- **<xs:sequence>** o tipo complexo é uma sequência de elementos;
- **<xs:element name="de" type="xs:string">** o elemento é uma string;
- **<xs:element name="para" type="xs:string">** o elemento é uma string;
- **<xs:element name="assunto" type="xs:string">** o elemento é uma string;
- **<xs:element name="corpo" type="xs:string">** o elemento é uma string;

RESUMO

CARACTERÍSTICAS	DESCRIÇÃO
XML SEPARA DADOS DA APRESENTAÇÃO	XML não mantém nenhuma informação sobre como os dados serão exibidos, logo um mesmo documento XML pode ser utilizado em vários cenários de apresentação diferentes.
XML FREQUENTEMENTE COMPLEMENTA O HTML	XML é comumente utilizado para armazenar e transportar dados enquanto HTML é utilizado para formatação e exibição dos mesmos dados – ambos em arquivos separados e tratados independentemente.
XML SUPORTE A UNICODE	XML oferece suporte a Unicode, o que permite a comunicação de quase todas as informações em qualquer linguagem humana escrita.
XML SE ADAPTA A AVANÇOS TECNOLÓGICOS	XML pode se adaptar às novas tecnologias por causa de sua natureza independente de plataforma ou tecnologia. Logo, é uma ótima opção para armazenamento de dados por um longo período.
XML TRATA DADOS EM UMA ESTRUTURA DE ÁRVORE	XML mantém uma estrutura hierárquica de elementos – tanto que o primeiro elemento é sempre o elemento raiz. Isso facilita a representação de dados hierárquicos.
XML É UM FORMATO QUE PODE SER VALIDADO	XML permite fornecer um segundo documento XML – chamado XSD – para descrever exatamente como o arquivo de dados deve ser estruturado, facilitando seu processamento.
XML PERMITE CRIAR OUTRAS LINGUAGENS	XML é uma metalinguagem extensível, logo permite criar outras linguagens. Atualmente, existem linguagens baseadas em XML como WSDL, RSS e XHTML.
XML PERMITE BUSCAS EFICIENTES	Como elementos podem ser unicamente "etiquetados" por meio de tags, isso facilita buscas de dados dentro de documentos.



ELEMENTO

Um elemento é tudo que se encontra entre a tag inicial e uma tag final, incluindo a própria tag do elemento. Ele pode conter outros elementos, textos e atributos – ou também ser vazio.

```
<livro>

</livro>
```

```
<livro

/>
```

ATRIBUTO

Atributos são informações adicionais sobre um elemento. Eles vêm dentro da tag de início de um elemento entre aspas (simples ou duplas) e em um formato nome=valor:

```
<pessoa genero="feminino">
<pessoa genero='feminino>
```

NAMESPACES

Namespaces são recursos que permitem evitar conflitos de nomes de elementos. Ele pode ser inserido na raiz ou no próprio elemento e representa um identificador único (URI)

```
<raiz xmlns:a="http://www.flamengo.com.br"
xmlns:b="https://prefeitura.rio">

<a:flamengo>
  <a:esporte>Futebol</a:esporte>
  <a:mascote>Urubu</a:mascote>
</a:flamengo>

<b:flamengo>
  <b:populacao>50.640</b:populacao>
  <b:distrito>Zona Sul</b:distrito>
</b:flamengo>

</raiz>
```

```
<flamengo xmlns:a="http://www.flamengo.com.br">
  <esporte>Futebol</esporte>
  <mascote>Urubu</mascote>
</flamengo>

<flamengo xmlns:b="https://prefeitura.rio">
  <populacao>50.640</populacao>
  <distrito>Zona Sul</distrito>
</flamengo>
```

COMENTÁRIOS

A sintaxe para escrever comentários é extremamente simples, só lembrando que não se pode utilizar dois traços no meio do comentário para não confundir o processador do documento:

```
<!-- Isso é um comentário válido -->
<!-- Isso é um comentário -- inválido -->
```

ESPAÇOS EM BRANCO

Em contraste com HTML, XML não trunca ou elimina múltiplos espaços em branco em documentos (Ex: espaços, tabs, quebra de linha).

```
<nome>DiegoCarvalho</nome>
<nome>Diego Carvalho</nome>
<nome>Diego                    Carvalho</nome>
```

CARACTERES ESPECIAIS

Existem caracteres especiais que devem ser escapados pelas entidades apresentadas na tabela seguinte. É importante destacar que os únicos elementos terminantemente proibidos são < e &.

MENOR QUE	MAIOR QUE	E COMERCIAL	APÓSTROFO	ASPAS
<	>	&	'	"
&lt	&gt	&amp	&apos	&quot

XML BEM FORMADO

REGRA 1	Documentos XML devem possuir um único elemento-raiz.
REGRA 2	Todos os elementos devem conter uma *tag* de fechamento.
REGRA 3	Elementos devem estar corretamente aninhados.
REGRA 4	Atributos devem possuir valor entre aspas simples ou duplas.
REGRA 5	Nomes de tags e atributos são Case-Sensitive.

DOCUMENT TYPE DEFINITION

```
<!DOCTYPE carta
[
<!ELEMENT carta (de,para,assunto,corpo)>
<!ELEMENT de (#PCDATA)>
<!ELEMENT para (#PCDATA)>
<!ELEMENT assunto (#PCDATA)>
<!ELEMENT corpo (#PCDATA)>
]>
```

- **!DOCTYPE carta** - define que carta é o elemento raiz do documento;
- **!ELEMENT carta** - define que o elemento carta deve conter esses quatro elementos em ordem;
- **!ELEMENT de** - define o elemento "de" como sendo do tipo #PCDATA;

- `!ELEMENT para` - define o elemento "para" como sendo do tipo #PCDATA;
- `!ELEMENT assunto` - define o elemento "assunto" como sendo do tipo #PCDATA;
- `!ELEMENT corpo` - define o elemento "corpo" como sendo do tipo #PCDATA.

XML SCHEMA DEFINITION (XSD)

```
<xs:element name="carta">

<xs:complexType>
  <xs:sequence>
    <xs:element name="de" type="xs:string"/>
    <xs:element name="para" type="xs:string"/>
    <xs:element name="assunto" type="xs:string"/>
    <xs:element name="corpo" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

</xs:element>
```

Esse arquivo pode ser interpretado da seguinte maneira:

- `<xs:element name="carta">` define um elemento chamado "note";
- `<xs:complexType>` "carta" é um elemento do tipo complexo;
- `<xs:sequence>` o tipo complexo é uma sequência de elementos;
- `<xs:element name="de" type="xs:string">` o elemento é uma string;
- `<xs:element name="para" type="xs:string">` o elemento é uma string;
- `<xs:element name="assunto" type="xs:string">` o elemento é uma string;
- `<xs:element name="corpo" type="xs:string">` o elemento é uma string;

QUESTÕES COMENTADAS – CESPE

1. **(FGV / TCE-TO – 2022)** Um documento XML é considerado bem formado quando segue as regras de sintaxe estabelecidas na especificação da linguagem. A alternativa que apresenta um documento XML bem formado é:

a) `<Letra> eh uma tag igual a </letra>`
b) `<p>Este eh um paragrafo comum.</p>`
c) `<!--Este eh um -- comentario padrao -->`
d) `<mensagem>salario < 1000</mensagem>`
e) `<b><i>Este texto eh negrito e italico</b></i>`

Comentários:

(a) Errado. <Letra> deve ser fechado com </Letra> e, não, </letra>; (b) Correto; (c) Errado, não é permitido ter – dentro do comentário; (d) Errado, o caractere < confunde o processador e não deve ser utilizado; (e) Errado. Primeiro, fecha-se o </i> e depois o .

```
<!-- Isso é um comentário válido -->
<!-- Isso é um comentário -- inválido -->
```

MENOR QUE	MAIOR QUE	E COMERCIAL	APÓSTROFO	ASPAS
<	>	&	'	"
&lt	&gt	&amp	&apos	&quot

Gabarito: Letra B

---

2. **(CESPE / DPDF - 2022)** Nos códigos em XML a seguir, sexo é um atributo no código A e um elemento no código B, mas ambos os códigos fornecem as mesmas informações.

código A
```
<pessoa sexo="fem">
        <nome>Maria</nome>
        <sobrenome>Silva</sobrenome>
</pessoa>
```

código B
```
<pessoa>
        <sexo>fem</sexo>
        <nome>Maria</nome>
        <sobrenome>Silva</sobrenome>
</pessoa>
```

Comentários:

Um elemento é considerado tudo que se encontra entre a tag inicial e a tag final. Dentro de um elemento, podem conter outros elementos, textos e atributos. Dessa forma, no código A, *sexo* é um atributo do elemento *pessoa* (observem que ele vem no formato nome=valor). Já no código B, é um elemento. Além disso, não há diferenciação entre os dois modos de utilização.

Gabarito: Correto

---

3. **(CESPE / Polícia Federal - 2018)** Em arquivos no formato XML, as tags não são consideradas metadados.

Comentários:

As tags são consideradas metadados porque fornecem informações sobre os dados que elas envolvem e dados sobre dados são metadados.

Gabarito: Errado

---

4. **(CESPE / SEDF – 2017)** Um dos objetivos do projeto XML é que o número de recursos opcionais da linguagem deve ser maximizado para torná-la versátil e adaptável.

Comentários:

De acordo com a documentação oficial, o número de recursos opcionais deve ser mantido no absoluto mínimo, idealmente zero. O que torna a linguagem versátil e adaptável é a sua extensibilidade em relação à criação de tags customizadas.

Gabarito: Errado

---

5. **(CESPE / TCE-PA – 2016)** Em um documento XML, deve haver diferenciação entre letras maiúsculas e minúsculas e os comentários devem ter a seguinte sintaxe:

<!--comentario-->.

Comentários:

Perfeito! Essa é a sintaxe correta de um comentário em XML.

Gabarito: Correto

---

6. **(CESPE / TCE-PA – 2016)** As desvantagens dos esquemas XML incluem a falta de suporte a diferentes tipos de dados.

Comentários:

O XML possui vários tipos de dados predefinidos (Ex: string, booleano, inteiro, data, etc). Além disso, existe a possibilidade de criar o seu próprio tipo.

Gabarito: Errado

---

7. **(CESPE / TCE-PA – 2016)** Um arquivo XML deve conter, no máximo, 1.024 tags. Se o uso de uma quantidade maior de tags for necessário, deve-se adotar o seguinte recurso, a fim de aumentar a quantidade de tags referenciadas pelo arquivo XML principal: um arquivo XML fazer referência a outro.

Comentários:

Não existe quantidade máxima limite de tags.

Gabarito: Errado

QUESTÕES COMENTADAS – FGV

8. **(FGV / FUNSAÚDE - 2021)** Maria está editando um arquivo XML por meio do bloco de notas do Windows, e deve tomar cuidado com certos caracteres que têm funções especiais. Assinale a lista que contém apenas caracteres especiais do XML.

a) < > & #
b) < > & "
c) > & " /
d) @ " / \
e) / \ @ "

Comentários:

MENOR QUE	MAIOR QUE	E COMERCIAL	APÓSTROFO	ASPAS
<	>	&	'	"
&lt	&gt	&amp	&apos	&quot

Gabarito: Letra B

---

9. **(FGV / TCE-AM – 2021)** O código XML sintaticamente correto é:

a)
```
    <produtos>
  <livro título="Estruturas">
    <ano><2021></ano>
  </livro>
</produtos>
```

b)
```
<produtos>
  <livro título=Estruturas>
    <ano>&<2021&></ano>
  </livro>
</produtos>
```

c)
```
<produtos>
  <livro título="Estruturas">
    <ano>2021<ano>
  <livro>
</produtos>
```

d)
```
<produtos>
  <livro título=Estruturas>
    <ano>"<2021>”</ano>
  </livro>
</produtos>
```

```
<produtos>
  <livro título="Estruturas">
    <ano>&lt;2021&gt;</ano>
  </livro>
</produtos>
```

e)

Comentários:

(a) Errado, os caracteres "<" e ">" precisam ser escapados; (b) Errado, faltam as aspas em "Estruturas"; (c) Errado, faltou fechar a tag </ano> e a tag </livro>; (d) Errado, faltam as aspas em "Estruturas"; (e) Correto. Alguns caracteres especiais devem ser especificados com o uso de entidades pré-definidas (no caso & lt; , & amp; e & gt; , respectivamente).

Gabarito: Letra E

---

10. (FGV / IMBEL – 2021) O uso do XML é bastante difundido no Brasil para troca de dados em aplicações como notas fiscais, procedimentos médicos, e várias outras. Assinale a utilidade de um Schema XML em aplicações dessa natureza.

a) Direciona os aplicativos que leem arquivos no formato XML.
b) Permite a conversão automática de arquivos XML para o formato CSV.
c) Estabelece precisamente a versão do XML em uso num arquivo no formato XML.
d) Permite que um Web Service interprete corretamente um arquivo de dados no formato XML.
e) Permite a verificação da estrutura e regras de preenchimento de um arquivo contendo dados no formato XML.

Comentários:

(a) Errado, isso é uma função do sistema operacional; (b) Errado, essa definitivamente não é uma aplicação do XML Schema; (c) Errado, isso é função do prolog e, não, do XML Schema; (d) Errado, essa não é uma de suas funções; (e) Correto, o XML Schema realmente permite a verificação da estrutura e regras de preenchimento de um arquivo contendo dados no formato XML.

Gabarito: Letra E

---

11. (FGV / MPE-RJ – 2019) A troca de dados entre sistemas computacionais é normalmente realizada por meio de arquivos que seguem padrões de formato e organização. Desse modo, diferentes agentes com diferentes equipamentos podem enviar e receber dados estruturados muito facilmente. Nesse contexto, analise um trecho do conteúdo de um dado arquivo a seguir.

```
<nota>
          <para>Rita</para>
          <de>Bernardo</de>
          <título>Lembrete</título>
          <texto>O pacote &lt;chegou&gt; ...</texto>
```

```
</nota>
```

Com base nesse trecho, é correto deduzir que a organização desse arquivo segue o padrão conhecido como:

a) CSS.
b) CSV.
c) ODF.
d) PDF.
e) XML.

Comentários:

(a) Errado, CSS é uma linguagem de estilo de páginas web; (b) Errado, CSV é um formato de dados tabulares separados por um delimitador; (c) Errado, ODF é um formato para criação de arquivos do LibreOffice; (d) Errado, PDF é um formato de documentos portáveis; (e) Correto.

Gabarito: Letra E

---

12. (FGV / DPE-RJ – 2019) Considere os trechos XML exibidos a seguir.

I.
```
<p>Um primeiro exemplo</p>
<br/>
```

II.
```
<message>Texto breve</message>
```

III.
```
<b><i>Texto com destaque.</b></i>
```

IV.
```
<p>Note que, para x>1, a resposta é sim.</p>
```

O número de trechos válidos é:

a) 0.
b) 1.
c) 2.
d) 3.
e) 4.

Comentários:

(I) Errado, não pode haver elementos após o fechamento do elemento raiz; (II) Correto; (III) Errado, as tags não estão aninhadas corretamente; (IV) Errado, recomenda-se que o caractere “>” seja escapado por meio de um &gt.

Gabarito: Letra B

---

13. (FGV / MPE-AL – 2018) No XML, a sequência de símbolos

```
&lt;
```

Representa:

a) <
b) >
c) “
d) &
e) ’

Comentários:

MENOR QUE	MAIOR QUE	E COMERCIAL	APÓSTROFO	ASPAS
<	>	&	‘	“
&lt	&gt	&amp	&apos	&quot

Gabarito: Letra A

---

14. (FGV / Câmara de Salvador-BA – 2018) Analise o conteúdo XML de um arquivo de seis linhas, exibido a seguir.

```
<?xml version="1.0" encoding="UTF-8"?>
<database catalogo="BD" user="U1">
<SQL>
  select * FROM T where a < 10
</SQL>
</database>
```

A validação desse arquivo apontaria um erro na linha de número:

a) 1.
b) 2.
c) 3.
d) 4.
e) 6.

Comentários:

Ocorrerá um erro na Linha 4, dado que é recomendado escapar o caractere < por meio de um &lt.

Gabarito: Letra D

---

15. (FGV / Prefeitura de Niterói-RJ – 2018) Considere a declaração do tipo de documento (DTD) a seguir.

```
<!ELEMENT blog (nome, autor+, artigo*, permalink?) >
<!ELEMENT nome (#PCDATA)>
<!ELEMENT autor (#PCDATA)>
<!ELEMENT artigo (#PCDATA)>
<!ELEMENT permalink (#PCDATA)>
```

Em um documento XML, que obedece a esse conjunto de regras,

a) deve haver precisamente um elemento do tipo artigo.
b) podem ocorrer vários elementos do tipo permalink.
c) pode conter um elemento do tipo nome e outro elemento do tipo autor, em qualquer ordem.
d) deve existir um elemento do tipo autor, sendo permitidas múltiplas ocorrências deste tipo de elemento.
e) não é necessário haver um elemento do tipo nome.

Comentários:

(a) Errado, o asterisco indica que o elemento pode aparecer zero ou mais vezes; (b) Errado, a interrogação indica que o elemento deve aparecer zero ou uma vez; (c) Errado, a ordem dos elementos entre parênteses deve ser respeitada; (d) Correto, o sinal de mais indica que o elemento pode aparecer uma ou mais vezes; (e) Errado, quando não há nenhum símbolo, indica que deve haver necessariamente um único elemento.

Gabarito: Letra D

---

16. (FGV / Prefeitura de Niterói-RJ – 2018) XML é uma linguagem de marcação projetada para descrever e transportar dados. Dado que em um documento XML é permitido ao desenvolvedor de software definir seus próprios elementos, pode ser necessário utilizar namespaces para evitar conflitos de nomes. Em relação à namespaces em XML, analise as afirmativas a seguir.

I. Um namespace pode ser declarado no elemento em que é utilizado ou no elemento raiz do documento XML.
II. O atributo uri é reservado em XML para indicar que um prefixo está associado ao namespace.
III. As várias declarações de namespace com prefixos podem ser feitas em um elemento, mas devem possuir prefixos diferentes.

Assinale:

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

Comentários:

(I) Correto; (II) Errado, ele é utilizado para identificar de forma única um prefixo de um namespace; (III) Correto, não há nenhum problema desde que os prefixos sejam diferentes.

Gabarito: Letra D

---

17. (FGV / IBGE – 2017) As declarações de elementos na DTD determinam a possível estrutura de um documento XML. Analise a DTD a seguir:

```
<!DOCTYPE memo
[
<!ELEMENT memo (from, to+, date?, content)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT content (p*)>
<!ELEMENT p (#PCDATA)>
]>
```

É correto afirmar que o(s) elemento(s):

a) **memo** pode conter os elementos **from**, **to**, **date** e **content** em qualquer ordem;
b) **content** deve conter um ou mais elementos **p**;
c) **date** é opcional;
d) **to** é obrigatório e precisa ocorrer mais de uma vez dentro do elemento **memo**;
e) **from**, **to** e **date** podem conter qualquer um dos elementos descritos na DTD.

Comentários:

(a) Errado, a ordem deve ser respeitada; (b) Errado, deve conter necessariamente zero ou mais elementos; (c) Correto; (d) Errado, deve ocorrer uma ou mais vezes; (e) Errado, não há nada que indique isso.

Gabarito: Letra C

---

18. (FGV / ALERJ – 2017) XML (Extensible Markup Language) é um sistema de codificação que permite que diferentes tipos de informação sejam distribuídos através da World Wide Web. Com a XML, diversos sistemas de informação, semelhantes ou não, se comunicam de forma transparente entre si. Em relação à linguagem XML, analise as afirmativas a seguir:

I. Seções CDATA podem ocorrer em qualquer parte de um documento XML e devem ser utilizadas para inserir blocos de texto que contenham caracteres especiais como & e <.
II. Documentos XML bem formados devem ter um DTD (Document Type Definition) associado e obedecer a todas as regras que o DTD contém.
III. Na linguagem XML é permitido omitir as tags finais em elementos não vazios.

Está correto o que se afirma em:

a) somente I;
b) somente II;
c) somente III;
d) somente I e II;
e) I, II e III.

Comentários:

(I) Correto, elas podem ser utilizadas em qualquer parte do documento e são úteis para textos que contenham caracteres especiais – indicando que esses caracteres não devem ser processados pelo parser; (II) Errado, não é obrigatório ter um DTD ou XML Schema; (III) Errado, não é permitido omitir tags de fechamento em elementos vazios ou não.

Gabarito: Letra A

---

19. (FGV / Prefeitura de Paulínia-SP – 2016) Analise o trecho de um documento XML exibido a seguir.

```
<message>salary &lt; 1000</message>
```

Assinale a opção que indica o significado do símbolo "&lt".

a) É uma diretiva de formatação de texto.
b) Representa o caracter "<".
c) É um operador de multiplicação de constantes.
d) É uma constante matemática da biblioteca "&math".
e) Representa um bookmark.

Comentários:

MENOR QUE	MAIOR QUE	E COMERCIAL	APÓSTROFO	ASPAS

<	>	&	'	"
&lt	&gt	&amp	&apos	&quot

Gabarito: Letra B

---

20. (FGV / CODEBA – 2016) Analise o seguinte trecho de XML Schema (XSD).

```
<xs:complexType name="TipoEstudante">
 <xs:sequence>
  <xs:element name="nome" type="xs:string"/>
   <xs:element name="sobrenome" type="xs:string"/>
   <xs:element name="notas" type="xs:positiveInteger"/>
 </xs:sequence>
  <xs:attribute name="matricula" type="xs:positiveInteger"/>
</xs:complexType>
```

Assinale o elemento XML cuja definição está de acordo a especificação de "TipoEstudante"

a)
```
<estudante "493">
 <nome>Maria</nome>
 <sobrenome>Ferreira</sobrenome>
 <notas>95</notas>
</estudante>
```

b)
```
<estudante matricula="493">
 <nome>Maria</nome>
 <sobrenome>Ferreira</sobrenome>
 <notas>9.5, 8.8, 10.0</notas>
</estudante>
```

c)
```
<estudante matricula="493">
 <nome>Maria</nome>
 <sobrenome>Ferreira</sobrenome>
</estudante>
```

d)
```
<estudante matricula="493">
 <nome>Maria</nome>
 <sobrenome>Ferreira</sobrenome>
 <notas>95</notas>
</estudante>
```

e)
```
<estudante matricula="493">
 <notas>95, 27, 48</notas>
</estudante>
```

Comentários:

Vejam como não é difícil de entender: o XSD indica que temos um tipo complexo chamado TipoEstudante que contém três elementos: nome (string), sobrenome (string) e notas (inteiro positivo). Além disso, ele informa que há um atributo chamado "matrícula" que também é um

inteiro positivo. Logo, vamos analisar os itens: (a) Errado, falta o atributo "matrícula" e nota é apenas um valor; (b) Errado, nota é um inteiro positivo; (c) Errado, falta o elemento "notas"; (d) Correto; (e) Errado, falta o elemento nome e sobrenome – e nota é apenas um valor.

Gabarito: Letra D

---

21. (FGV / DPE-RO – 2015) Os trechos contendo XML mal-formado, válido e inválido, respectivamente, são:

```
<!DOCTYPE processo [
<!ELEMENT sujeitos (juiz, autor+, reu+)>
<!ELEMENT juiz (#PCDATA)>
<!ELEMENT autor (#PCDATA)>
<!ELEMENT reu (#PCDATA)>
]>
```

e os seguintes trechos de documento XML:

I)
```
<processo>
 <sujeitos>
   <juiz>Dr. Pedro da Silva</juiz>
  <autor>Fulano de Souza</autor>
  <reu>Cicrano Pereira</reu>
 </sujeitos>
</processo>
```

II)
```
<processo>
 <sujeitos>
  <autor>Fulano de Souza</autor>
  <reu>Cicrano Pereira</reu>
 </sujeitos>
</processo>
```

III)
```
<processo>
 <sujeitos>
  <juiz>Dr. Pedro da Silva
  <autor>Fulano de Souza
  <reu>Cicrano Pereira
 </sujeitos>
</processo>
```

a) I, II e III;
b) I, III e II;
c) II, I e III;
d) III, I e II;
e) III, II e I.

Comentários:

Vamos começar analisando o DTD: (1) note que os elementos juiz, autor e réu devem vir necessariamente nessa ordem; (2) note que juiz deve aparecer necessariamente uma vez, e autor e réu podem aparecer uma ou mais vezes. Dito isso, já podemos responder à questão: (I) Trata-se de um trecho bem formado, visto que não possui erros de sintaxe e válido, visto que obedece fielmente ao DTD; (II) Trata-se de um trecho bem formado, visto que não possui erros de sintaxe, mas é inválido, visto que juiz deve aparecer necessariamente uma vez; (III) Trata-se de um trecho mal formado, visto não apresenta as tags de fechamento de juiz, autor e réu, portanto é um trecho consequentemente inválido.

Gabarito: Letra D

---

22. (FGV / TJ-PI – 2015) Num trecho XML, o comentário "Trecho em teste" deve ser introduzido como:

a) < !-- Trecho -- em -- teste -- >
b) < !-- Trecho em teste >
c) < !Trecho em teste >
d) < !--Trecho em teste -->
e) <-- Trecho em teste -->

Comentários:

(a) Errado, não pode haver traços dentro do comentário; (b) Errado, faltam os traços da tag de fechamento; (c) Errado, faltam os traços da tag de abertura e fechamento; (d) Correto; (e) Errado, falta o ponto de exclamação na tag de abertura.

Gabarito: Letra D

---

23. (FGV / DPE-RO – 2015) A representação em XML da agenda de uma pessoa em que o telefone do usuário João está representado como um atributo é:

a)

```
<usuario>
 <nome>Joao</nome>
 <telefone>5555-5555</telefone>
</usuario>
```

b)

```
<telefone nome="João">4444-44444</telefone>
```

c)

```
<nome>joão
 <telefone>3333-3333</telefone>
</nome>
```

d)

```
<usuario>
 <nome>João</nome>
 <atrib>7777-888</atrib>
</usuario>
```

e)

```
<usuario telefone="9999-9999">
 <nome>João</nome>
</usuario>
```

Comentários:

(a) Errado, telefone é um elemento; (b) Errado, telefone é um elemento; (c) Errado, telefone é um elemento; (d) Errado, nem sequer há telefone; (e) Correto, telefone é um atributo do elemento usuário de nome João.

Gabarito: Letra E

---

24. (FGV / PGE-RO – 2015) Analise, abaixo, a lista de definições que podem ser estabelecidas por meio de um esquema (schema) para um documento XML:

I. os elementos que podem ser utilizados;
II. os tipos de dados para elementos e atributos;
III. valores default para elementos e atributos;
IV. espaços reservados para comentários.

Somente estão corretas as afirmativas:

a) I e II;
b) I e III;
c) II e III;
d) I, II e III;
e) II, III e IV.

Comentários:

Um esquema pode definir os elementos que podem ser utilizados, os tipos de dados dos elementos e atributos e os valores default para elementos e atributos, mas não podem definir espaços reservados para comentários.

Gabarito: Letra D

---

25. (FGV / PGE-RO – 2015) São requisitos de um documento XML, EXCETO:

a) deve ter um elemento raiz, responsável por aninhar os demais elementos que formam o documento.
b) deve ser bem formado, ou seja, possuir somente uma tag raiz, possuir tags de fechamento, etc.;
c) pode conter atributos que devem ser únicos dentro do elemento, e seus valores devem estar envoltos em aspas.
d) deve ser válido quando se desejar formalização na representação das informações.
e) deve começar por uma declaração de conteúdo do elemento.

Comentários:

Todos os itens estão perfeitos com exceção do último – não é obrigatório começar com uma declaração de conteúdo do elemento.

Gabarito: Letra E

---

26. (FGV / TJ-GO – 2014) Observe os principais tópicos de um arquivo XML de uma nota fiscal eletrônica.

O atributo *xmlns* define o que é conhecido como:

a) Namespace;
b) Name Source;
c) Named Schema;
d) Name Status;
e) Name Server.

Comentários:

A imagem está péssima, mas o importante é que xmlns significa XML Namespace.

Gabarito: Letra A

---

27. (FGV / AL-MA – 2013) Dentre as alternativas a seguir, selecione aquelas que correspondem a especificações elaboradas com a finalidade de definir regras de validação (esquemas) para documentos XML:

I. XSLT
II. DTD
III. XML Schema

Assinale:

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

Comentários:

DTD e XML Schema têm a finalidade de definir regras de validação (esquemas) para documentos XML. O XSLT é utilizado para transformar documentos XML em outros documentos que aqui não cabe detalhar.

Gabarito: Letra D

---

28. (FGV / AL-MA – 2013) O scritp XML a seguir, que faz referência ao esquema verifica.xsd, está sintaticamente incorreto porque UTF-8 não é suportado no XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<addresses xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation='verifica.xsd'>
  <endereco>
  <nome>JoaoTestador</nome>
  <rua>CPD Informatica 00</rua>
</endereco>
```

Comentários:

Vamos por partes: em primeiro lugar, veja que a questão escreveu scritp em vez de script – não fui eu que errei; em segundo lugar, veja que realmente há uma referência a um esquema externo chamado verifica.xsd; por fim, todo processador XML precisa necessariamente suportar as codificações UTF-8 e UTF-16 (e pode, inclusive, suportar outros tipos de codificação de caracteres).

Gabarito: Errado

QUESTÕES COMENTADAS – CESGRANRIO

29. (CESGRANRIO / IPEA - 2024) O Ipea pretende publicar a Tabela de índices de custos e preços abaixo, relativa aos meses do primeiro trimestre de 2023. Para publicá-la, é necessário colocá-la no formato XML. Desconsiderando - se a parte de DOCTYPE e Style, ao colocar essa Tabela no formato XML, obtém-se:

a)
```
<Indices>
  <Indice> ICTI IPCA IGPM </Indice>
  <Indice> 182,34 0,53 0,21 </Indice >
  <Indice> 183,16 0,84 - 0,06 </Indice >
<Indice> 183,34 0,76 0,05 </Indice >
```

b)
```
<Indices>
 <Indice>
   <ICTI> 182,34 </ICTI>
   <IPCA> 0,53 </IPCA>
   <IGPM> 0,21 </IGPM>
 </Indice>
 <Indice>
  <ICTI> 183,16 </ICTI>
  <IPCA> 0,84 </IPCA>
  <IGPM> - 0,06 </IGPM>
 </Indice>
 <Indice>
  <ICTI> 183,34 </ICTI>
  <IPCA> 0,76 </IPCA>
  <IGPM> 0,05 </IGPM>
 </Indice>
</Indices>
```

c)
```
<Indices>
 <1>
 <ICTI> 182,34 </ICTI>
 <IPCA> 0,53 </IPCA>
 <IGPM> 0,21 </IGPM>
 <2>
 <ICTI> 183,16 </ICTI>
```

<IPCA> 0,84 </IPCA>
<IGPM> - 0,06 </IGPM>
<3>
<ICTI> 183,34 </ICTI>
<IPCA> 0,76 </IPCA>
<IGPM> 0,05 </IGPM>
</Indices>

d)
<Indices>
<ICTI> 182,34 </ICTI>
<IPCA> 0,53 </IPCA>
<IGPM> 0,21 </IGPM>
</Indices>
<Indices>
<ICTI> 183,16 </ICTI>
<IPCA> 0,84 </IPCA>
<IGPM> - 0,06 </IGPM>
</Indices>
<Indices>
<ICTI> 183,34 </ICTI>
<IPCA> 0,76 </IPCA>
<IGPM> 0,05 </IGPM>
</Indices>

e)
<Indices>
<insert>
<1>
<ICTI> 182,34 </ICTI>
<IPCA> 0,53 </IPCA>
<IGPM> 0,21 </IGPM>
</1>
<insert>
<2>
<ICTI> 183,16 </ICTI>
<IPCA> 0,84 </IPCA>
<IGPM> - 0,06 </IGPM>
</2>
<insert>
<3>
<ICTI> 183,34 </ICTI>
<IPCA> 0,76 </IPCA>
<IGPM> 0,05 </IGPM>

```
  </3>
</Indices>
```

Comentários:

(a) Errado. A alternativa apresenta um formato incorreto. As tags Indice não possuem atributos para identificar os dados;

(b) Correto. A alternativa utiliza tags para cada índice (ICTI, IPCA e IGPM) dentro de cada mês, organizando os dados de forma clara e legível;

(c) Errado. A alternativa utiliza números como identificadores de meses, o que pode dificultar a interpretação do conteúdo. A utilização de tags específicas para cada mês seria mais adequada;

(d) Errado. A alternativa cria um bloco Indices para cada mês, o que torna o código redundante e dificulta a leitura;

(e) Errado. A alternativa utiliza tags insert e números como identificadores de meses, o que torna o código mais complexo e menos intuitivo.

Gabarito: Letra B

---

30. **(CESGRANRIO / TRANSPETRO – 2023)** Um profissional de Informática está trabalhando em um projeto que envolve a manipulação de documentos XML. Ele precisa garantir que os documentos XML estejam bem-formados e válidos, de acordo com as especificações do XML 1.1. Uma das regras que ele deverá seguir para garantir que um documento XML 1.1 seja válido é que o(s):

a) documento pode ter um ou mais elementos raiz.
b) documento deve começar com uma declaração XML.
c) nomes dos elementos são insensíveis a maiúsculas e minúsculas.
d) atributos devem ter o mesmo nome se estiverem no mesmo elemento.
e) comentários XML devem aparecer como atributos de uma etiqueta (tag).

Comentários:

(a) Errado. O XML 1.1 permite apenas um elemento raiz;

(b) Correta. Um documento XML 1.1 deve começar com uma declaração XML que especifica a versão do XML e o tipo de codificação de caracteres utilizada;

(c) Errado. O XML 1.1 é sensível a maiúsculas e minúsculas nos nomes dos elementos e atributos;

(d) Errado. Atributos com o mesmo nome no mesmo elemento devem ter valores diferentes. Essa regra não garante a validade do documento;

(e) Errado. Comentários XML não podem ser atributos de uma tag. Eles devem ser inseridos entre as tags ou antes da declaração XML.

Gabarito: Letra B

---

31. **(CESGRANRIO / BASA - 2022)** Um projetista de sistemas está desenvolvendo um sistema e precisou programar um arquivo XSLT. Neste arquivo, ele precisou inserir um elemento para aplicar uma regra de modelo, a partir de uma folha de estilo importada, ao invés de uma regra equivalente, a partir da folha de estilo principal, mas sem que este elemento apareça como o primeiro nó filho de .

Para este caso, o elemento que deve ser inserido para aplicar tal regra nesse arquivo XSLT é o

a) apply-imports
b) apply_templates
c) imports
d) include
e) Template

Comentários:

(a) Correto. O elemento <xsl:apply-imports> é usado dentro de um template para aplicar as regras de template importadas. Se houver regras de template equivalentes na folha de estilo principal, <xsl:apply-imports> permite que o processador XSLT ignore essas regras principais em favor das regras importadas, adequando-se à descrição fornecida;

(b) Errado. <xsl:apply-templates> é usado para aplicar regras de template a nós selecionados, mas não especificamente para escolher regras de templates importados sobre os da folha de estilo principal;

(c) Errado. "imports" não é um elemento XSLT válido. A importação de folhas de estilo em XSLT é feita com o elemento <xsl:import>;

(d) Errado. <xsl:include> é usado para incluir templates de outra folha de estilo XSLT dentro da folha de estilo atual. Diferente de <xsl:import>, ele não tem a funcionalidade específica de priorizar regras de templates importados sobre os da folha de estilo principal;

(e) Errado. <xsl:template> é o elemento usado para definir uma regra de template. Sozinho, não serve para aplicar uma regra de modelo de uma folha de estilo importada sobre a folha de estilo principal.

Gabarito: Letra A

32. (CESGRANRIO / BASA – 2022) Ao desenvolver um sistema de notícias, a empresa X decidiu manter as notícias em um formato XML, como o do exemplo a seguir:

```
<?xml version="1.0"?>
<news>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</news>
```

Mais tarde, entendeu que, para esse formato exemplificado acima, seria melhor definir um esquema em XSD. Que fragmento de código XSD deve conter esse esquema para permitir que o exemplo apresentado seja validado corretamente, quando nele for incluída a referência ao esquema completo?

a)
```
<xs:element name="news">
   <xs:complexType>
    <xs:sequence> <xs:element name="heading" type="xs:string"/>
     <xs:element name="body" type="xs:string"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
```

b)
```
<xs:element name="news">
    <xs:sequence>
     <xs:element name="heading" type="xs:string"/>
     <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:element>
```

c)
```
<xs:element name="news">
     <xs:element name="heading" type="xs:string"/>
    <xs:element name="body" type="xs:string"/>
  </xs:element>
```

d)
```
<!ELEMENT news (heading, body)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body (#PCDATA)>
```

e)
```
<!ELEMENT news (heading, body)>
  <!ELEMENT heading (text)>
```

<!ELEMENT body (text)>

Comentários:

Vocês lembram que a linguagem de esquema XML é definida pelo padrão XML Schema Definition (XSD)? É uma maneira de descrever a estrutura e restringir o conteúdo de documentos XML. Após o lembrete, vamos juntos analisar as alternativas.

(a) Correto. Este fragmento define um esquema XSD completo para o exemplo de notícia. Ele inclui: Elemento news como raiz, com tipo complexo; Sequência de dois elementos filhos: heading e body; Definição do tipo de cada elemento filho como xs:string.

(b) Errado. Este fragmento define apenas a sequência de elementos, mas não seus tipos; (c) Errado. Este fragmento define os elementos, mas não seus tipos; (d) Errado. Este fragmento usa DTD (Document Type Definition), que não é XSD; (e) Errado. Este fragmento define os elementos com conteúdo text, o que não valida o exemplo dado, que possui conteúdo específico em body.

Gabarito: Letra A

---

33. **(CESGRANRIO / BASA – 2021)** Ao participar de uma equipe para desenvolvimento de um website para a intranet do banco em que trabalhava, um programador teve como missão criar uma tabela HTML a partir de um arquivo XML que indicava clientes e seus saldos.

O fragmento de XML a seguir é um exemplo da estrutura do XML do arquivo que conterá os dados:

```
<?xml version="1.0" encoding="UTF - 8"?>
<clientes>
  <cliente>
 <nome>Ana Zurique</nome>
  <saldo>3000</saldo>
 </cliente>
 <cliente>
  <nome>Bernardo Washington</nome>
  <saldo>4500</saldo>
</cliente>
<cliente>
  <nome>Carlos York</nome>
  <saldo>12345</saldo>
</cliente>
</cliente>
```

Para esse arquivo, a tabela gerada deve ter a seguinte aparência:

Cliente	Saldo
Ana Zurique	3000
Bernardo Washington	4500
Carlos York	12345

Inicialmente, o programador construiu o seguinte arquivo em XSLT:

```
<?xml version="1.0" encoding="UTF - 8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
 <table border="1">
<tr>
 <th>Cliente</th>
 <th>Saldo</th>
 </tr>
 <! - - Código Para os Dados - - >
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Que sequência de instruções deve substituir o comentário , de forma a gerar a tabela no formato apresentado?

a)

```
 <xsl:for - each select="clientes/cliente/">
<tr>
<td><xsl:value - of select="nome"/></td>
<td><xsl:value - of select="saldo"/></td>
</tr>
</xsl:for - each>
```

b)

```
<xsl:for - each select="clientes/cliente">
<tr>
<td><xsl:value - of select="clientes/cliente/nome"/></td>
<td><xsl:value - of select="clientes/cliente/saldo"/></td>
</tr>
</xsl:for - each>
```

c)
<xsl:for - each select="clientes/cliente">
<tr>
<td><xsl:value - of select="nome"/></td>
<td><xsl:value - of select="saldo"/></td>
</tr>
</xsl:for - each>

d)
<xsl:for - each select="clientes/">
<tr>
<td><xsl:value - of select="cliente/nome"/></td>
<td><xsl:value - of select="cliente/saldo"/></td>
</tr>
</xsl:for - each>

e)
<xsl:for - each select="clientes">
<tr>
<td><xsl:value - of select="cliente/nome"/></td>
<td><xsl:value - of select="cliente/saldo"/></td>
</tr>
</xsl:for - each>

Comentários:

xsl:for-each e xsl:value-of são instruções usadas em XSLT, linguagem usada para transformar documentos XML em outros formatos, como HTML, texto ou até mesmo em outros documentos XML. Essas instruções são utilizadas para percorrer e manipular elementos XML durante o processo de transformação.

(a) Errado. Há um erro no XPath utilizado no xsl:for-each. A barra final ("/") após "clientes/cliente/" não é necessária e causa a seleção incorreta dos elementos. O XPath "clientes/cliente/" seleciona todos os elementos "cliente" que são filhos diretos do elemento "clientes".

(b) Errado. Esta sequência é redundante, pois seleciona "clientes/cliente/nome" e "clientes/cliente/saldo" para cada cliente.

(c) Correta. Esta sequência utiliza: xsl:for-each para iterar sobre cada cliente no XML e xsl:value-of para extrair o nome e o saldo de cada cliente.

(d) Errado. Esta sequência seleciona "cliente/nome" e "cliente/saldo", mas não há garantia de que esses elementos existam no contexto atual.

(e) Errado. Esta sequência é semelhante à alternativa d) e pode gerar erros se "cliente" não for o elemento filho direto de "clientes".

Gabarito: Letra C

---

34. (CESGRANRIO / CEF – 2021) Um arquivo, contendo um documento XML, contém exatamente a seguinte informação:

```
<?xml version="1.0"?>
<PEDIDOS>
<PEDIDO>
<TITULO>Pedido de Empréstimo</TITULO>
<REQUERENTE>José da Silva</REQUERENTE>
<CPF>999.999.999 - 99</CPF>
<VALOR>20000</VALOR>
<PEDIDO>
<PEDIDOS>
```

A partir desse documento apenas, um processador XML pode garantir que o arquivo é

a) bem - formado, apenas
b) bem - formado e normalizado
c) bem - formado e válido
d) normalizado, apenas
e) válido, apenas

Comentários:

É importante ressaltar que um processador XML pode verificar se um arquivo é bem-formado, mas não pode garantir que ele esteja normalizado ou válido sem informações adicionais. A normalização e a validação são etapas opcionais que podem ser utilizadas para melhorar a qualidade e a confiabilidade de um documento XML.

(a) Correto. Um processador XML pode garantir que o arquivo é bem-formado, porque possui uma única tag raiz (PEDIDOS). Todas as tags são fechadas corretamente e a estrutura hierárquica das tags está perfeita; (b) O arquivo não está normalizado, pois a tag PEDIDO é repetida; (c) O arquivo não é válido, pois não há uma DTD (Document Type Definition) ou esquema XSD para verificar sua validade; (d) O arquivo não está normalizado; (e) O arquivo não é válido.

Gabarito: Letra A

---

35. (CESGRANRIO / Caixa – 2021) As fontes (feed) RSS devem todas fornecer informações em

a) CSS
b) HTML 1.0
c) HTML 1.1
d) SOAP
e) XML

Comentários:

A fonte (feed) RSS deve fornecer informações em XML (eXtensible Markup Language). O RSS (Really Simple Syndication) é uma tecnologia de distribuição de conteúdo web atualizado de um site para outros sites ou para usuários que assinam o feed. O formato padrão para estruturar e representar essas informações é o XML devido à sua flexibilidade e capacidade de representar dados de forma estruturada e legível por máquina.

Gabarito: Letra E

---

36. (CESGRANRIO / TRANSPETRO – 2018) Considerando a linguagem XML, qual é o exemplo correto de uso de um atributo chamado "src" que recebe o valor "computador.gif" em um elemento de nome "img"?

a)
b) <img src="computador.gif "/img>
c) <img> "src=computador.gif " </img>
d) <img> <src> computador.gif </src> </img>
e) <img> src="computador.gif " </img>

Comentários:

(a) Correto; (b) Errado, a tag de fechamento deve ser /> ou </img>; (c) Errado, a tag de abertura deve ser <img e, não, <img>; (d) Errado, src é um atributo e, não, um elemento; (e) Errado, a tag de abertura deve ser <img e, não, <img>.

Gabarito: Letra A

---

37. (CESGRANRIO / PETROBRAS – 2018) Qual linguagem de marcação, fundamental para o estabelecimento de serviços Web, que compõe uma Arquitetura Orientada a Serviços, é usada para que dados sejam apresentados, comunicados e armazenados?

a) HTML;
b) XML;
c) JAVA;
d) JAVASCRIPT;

e) C#.

Comentários:

(a) Errado. HTML é uma linguagem de marcação utilizada para criar e estruturar páginas na Web, mas não é especificamente projetada para comunicação de dados em uma Arquitetura Orientada a Serviços;

(b) Correto. XML é uma linguagem de marcação que permite a criação de documentos com dados estruturados. É amplamente utilizada em serviços Web e Arquiteturas Orientadas a Serviços (SOA) para a comunicação, apresentação e armazenamento de dados de maneira padronizada e independente de plataforma;

(c) Errado. Java é uma linguagem de programação, não uma linguagem de marcação. Embora seja amplamente utilizada para desenvolver aplicações web e serviços, não se encaixa no contexto da questão;

(d) Errado. JavaScript é uma linguagem de programação usada principalmente para criar scripts do lado do cliente em páginas web. Não é uma linguagem de marcação nem é usada primariamente para comunicação de dados em SOA;

(e) Errado. C# é uma linguagem de programação orientada a objetos desenvolvida pela Microsoft, usada para desenvolver uma grande variedade de aplicações, incluindo serviços web, mas não é uma linguagem de marcação.

Gabarito: Letra B

---

38. (CESGRANRIO / BASA – 2018) Considere o esquema XML a seguir:

```
<xs:elemente name="rectangle" type = "area"/>
<xs:complexType name = "area">
  <xs:attribute name="x1" type="xs:decimal" />
  <xs:attribute name="y1" type="xs:decimal" />
  <xs:attribute name="x2" type="xs:decimal" />
  <xs:attribute name="y2" type="xs:decimal" />
<xs:complexType>
```

Um elemento XML válido, segundo esse esquema, é:

a) <area><x1>1</x1><y1>1</y1><x2>2</x2><y2>3</y2></area>
b) <area x1="1" y1="1" x2="4" y2="5" />
c) <rectangle x1="5" y1="4" x2="1" y2="1"/>
d) <area><x1>4</x1><y1>4</y1><x2>2</x2><y2>3</y2></area>
e) <rectangle><x1>1</x1><y1>1</y1><x2>2</x2><y2>3</y2></rectangle>

Comentários:

(a) Errado. Essa alternativa utiliza elementos para representar os atributos (x1, y1, x2, y2), o que contradiz a definição no esquema XML que especifica esses como atributos do elemento, não como subelementos;

(b) Errado. Apesar de b representar os dados como atributos, o elemento correto definido pelo esquema é rectangle, não área;

(c) Correto. Esta alternativa está correta porque rectangle é o elemento definido no esquema, e os atributos x1, y1, x2, y2 são especificados de acordo com as regras do esquema;

(d) Errado. Assim como em a, esta alternativa representa os valores como subelementos, o que não está de acordo com a definição do esquema;

(e) Errado. Apesar de usar o nome do elemento rectangle, esta alternativa também representa os valores como subelementos, o que vai contra o esquema que define esses valores como atributos.

Gabarito: Letra C

---

39. **(CESGRANRIO / TRANSPETRO – 2018)** Documentos XML são estruturados segundo uma hierarquia de unidades informacionais chamadas de nós. Qual tecnologia XML fornece ao desenvolvedor uma API para adicionar, editar e remover esses nós?

a) XMI
b) XSDL
c) XSLT
d) XML DOM
e) XML Schema

Comentários:

(a) Errado. XMI (XML Metadata Interchange) é uma especificação para troca de metadados via XML. Não é uma API para manipulação direta de documentos XML;

(b) Errado. Não existe uma tecnologia XML conhecida como XSDL. Pode haver confusão com WSDL (Web Services Description Language) ou XSD (XML Schema Definition), mas nenhum desses é uma API para manipulação de nós em documentos XML;

(c) Errado. XSLT (eXtensible Stylesheet Language Transformations) é uma linguagem para transformação de documentos XML em outros tipos de documentos (XML, HTML, texto puro, etc.). Embora possa alterar a estrutura de um documento XML, não é uma API para adicionar, editar ou remover nós diretamente;

(d) Correto. XML DOM (Document Object Model) é uma interface de programação de aplicações (API) que permite aos desenvolvedores adicionar, modificar e remover nós em documentos XML. O DOM representa o documento como uma árvore de nós, onde cada nó pode ser manipulado programaticamente;

(e) Errado. XML Schema é uma linguagem para definir a estrutura e restringir o conteúdo de documentos XML. Não fornece uma API para a manipulação de nós dentro de documentos XML.

Gabarito: Letra D

---

40. (CESGRANRIO / BASA – 2014) Sabendo que um arquivo XML está sintaticamente correto e que pode ser consumido ou processado por um parser XML, de acordo com a especificação XML, pode-se afirmar, com certeza, que ele é:

a) autorizado
b) validado
c) certificado
d) compilado
e) bem formado

Comentários:

(a) Errado. "Autorizado" não é um termo usado na especificação XML para descrever a correção sintática ou a capacidade de um arquivo XML ser processado;

(b) Errado. "Validado" refere-se a um arquivo XML que foi verificado e está em conformidade com um esquema ou DTD (Document Type Definition). A validação vai além da correção sintática, verificando também a conformidade estrutural com uma definição de esquema;

(c) Errado. "Certificado" não é um termo aplicável ao contexto de arquivos XML em relação à sua estrutura ou sintaxe;

(d) Errado. "Compilado" é um termo geralmente associado a linguagens de programação que necessitam de compilação para serem executadas. Arquivos XML são interpretados por parsers e não são compilados;

(e) Correto. "Bem formado" é o termo usado para descrever um arquivo XML que segue as regras sintáticas da especificação XML. Isso significa que o arquivo pode ser adequadamente interpretado por um parser XML, indicando que todas as tags estão corretamente abertas e fechadas, os atributos estão corretamente citados, e o documento possui uma única raiz.

Gabarito: Letra E

---

41. (CESGRANRIO / AGC – 2014) Analise o seguinte DTD

```
<?xml version="1.0" ?>
<!DOCTYPE A [
 <!ELEMENT A (B,C)>
 <!ELEMENT B (#PCDATA)>
 <!ELEMENT C (D?,E)>
 <!ELEMENT D (#PCDATA)>
 <!ELEMENT E (#PCDATA)>
]>
```

Segundo o DTD acima, que documento XML NÃO é válido?

a) <A><B>Teste</B> <C><E>Teste</E> </C></A>
b) <A><B>Teste</B> <C><D>Teste</D> <E>Teste</E></C></A>
c) <A><B>Teste</B> <C>Teste<D>Teste</D> <E>Teste</E></C></A>
d) <A><B> </B> <C> <D>Teste</D><E>Teste</E></C></A>
e) <A><B></B><C><D></D><E></E></C></A>

Comentários:

(a) Correto. Está de acordo com o DTD. Elemento A contém B e C, e C contém E como permitido; (b) Correto. Está de acordo com o DTD. Elemento A contém B e C, C contém opcionalmente D seguido por E, conforme especificado; (c) Errado. O conteúdo diretamente dentro de C deveria ser elementos D (opcional) seguido por E, mas o exemplo inclui texto diretamente dentro de C ("Teste" antes de <D>Teste</D>), o que viola o DTD; (d) Correto. Está de acordo com o DTD. A contém B e C, onde C tem opcionalmente D e obrigatoriamente E; (e) Correto. Está de acordo com o DTD. Apesar de D e E estarem vazios, eles estão presentes na ordem correta dentro de C.

Gabarito: Letra C

---

42. (CESGRANRIO / BASA - 2014) Seja o arquivo XML abaixo:

```
<?xml version=”1.0” encoding=”UTF - 8”?>
<T><P N=”1”>
<K N=”1” M=”G”>Texto</K>
<K N=”2” M=”H”>Texto</K>
<K></K></P>
<P><F>Texto</F></P>
</T>
```

Que DTD permite que esse arquivo seja considerado válido?

a)
```
<!ATTLIST ( P+ ) >
<!ATTLIST P ( K?, F? ) >
<!ELEMENT P N NMTOKEN #IMPLIED >
<!ATTLIST K ( #PCDATA ) >
<!ELEMENT K M NMTOKEN #IMPLIED >
<!ELEMENT K N NMTOKEN #IMPLIED >
<!ATTLIST F ( #PCDATA ) >
```

b)
```
<!ELEMENT T ( P+ ) >
<!ELEMENT P ( K*, F? ) >
<!ATTLIST P N NMTOKEN #IMPLIED >
<!ELEMENT K ( #PCDATA ) >
<!ATTLIST K M NMTOKEN #IMPLIED >
<!ATTLIST K N NMTOKEN #IMPLIED >
<!ELEMENT F ( #PCDATA ) >
```

c)
```
<!ELEMENT T ( P+ ) >
<!ELEMENT P ( K*, F? ) >
<!ATTLIST P N NMTOKEN #REQUIRED >
<!ELEMENT K ( #PCDATA ) >
<!ATTLIST K M NMTOKEN #IMPLIED >
<!ATTLIST K N NMTOKEN #IMPLIED >
<!ELEMENT F ( #PCDATA ) >
```

d)
```
<!ELEMENT T ( P+ ) >
<!ELEMENT P ( K?, F? ) >
<!ATTLIST P N NMTOKEN #IMPLIED >
<!ELEMENT K ( #PCDATA ) >
<!ATTLIST K M NMTOKEN #IMPLIED >
<!ATTLIST K N NMTOKEN #IMPLIED >
<!ELEMENT F ( #PCDATA ) >
```

e)
```
<!ATTLIST ( P+ ) >
<!ATTLIST P ( K*, F? ) >
<!ELEMENT P N NMTOKEN #IMPLIED >
<!ATTLIST K ( #PCDATA ) >
<!ELEMENT K M NMTOKEN #IMPLIED >
<!ELEMENT K N NMTOKEN #IMPLIED >
<!ATTLIST F ( #PCDATA ) >
```

Comentários:

Analisando o XML, podemos entender sua estrutura e os elementos envolvidos:

- O elemento T é o elemento raiz do documento XML.
- Dentro do elemento T, temos dois elementos P.

- Cada elemento P pode conter vários elementos K e opcionalmente um elemento F.
- Os elementos K podem conter texto e têm os atributos N e M.
- O elemento F pode conter apenas texto.

Com base nessa análise, podemos construir o DTD correspondente que valida o XML:

```
<!ELEMENT T (P+)>
<!ELEMENT P (K*, F?)>
<!ATTLIST P N NMTOKEN #IMPLIED>
<!ELEMENT K (#PCDATA)>
<!ATTLIST K N NMTOKEN #IMPLIED>
<!ATTLIST K M NMTOKEN #IMPLIED>
<!ELEMENT F (#PCDATA)>
```

Nesse DTD:

- T é definido como tendo um ou mais elementos P.
- P pode conter zero ou mais elementos K e opcionalmente um elemento F.
- P pode ter um atributo N.
- K contém dados de texto (#PCDATA).
- K pode ter atributos N e M.
- F contém dados de texto (#PCDATA).

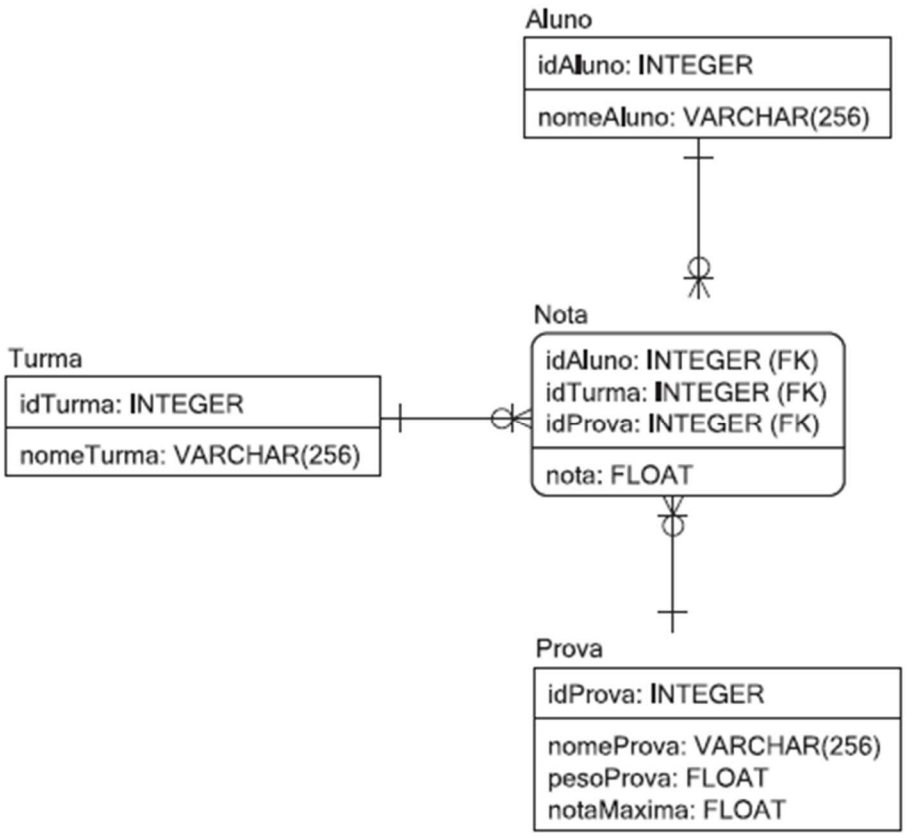
As demais opções não estão corretas pois apresentam erros de sintaxe ou não abordam adequadamente a estrutura do XML.

Gabarito: Letra B

---

43. (CESGRANRIO / CEFET RJ - 2014) Uma universidade decidiu alterar seu sistema acadêmico, atualmente escrito em Delphi, para aceitar uma interface Web. Para isso, decidiu adotar as tecnologias Ajax e PHP.

A primeira parte do trabalho será alterar o subsistema de avaliação, chamado de NOTAS. O modelo de dados atual desse subsistema é bastante simples, e é descrito pelo modelo diagrama a seguir, que usa a notação da Engenharia da Informação.



Que fragmento de código XML o sistema NOTAS pode usar para representar corretamente uma linha da tabela Turma?

a) </TURMA></IDTURMA>1<IDTURMA></NOMETURMA>Cálculo<NOMETURMA><TURMA>
b) <TURMA><IDTURMA><NOMETURMA>1</IDTURMA>Cálculo</NOMETURMA></TURMA>
c) <TURMA><IDTURMA>1</IDTURMA><NOMETURMA>Cálculo</NOMETURMA></TURMA>
d) <TURMA><IDTURMA>1<IDTURMA><NOMETURMA>Cálculo<NOMETURMA><TURMA>
e)
<TURMA/><IDTURMA/>Cálculo</IDTURMA><NOMETURMA/>1</NOMETURMA></TURMA>

Comentários:

(a) Errado. Esta alternativa está incorreta porque todas as tags estão fechadas antes mesmo de serem abertas corretamente. Além disso, a estrutura está confusa e desorganizada;

(b) Errado. Nesta alternativa, a tag <NOMETURMA> está fechada antes da tag <IDTURMA>, o que viola a ordem correta de fechamento das tags. Isso resulta em uma estrutura incorreta;

(c) Esta alternativa está correta. As tags estão corretamente abertas e fechadas na ordem adequada, e os elementos IDTURMA e NOMETURMA estão corretamente aninhados dentro do elemento TURMA;

(d) Errado. Todas as tags estão abertas, mas nenhuma delas está sendo fechada, o que resulta em uma estrutura incorreta e inválida;

(e) Errado. Nesta alternativa, todas as tags são auto-fechadas com a notação />, o que não é apropriado para o contexto apresentado. Além disso, a ordem das tags está incorreta e não segue a estrutura esperada.

Gabarito: Letra C

---

44. (CESGRANRIO / IBGE - 2013) O gerente acadêmico de uma universidade solicitou ao setor de tecnologia da informação que fosse desenvolvida uma ferramenta que permitisse a distribuição dos currículos dos professores em diferentes formatos, uma vez que isso é essencial para promover o intercâmbio de informações entre diferentes instituições de ensino do Brasil e do exterior. Sabendo-se que os currículos que estão armazenados na base de dados da universidade são documentos XML válidos, qual tecnologia XML deve ser empregada na construção dessa ferramenta?

a) XSD
b) PDF
c) XSL
d) XKMS
e) HTML

Comentários:

(a) Errado. XSD (XML Schema Definition) é usado para definir a estrutura e validar documentos XML, não para transformá-los em diferentes formatos.

(b) Errado. PDF é um formato de arquivo para apresentação de documentos de forma independente de software, hardware ou sistema operacional, mas não é uma tecnologia XML utilizada para transformar ou distribuir documentos XML em diferentes formatos.

(c) Correto. XSL (eXtensible Stylesheet Language), especialmente em conjunto com XSLT (XSL Transformations), é a tecnologia XML projetada para transformar documentos XML em outros formatos de documento. Ela permite que os dados XML sejam apresentados em diferentes formatos, como HTML para web, PDF para documentos impressos, ou outros formatos XML específicos de domínio, atendendo à necessidade de distribuição de currículos em diversos formatos.

(d) Errado. XKMS (XML Key Management Specification) é relacionado à gestão de chaves criptográficas para serviços web, não à transformação ou distribuição de documentos XML em diferentes formatos.

(e) Errado. HTML é uma linguagem de marcação usada para criar páginas web. Embora possa ser um dos alvos da transformação dos documentos XML, não é a tecnologia que permite essa transformação.

Gabarito: Letra C

---

45. (CESGRANRIO / LIQUIGÁS - 2013) Muitas tecnologias usadas pela indústria de software favorecem a implantação de melhorias na gestão de processos integrados de negócios. Um exemplo disso é o uso de:

a) softwares abertos.
b) bancos de dados relacionais.
c) computação móvel em larga escala.
d) orientação a objetos como paradigma de desenvolvimento.
e) XML para a troca de informações entre sistemas.

Comentários:

(a) Errado. Softwares abertos (ou software livre) podem contribuir para a flexibilidade e redução de custos no desenvolvimento de sistemas. No entanto, essa alternativa não destaca diretamente a gestão de processos de negócios integrados;

(b) Errado. Bancos de dados relacionais são fundamentais para armazenamento e consulta de dados de forma estruturada, mas o benefício mais direto na gestão de processos integrados de negócios não é tão específico quanto o oferecido por outras tecnologias listadas;

(c) Errado. Computação móvel em larga escala permite o acesso a sistemas e dados de qualquer lugar, o que é certamente benéfico para negócios. No entanto, não foca especificamente na integração de processos de negócios como outras opções;

(d) Errado. A orientação a objetos é um paradigma de desenvolvimento que ajuda na organização e estruturação do código fonte, melhorando a manutenção e compreensão dos sistemas. No entanto, não é a tecnologia mais diretamente associada à melhoria da gestão de processos de negócios integrados;

(e) Correto. XML (eXtensible Markup Language) é uma tecnologia chave para a troca de informações entre sistemas, facilitando a integração de dados entre diferentes plataformas e sistemas. Isso é essencial para a gestão de processos de negócios integrados, pois permite que sistemas distintos se comuniquem de forma eficiente, compartilhando dados de negócios em um formato padronizado.

Gabarito: Letra E

---

46. (CESGRANRIO / AGC – 2012) Um documento XML bem formado (well - formed) segue as restrições de sintaxe definidas pela especificação XML.

PORQUE

Um documento XML bem formado deve, necessariamente, estar em conformidade com uma definição em DTD (Document Type Definition) ou em XML Schema.

Analisando - se as afirmações acima, conclui-se que:

a) as duas afirmações são verdadeiras, e a segunda justifica a primeira.
b) as duas afirmações são verdadeiras, e a segunda não justifica a primeira.
c) a primeira afirmação é verdadeira, e a segunda é falsa.
d) a primeira afirmação é falsa, e a segunda é verdadeira.
e) as duas afirmações são falsas.

Comentários:

A primeira afirmação é verdadeira. Um documento XML bem formado é aquele que segue as regras básicas de sintaxe conforme definido pela especificação XML. Isso inclui, mas não se limita a, ter uma única raiz, usar corretamente os fechamentos de tags, citar atributos apropriadamente, entre outros requisitos sintáticos.

A segunda afirmação é falsa. Um documento XML pode ser bem formado sem necessariamente estar em conformidade com um DTD ou XML Schema. A conformidade com DTD ou XML Schema diz respeito à validade do documento XML, que é uma noção diferente de estar bem formado. Estar bem formado é um requisito prévio para a validação, mas um documento pode ser bem formado sem ser validado contra uma DTD ou um XML Schema.

Gabarito: Letra C

---

47. (CESGRANRIO / Petrobras – 2012) Solicitado a preparar um arquivo de teste em XML para um sistema de controle de pedidos de uma distribuidora de petróleo, um analista de sistemas gerou o seguinte documento:

```
< ? xml version="1.0" encoding="UTF-8"? >
< ! DOCTYPE cliente SYSTEM "C:\postos.dtd" >
< cliente >
    < posto >
        < cnpj >
        53.726.891/0001-24
< /cnpj >
        < pedidos >
            < pedido >
```

```
                    < produto >
                Gasolina
< /produto >
                    < quantidade >
                10.000
< /quantidade >
                < /pedido >
                < pedido >
                    < produto >
                Gasolina
< /produto >
                < /pedido >
            < /pedidos >
        < /posto >
< /cliente >
```

Considere o DTD abaixo, salvo no arquivo C:\postos.dtd.

```
< ? xml version="1.0" encoding="UTF-8"? >
< ! ELEMENT quantidade (#PCDATA) >
< ! ELEMENT produto (#PCDATA) >
< ! ELEMENT posto (cnpj,pedidos*) >
< ! ELEMENT pedidos (pedido*) >
< ! ELEMENT pedido (produto, quantidade)m>
< ! ELEMENT cnpj (#PCDATA) >
< ! ELEMENT cliente (posto) >
```

O arquivo preparado pelo analista está em

a) formato diferente do XML.
b) XML, mas não é válido e não é bem-formado.
c) XML, é bem-formado, mas não é válido.
d) XML, é válido, mas não é bem-formado.
e) XML, é válido e bem-formado.

Comentários:

A estrutura do documento XML parece seguir as regras de bem-formação, com tags de abertura e fechamento correspondentes, declaração de tipo de documento (DOCTYPE) e codificação (*encoding*) apropriada. Isso sugere que o documento é bem formado. No entanto, ao analisar a conformidade com o DTD, percebe-se uma incompatibilidade. O DTD especifica que cada <pedido> deve conter um <produto> e uma <quantidade>. No entanto, o segundo <pedido> no documento XML não inclui uma tag <quantidade>, o que viola a definição do DTD.

Logo, enquanto o documento é XML e bem-formado (satisfaz as regras básicas de sintaxe do XML), ele não é válido pois não atende completamente à estrutura definida pelo DTD fornecido.

Gabarito: Letra C

48.(CESGRANRIO / Liquigás – 2012) Com a proliferação de aplicações e serviços utilizados na Internet, o conjunto geral de marcadores presente na linguagem HTML começou a se tornar restritivo, e a necessidade de extensões para criar novos tipos de marcadores começou a surgir. Uma das soluções adotadas pelo W3C foi padronizar uma nova linguagem com a capacidade de ser extensível, sobre a qual rótulos pudessem ser criados de acordo com a necessidade das aplicações. De fato, tal linguagem é muito mais uma metalinguagem, no sentido de que, a partir dela, outras linguagens (até mesmo a própria HTML) com suas marcações poderiam ser geradas. Essa metalinguagem é conhecida como:

a) UML
b) WML
c) XML
d) VML
e) SVG

Comentários:

(a) Errado. UML (Unified Modeling Language) é uma linguagem de modelagem para especificação, visualização, construção e documentação dos artefatos de sistemas de software, mas não é uma metalinguagem para a criação de linguagens de marcação;

(b) Errado. WML (Wireless Markup Language) é uma linguagem de marcação baseada em XML usada para criar páginas que podem ser acessadas em dispositivos móveis via WAP (Wireless Application Protocol), mas não é uma metalinguagem para a criação de outras linguagens;

(c) Correto. XML (eXtensible Markup Language) é uma metalinguagem que permite a definição de outras linguagens de marcação para necessidades específicas. XML foi projetado para ser extensível e transportável, permitindo que desenvolvedores criem suas próprias tags para atender às necessidades específicas das aplicações;

(d) Errado. VML (Vector Markup Language) é uma linguagem de marcação XML para gráficos vetoriais bidimensionais, mas não é uma metalinguagem destinada à criação de outras linguagens;

(e) Errado. SVG (Scalable Vector Graphics) é uma linguagem de marcação XML para descrever gráficos vetoriais bidimensionais, tanto estáticos quanto animados, mas não serve como uma metalinguagem para gerar novas linguagens de marcação.

Gabarito: Letra C

49.(CESGRANRIO / Transpetro – 2012) Considere o documento DTD a seguir.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT livros (titulo|autores)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT autores (#PCDATA)>
```

O trecho de documento XML consistente com o DTD acima é:

a)
```
<livros>
   <titulo>Principia Mathematica</titulo>
   <autores>Isaac Newton</autores>
   </livros>
```
b)
```
<livros>
   <autores>Isaac Newton</autores>
   <titulo>Principia Mathematica</titulo>
   </livros>
```
c)
```
<livros>
   <autores>
   <autores>Alfred North Whitehead</autores>
   <autores>Bertrand Russel</autores>
   </autores>
   </livros>
```
d)
```
<livros>
   <titulo>Principia Mathematica</titulo>
   <autores>
   <autores>Alfred North Whitehead </autores>
   <autores>Bertrand Russel</autores>
   </autores>
    </livros>
```
e)
```
<livros>
   <titulo>Principia Mathematica</titulo>
    </livros>
```

Comentários:

O DTD define que o elemento <livros> pode conter ou <titulo> ou <autores>, mas não ambos simultaneamente. Isso é indicado pela utilização do operador | na declaração do elemento <livros>.

Assim, somente um dos elementos <titulo> ou <autores> pode aparecer dentro de um único <livros> de acordo com este DTD;

As opções (a) e (b) apresentam <livros> contendo tanto <titulo> quanto <autores>, o que vai contra a definição do DTD;

A opção (c) tenta definir múltiplos <autores> dentro de um <autores>, o que também contraria a definição do DTD, pois <autores> é definido para conter apenas #PCDATA, não outros elementos;

A opção (d) apresenta uma estrutura semelhante à (c), adicionando um <titulo>, mas continua violando o DTD ao tentar incluir múltiplos <autores> dentro de um <autores>;

A opção (e) está correta de acordo com o DTD, pois inclui apenas um <titulo> dentro de <livros>, seguindo a regra de que <livros> pode conter ou <titulo> ou <autores>.

Gabarito: Letra E

---

50. (CESGRANRIO / Petrobras – 2012) Na linguagem XSL,

a) o XSD é o responsável por transformar documentos XML em XHTML.
b) o XSL-FO é o componente que permite a navegação através de um documento XML.
c) o SVG é o componente responsável por descrever gráficos vetoriais bidimensionais.
d) as regras de transformação residem em um arquivo DTD.
e) as transformações podem ocorrer tanto no servidor como no cliente.

Comentários:

(a) Errada. O responsável por transformar documentos XML em XHTML é o XSLT (Extensible Stylesheet Language Transformation), não o XSD (XML Schema Definition);

(b) Errada. O XSL-FO (Extensible Stylesheet Language Formatting Objects) não é responsável pela navegação em um documento XML. Ele é usado para formatação de dados XML para impressão ou visualização;

(c) Errada. O SVG (Scalable Vector Graphics) é responsável por descrever gráficos vetoriais bidimensionais, não o XSL-FO;

(d) Errada. As regras de transformação residem em arquivos XSL, não em um arquivo DTD (Document Type Definition). O DTD é usado para definir a estrutura e validação de documentos XML;

(e) Correta. As transformações podem ocorrer tanto no servidor quanto no cliente, dependendo do contexto e dos requisitos da aplicação. O XSLT pode ser processado no servidor para transformar os dados antes de serem enviados para o cliente, ou pode ser processado no cliente pelo navegador.

Gabarito: Letra E

---

51. (CESGRANRIO / Petrobras – 2012) Sobre o XML DOM, que define uma forma padrão para acessar e manipular documentos XML, considere as afirmativas a seguir.

I - Utiliza um modelo dirigido por eventos para ler documentos XML.

II - Por ser uma API definida através de uma linguagem de definição de interface (IDL), é independente em relação a plataformas e linguagens de programação.

III - É bastante eficiente em relação ao consumo de memória, mesmo no caso de grandes documentos XML.

É correto APENAS o que se afirma em:

a) I
b) II
c) III
d) I e II
e) I e IIII

Comentários:

(I) Errado. O XML DOM utiliza um modelo hierárquico em árvore para representar os documentos XML, não um modelo dirigido por eventos;

(II) Correto. O XML DOM é definido através de uma linguagem de definição de interface (IDL), o que o torna independente em relação a plataformas e linguagens de programação;

(III) Errado. O XML DOM pode consumir uma quantidade significativa de memória, especialmente ao lidar com grandes documentos XML, o que pode afetar sua eficiência.

Gabarito: Letra B

QUESTÕES COMENTADAS – DIVERSAS BANCAS

52. (VUNESP / TJM-SP – 2021) No XML, os nomes de elementos:

a) não diferenciam maiúsculas de minúsculas.
b) devem ser iniciados com um caractere letra ou sublinhado.
c) podem conter letras, números, hifens, sublinhados, pontos ou espaços.
d) não podem conter caracteres acentuados.
e) não podem fazer uso de nomes existentes no HTML.

Comentários:

(a) Errado, XML é case-sensitive; (b) Correto; (c) Errado, podem conter letras, números, hífens, sublinhados e pontos – espaço, não; (d) Errado, podem – sim – conter caracteres acentuados; (e) Errado, podem sem nenhum problema.

Gabarito: Letra B

---

53. (VUNESP / SEMAE DE PIRACICABA – 2021) A opção que representa a forma correta de inserção de um comentário dentro de um arquivo XML é:

a) // meu comentário
b) <!-- meu comentário -->
c) /* meu comentário */
d) # meu comentário
e) ** meu comentário

Comentários:

Comentários utilizam a sintaxe: <!-- meu comentário -->

Gabarito: Letra B

---

54. (IDIB / CRF-MS – 2021) Em relação ao XML, analise as afirmativas a seguir:

(I) O XML é uma linguagem de marcação como o HTML.
(II) O XML é uma linguagem de programação para ser compilada.
(III) O XML é utilizado para armazenar e transportar dados.

É correto o que se afirma:

a) apenas em I e II..

b) apenas em II e III.
c) apenas em I.
d) apenas em I e III.

Comentários:

(I) Correto; (II) Errado, não se trata de uma linguagem de programação e, sim, uma linguagem de marcação; (III) Correto.

Gabarito: Letra D

---

55. (VUNESP / Prefeitura de Presidente Prudente-SP – 2021) Segundo a especificação do XML, se um documento XML é considerado válido, então, é correto afirmar que:

a) ele também é bem-formado.
b) ele possui uma Definição de Tipo de Documento (DTD), mas a sintaxe não necessariamente está correta.
c) a sintaxe dele está correta, mas ele não possui uma Definição de Tipo de Documento (DTD).
d) todos os elementos que compõem o documento possuem, no máximo, um único elemento filho.
e) todos os elementos possuem a propriedade "id" e estão corretamente identificados.

Comentários:

(a) Correto, um documento válido necessariamente é bem formado; (b) Errado, ele não precisa ter necessariamente um DTD; (c) Errado, ele pode ou não possuir um DTD; (d) Errado, isso não faz qualquer sentido lógico; (e) Errado, não é obrigatório ter uma propriedade "id".

Gabarito: Letra A

---

56. (COMPERVE / TJ-RN – 2020) Alguns caracteres causam problemas quando são colocados dentro de conteúdo ou como valores de atributos no XML. Por isso, certos caracteres são proibidos na linguagem, tais como:

a) = e "
b) < e !
c) ! e =
d) " e <

Comentários:

Os caracteres terminantemente proibidos são **<** e **&**; outros que a documentação recomenda veemente que não se utilizem são **>**, **'** e **"**.

Gabarito: Letra D

---

57. (AOCP / UFPB – 2019) O XML não é uma linguagem de programação, mas, sim, de marcação. Sobre XML, é correto afirmar que:

a) o XML é utilizado para aumentar a velocidade na troca de informação.
b) o XML pode ser definido como uma linguagem de metamarcação.
c) o XML nada mais é do que um arquivo que contém a codificação de exibição de uma página web.
d) o XML é uma linguagem de orientação a objetos.
e) o XML pode ser conceituado como uma linguagem de metaprogramação.

Comentários:

(a) Errado, não há nada que justifique uma alternativa como essa sem um parâmetro de comparação – por exemplo: ele é mais lento que o JSON; (b) Correto, pode ser considerado uma linguagem de metamarcação, dado que ela pode utilizar marcações para definir as próprias marcações; (c) Errado, trata-se de uma linguagem para o armazenamento, compartilhamento, e intercâmbio de dados; (d) Errado, não se trata de uma linguagem orientada a objetos – sequer existe o conceito de objeto em XML; (e) Errado, metaprogramação é a programação de programas que se autoprogramam ou programam outros programas, não há nenhuma relação com XML.

Gabarito: Letra B

---

58. (CVV / UFC – 2019) Sobre as características da linguagem XML (eXtensible Markup Language), é correto afirmar:

a) o usuário da linguagem XML pode definir novas tags para melhor estruturar a informação contida no arquivo.

b) a XML é uma linguagem de programação que necessita de um compilador específico para gerar o arquivo binário a ser executado em algum sistema.

c) a desvantagem da XML é a dependência da plataforma sobre a qual está executando, sendo necessárias adequações para cada tipo de sistema.

d) a característica de extensibilidade da linguagem está relacionada ao fato de ser possível criar funções a partir de um conjunto fixo de tags fornecido pela linguagem.

e) a linguagem XML fornece o recurso de tipagem dos dados, de forma que, por exemplo, é possível definir números inteiros e realizar operações sobre eles no programa XML.

Comentários:

(a) Correto, trata-se de uma linguagem extensível; (b) Errado, trata-se de uma linguagem de marcação e, não, de programação; (c) Errado, ele é independente de plataforma; (d) Errado, ela é considerada extensível porque permite estender a quantidade de tags, dado que cada usuário pode criar suas tags sem limitações; (e) Errado, ela não é uma linguagem de programação, logo não há que se falar em operações.

Gabarito: Letra A

---

59. (CCV / UFC – 2018) Qual dos seguintes fragmentos representa um fragmento XML bem formado?

a) <myElement myAttribute="someValue"/>
b) <myElement myAttribute="someValue'/>
c) <myElement myAttribute='someValue'>
d) <myElement myAttribute=someValue/>
e) <myElement myAttribute=someValue>

Comentários:

(a) Correto; (b) Errado, se abriu com aspas duplas deve fechar com aspas duplas; (c) Errado, a tag de fechamento está incorreta – deveria ser />; (d) Errado, utilizam-se aspas simples ou duplas; (e) Errado, utilizam-se aspas simples ou duplas – e a tag de fechamento também está incorreta.

Gabarito: Letra A

---

60. (QUADRIX / CRQ4-SP – 2018) Uma vantagem do DTD é que ele é escrito em linguagem XML, enquanto o XML-Schema possui outra sintaxe de programação.

Comentários:

DTD não é escrito em XML! Já o XML Schema é escrito em XML, mas utiliza uma sintaxe de marcação e, não, de programação.

Gabarito: Errado

---

61. (QUADRIX / CRQ4-SP – 2018) Para a descrição de um XML, tanto o XML-Schema quanto a DTD (Document Type Definition) podem definir os elementos e atributos que podem aparecer em um documento, os tipos de dados para elementos e atributos e os valores-padrão e fixos para elementos e atributos.

Comentários:

Perfeito! O DTD é mais antigo e o XML Schema é mais novo e poderoso, mas ambos podem ser utilizados para descrever um XML.

Gabarito: Correto

---

62. (CONSUPLAN / TRE-RJ – 2017) A respeito de XML, é INCORRETO afirmar que:

a) Processar um documento XML requer um software parser XML (ou processador XML).
b) Todo documento XML deve ter exatamente um elemento-raiz que contém todos os outros elementos.
c) Apesar de documentos XML serem altamente portáveis, visualizar ou modificar documentos XML requer softwares especializados.
d) Um documento XML pode referenciar uma Definição de Tipo de Documento (DTD) ou um esquema que define a estrutura adequada do documento XML.

Comentários:

Todos os itens estão corretos, exceto o terceiro! Visualizar ou modificar elementos XML não requerem softwares especializados – lembrem-se que se trata apenas de um documento de texto.

Gabarito: Letra C

---

63. (UFMT / UFSBA – 2017) Sobre XML (eXtended Markup Language), assinale a afirmativa correta.

a) Possui tecnologias para auxílio na execução de seu código, como DTD (Document Type Definition) e XML Schema.
b) É uma tecnologia recomendada pela W3C, projetada para armazenar e transportar dados.
c) É uma evolução do HTML, por isso páginas HTML migraram para páginas XHTML.
d) É estruturada na forma de árvore, mas permite a existência de mais de um nó raiz no documento.

Comentários:

(a) Errado, DTD e XML Schema não auxiliam na execução de código – eles ajudam a validar documentos; (b) Correto; (c) Errado, não se trata de uma evolução do HTML – eles possuem funções completamente diferentes; (d) Errado, deve apenas um e apenas um nó raiz.

Gabarito: Letra B

---

64. (VUNESP / Prefeitura de Presidente Prudente-SP – 2016) A Definição de Tipo de Documento (DTD) é utilizada no XML para:

a) especificar as transformações a serem aplicadas para reestruturar o documento em um novo formato.
b) validar a estrutura do documento.
c) armazenar valores com mais de 65535 bytes.
d) interligar múltiplos documentos.
e) associar código JavaScript ao documento.

Comentários:

DTD é utilizado para validar a estrutura de um documento – nenhum dos outros itens faz sentido!

Gabarito: Letra B

---

65. (CCV / UFC – 2016) Em um documento XML, o que a DTD representa?

a) Direct Type Definition.
b) Direct Type Document.
c) Dynamic Type Definition.
d) Document Type Definition.
e) Dynamic Type Document.

Comentários:

DTD é a sigla para Document Type Definition.

Gabarito: Letra D

---

66. (FUNRIO / IF-PA – 2016) Em relação as regras de sintaxe do XML são apresentadas as seguintes proposições:

I – Todo documento XML deve conter um elemento raiz que é o pai de todos os outros elementos.
II – Os elementos do XML não precisam estar devidamente aninhados.
III – Os valores dos atributos devem sempre estar entre aspas.

É correto apenas o que se afirma em

a) I.
b) II.
c) III.
d) I e II.
e) I e III.

Comentários:

(I) Correto; (II) Errado, eles precisam necessariamente estar aninhados de forma correta; (III) Correto, valores de atributos entre aspas sempre.

Gabarito: Letra E

LISTA DE QUESTÕES – CESPE

1. **(FGV / TCE-TO – 2022)** Um documento XML é considerado bem formado quando segue as regras de sintaxe estabelecidas na especificação da linguagem. A alternativa que apresenta um documento XML bem formado é:

a) `<Letra> eh uma tag igual a </letra>`
b) `<p>Este eh um paragrafo comum.</p>`
c) `<!--Este eh um -- comentario padrao -->`
d) `<mensagem>salario < 1000</mensagem>`
e) `<b><i>Este texto eh negrito e italico</b></i>`

2. **(CESPE / DPDF - 2022)** Nos códigos em XML a seguir, sexo é um atributo no código A e um elemento no código B, mas ambos os códigos fornecem as mesmas informações.

código A
```
<pessoa sexo="fem">
        <nome>Maria</nome>
        <sobrenome>Silva</sobrenome>
</pessoa>
```

código B
```
<pessoa>
        <sexo>fem</sexo>
        <nome>Maria</nome>
        <sobrenome>Silva</sobrenome>
</pessoa>
```

3. **(CESPE / Polícia Federal - 2018)** Em arquivos no formato XML, as tags não são consideradas metadados.

4. **(CESPE / SEDF – 2017)** Um dos objetivos do projeto XML é que o número de recursos opcionais da linguagem deve ser maximizado para torná-la versátil e adaptável.

5. **(CESPE / TCE-PA – 2016)** Em um documento XML, deve haver diferenciação entre letras maiúsculas e minúsculas e os comentários devem ter a seguinte sintaxe:

<!--comentario-->.

6. **(CESPE / TCE-PA – 2016)** As desvantagens dos esquemas XML incluem a falta de suporte a diferentes tipos de dados.

7. **(CESPE / TCE-PA – 2016)** Um arquivo XML deve conter, no máximo, 1.024 tags. Se o uso de uma quantidade maior de tags for necessário, deve-se adotar o seguinte recurso, a fim de aumentar a quantidade de tags referenciadas pelo arquivo XML principal: um arquivo XML fazer referência a outro.

LISTA DE QUESTÕES – FGV

8. **(FGV / FUNSAÚDE-CE – 2021)** Maria está editando um arquivo XML por meio do bloco de notas do Windows, e deve tomar cuidado com certos caracteres que têm funções especiais. Assinale a lista que contém apenas caracteres especiais do XML.

a) < > & #
b) < > & "
c) > & " /
d) @ " / \
e) / \ @ "

9. **(FGV / TCE-AM – 2021)** O código XML sintaticamente correto é:

a)
```
    <produtos>
  <livro título="Estruturas">
    <ano><2021></ano>
  </livro>
</produtos>
```

b)
```
<produtos>
  <livro título=Estruturas>
    <ano>&<2021&></ano>
  </livro>
</produtos>
```

c)
```
<produtos>
  <livro título="Estruturas">
    <ano>2021<ano>
  <livro>
</produtos>
```

d)
```
<produtos>
  <livro título=Estruturas>
    <ano>"<2021>”</ano>
  </livro>
</produtos>
```

e)
```
<produtos>
  <livro título="Estruturas">
    <ano>&lt;2021&gt;</ano>
  </livro>
</produtos>
```

10. **(FGV / IMBEL – 2021)** O uso do XML é bastante difundido no Brasil para troca de dados em aplicações como notas fiscais, procedimentos médicos, e várias outras. Assinale a utilidade de um Schema XML em aplicações dessa natureza.

a) Direciona os aplicativos que leem arquivos no formato XML.
b) Permite a conversão automática de arquivos XML para o formato CSV.
c) Estabelece precisamente a versão do XML em uso num arquivo no formato XML.
d) Permite que um Web Service interprete corretamente um arquivo de dados no formato XML.
e) Permite a verificação da estrutura e regras de preenchimento de um arquivo contendo dados no formato XML.

11. **(FGV / MPE-RJ – 2019)** A troca de dados entre sistemas computacionais é normalmente realizada por meio de arquivos que seguem padrões de formato e organização. Desse modo, diferentes agentes com diferentes equipamentos podem enviar e receber dados estruturados muito facilmente. Nesse contexto, analise um trecho do conteúdo de um dado arquivo a seguir.

```
<nota>
          <para>Rita</para>
          <de>Bernardo</de>
          <título>Lembrete</título>
          <texto>O pacote &lt;chegou&gt; ...</texto>
</nota>
```

Com base nesse trecho, é correto deduzir que a organização desse arquivo segue o padrão conhecido como:

a) CSS.
b) CSV.
c) ODF.
d) PDF.
e) XML.

12. **(FGV / DPE-RJ – 2019)** Considere os trechos XML exibidos a seguir.

I.
```
<p>Um primeiro exemplo</p>
<br/>
```

II.
```
<message>Texto breve</message>
```

III.
```
<b><i>Texto com destaque.</b></i>
```

IV.
```
<p>Note que, para x>1, a resposta é sim.</p>
```

O número de trechos válidos é:

a) 0.
b) 1.
c) 2.
d) 3.
e) 4.

13. **(FGV / MPE-AL – 2018)** No XML, a sequência de símbolos

```
&lt;
```

Representa:

a) <
b) >
c) “
d) &
e) ’

14. **(FGV / Câmara de Salvador-BA – 2018)** Analise o conteúdo XML de um arquivo de seis linhas, exibido a seguir.

```
<?xml version="1.0" encoding="UTF-8"?>
<database catalogo="BD" user="U1">
<SQL>
  select * FROM T where a < 10
</SQL>
</database>
```

A validação desse arquivo apontaria um erro na linha de número:

a) 1.
b) 2.
c) 3.
d) 4.
e) 6.

15. **(FGV / Prefeitura de Niterói-RJ – 2018)** Considere a declaração do tipo de documento (DTD) a seguir.

```
<!ELEMENT blog (nome, autor+, artigo*, permalink?) >
<!ELEMENT nome (#PCDATA)>
<!ELEMENT autor (#PCDATA)>
<!ELEMENT artigo (#PCDATA)>
<!ELEMENT permalink (#PCDATA)>
```

Em um documento XML, que obedece a esse conjunto de regras,

a) deve haver precisamente um elemento do tipo artigo.
b) podem ocorrer vários elementos do tipo permalink.
c) pode conter um elemento do tipo nome e outro elemento do tipo autor, em qualquer ordem.
d) deve existir um elemento do tipo autor, sendo permitidas múltiplas ocorrências deste tipo de elemento.
e) não é necessário haver um elemento do tipo nome.

16. **(FGV / Prefeitura de Niterói-RJ – 2018)** XML é uma linguagem de marcação projetada para descrever e transportar dados. Dado que em um documento XML é permitido ao desenvolvedor de software definir seus próprios elementos, pode ser necessário utilizar namespaces para evitar conflitos de nomes. Em relação à namespaces em XML, analise as afirmativas a seguir.

I. Um namespace pode ser declarado no elemento em que é utilizado ou no elemento raiz do documento XML.
II. O atributo uri é reservado em XML para indicar que um prefixo está associado ao namespace.
III. As várias declarações de namespace com prefixos podem ser feitas em um elemento, mas devem possuir prefixos diferentes.

Assinale:

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

17. **(FGV / IBGE – 2017)** As declarações de elementos na DTD determinam a possível estrutura de um documento XML. Analise a DTD a seguir:

```
<!DOCTYPE memo
[
<!ELEMENT memo (from, to+, date?, content)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT content (p*)>
<!ELEMENT p (#PCDATA)>
]>
```

É correto afirmar que o(s) elemento(s):

a) **memo** pode conter os elementos **from, to, date** e **content** em qualquer ordem;
b) **content** deve conter um ou mais elementos **p**;

c) **date** é opcional;
d) **to** é obrigatório e precisa ocorrer mais de uma vez dentro do elemento **memo**;
e) **from**, **to** e **date** podem conter qualquer um dos elementos descritos na DTD.

18. (FGV / ALERJ – 2017) XML (Extensible Markup Language) é um sistema de codificação que permite que diferentes tipos de informação sejam distribuídos através da World Wide Web. Com a XML, diversos sistemas de informação, semelhantes ou não, se comunicam de forma transparente entre si. Em relação à linguagem XML, analise as afirmativas a seguir:

I. Seções CDATA podem ocorrer em qualquer parte de um documento XML e devem ser utilizadas para inserir blocos de texto que contenham caracteres especiais como & e <.
II. Documentos XML bem formados devem ter um DTD (Document Type Definition) associado e obedecer a todas as regras que o DTD contém.
III. Na linguagem XML é permitido omitir as tags finais em elementos não vazios.

Está correto o que se afirma em:

a) somente I;
b) somente II;
c) somente III;
d) somente I e II;
e) I, II e III.

19. (FGV / Prefeitura de Paulínia-SP – 2016) Analise o trecho de um documento XML exibido a seguir.

```
<message>salary &lt; 1000</message>
```

Assinale a opção que indica o significado do símbolo "&lt".

a) É uma diretiva de formatação de texto.
b) Representa o caracter "<".
c) É um operador de multiplicação de constantes.
d) É uma constante matemática da biblioteca "&math".
e) Representa um bookmark.

20. (FGV / CODEBA – 2016) Analise o seguinte trecho de XML Schema (XSD).

```
<xs:complexType name="TipoEstudante">
 <xs:sequence>
  <xs:element name="nome" type="xs:string"/>
   <xs:element name="sobrenome" type="xs:string"/>
   <xs:element name="notas" type="xs:positiveInteger"/>
 </xs:sequence>
  <xs:attribute name="matricula" type="xs:positiveInteger"/>
</xs:complexType>
```

Assinale o elemento XML cuja definição está de acordo a especificação de "TipoEstudante"

a)
```
<estudante "493">
 <nome>Maria</nome>
 <sobrenome>Ferreira</sobrenome>
 <notas>95</notas>
</estudante>
```

b)
```
<estudante matricula="493">
 <nome>Maria</nome>
 <sobrenome>Ferreira</sobrenome>
 <notas>9.5, 8.8, 10.0</notas>
</estudante>
```

c)
```
<estudante matricula="493">
 <nome>Maria</nome>
 <sobrenome>Ferreira</sobrenome>
</estudante>
```

d)
```
<estudante matricula="493">
 <nome>Maria</nome>
 <sobrenome>Ferreira</sobrenome>
 <notas>95</notas>
</estudante>
```

e)
```
<estudante matricula="493">
 <notas>95, 27, 48</notas>
</estudante>
```

21. **(FGV / DPE-RO – 2015)** Os trechos contendo XML mal-formado, válido e inválido, respectivamente, são:

```
<!DOCTYPE processo [
<!ELEMENT sujeitos (juiz, autor+, reu+)>
<!ELEMENT juiz (#PCDATA)>
<!ELEMENT autor (#PCDATA)>
<!ELEMENT reu (#PCDATA)>
]>
```

e os seguintes trechos de documento XML:

I)
```
<processo>
 <sujeitos>
   <juiz>Dr. Pedro da Silva</juiz>
  <autor>Fulano de Souza</autor>
  <reu>Cicrano Pereira</reu>
 </sujeitos>
</processo>
```

II)
```
<processo>
 <sujeitos>
  <autor>Fulano de Souza</autor>
  <reu>Cicrano Pereira</reu>
 </sujeitos>
</processo>
```

III)
```
<processo>
 <sujeitos>
  <juiz>Dr. Pedro da Silva
  <autor>Fulano de Souza
  <reu>Cicrano Pereira
 </sujeitos>
</processo>
```

a) I, II e III;
b) I, III e II;
c) II, I e III;
d) III, I e II;
e) III, II e I.

22. **(FGV / TJ-PI – 2015)** Num trecho XML, o comentário "Trecho em teste" deve ser introduzido como:

a) < !-- Trecho -- em -- teste -- >
b) < !-- Trecho em teste >
c) < !Trecho em teste >
d) < !--Trecho em teste -->
e) <-- Trecho em teste -->

23. (FGV / DPE-RO – 2015) A representação em XML da agenda de uma pessoa em que o telefone do usuário João está representado como um atributo é:

a)
```
<usuario>
 <nome>Joao</nome>
 <telefone>5555-5555</telefone>
</usuario>
```

b)
```
<telefone nome="João">4444-44444</telefone>
```

c)
```
<nome>joão
 <telefone>3333-3333</telefone>
</nome>
```

d)
```
<usuario>
 <nome>João</nome>
 <atrib>7777-888</atrib>
</usuario>
```

e)
```
<usuario telefone="9999-9999">
 <nome>João</nome>
</usuario>
```

24. (FGV / PGE-RO – 2015) Analise, abaixo, a lista de definições que podem ser estabelecidas por meio de um esquema (schema) para um documento XML:

I. os elementos que podem ser utilizados;
II. os tipos de dados para elementos e atributos;
III. valores default para elementos e atributos;
IV. espaços reservados para comentários.

Somente estão corretas as afirmativas:

a) I e II;
b) I e III;
c) II e III;
d) I, II e III;
e) II, III e IV.

25. (FGV / PGE-RO – 2015) São requisitos de um documento XML, EXCETO:

a) deve ter um elemento raiz, responsável por aninhar os demais elementos que formam o documento.
b) deve ser bem formado, ou seja, possuir somente uma tag raiz, possuir tags de fechamento, etc.;

c) pode conter atributos que devem ser únicos dentro do elemento, e seus valores devem estar envoltos em aspas.
d) deve ser válido quando se desejar formalização na representação das informações.
e) deve começar por uma declaração de conteúdo do elemento.

26. (FGV / TJ-GO – 2014) Observe os principais tópicos de um arquivo XML de uma nota fiscal eletrônica.

```
<?xml version="1.0" encoding="UTF-8"?>
- <nfeProc versao="2.00" xmlns="http://www.portalfiscal.in
  + <NFe xmlns="http://www.portalfiscal.inf.br/nfe">
  + <protNFe versao="2.00">
</nfeProc>
```

O atributo *xmlns* define o que é conhecido como:

a) Namespace;
b) Name Source;
c) Named Schema;
d) Name Status;
e) Name Server.

27. (FGV / AL-MA – 2013) Dentre as alternativas a seguir, selecione aquelas que correspondem a especificações elaboradas com a finalidade de definir regras de validação (esquemas) para documentos XML:

I. XSLT
II. DTD
III. XML Schema

Assinale:

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

28. (FGV / AL-MA – 2013) O scritp XML a seguir, que faz referência ao esquema verifica.xsd, está sintaticamente incorreto porque UTF-8 não é suportado no XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<addresses xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation='verifica.xsd'>
  <endereco>
  <nome>JoaoTestador</nome>
  <rua>CPD Informatica 00</rua>
</endereco>
```

LISTA DE QUESTÕES – CESGRANRIO

29. (CESGRANRIO / IPEA - 2024) O Ipea pretende publicar a Tabela de índices de custos e preços abaixo, relativa aos meses do primeiro trimestre de 2023. Para publicá-la, é necessário colocá-la no formato XML. Desconsiderando - se a parte de DOCTYPE e Style, ao colocar essa Tabela no formato XML, obtém-se:

a)
```
<Indices>
  <Indice> ICTI IPCA IGPM </Indice>
  <Indice> 182,34 0,53 0,21 </Indice >
  <Indice> 183,16 0,84 - 0,06 </Indice >
<Indice> 183,34 0,76 0,05 </Indice >
```

b)
```
<Indices>
 <Indice>
   <ICTI> 182,34 </ICTI>
   <IPCA> 0,53 </IPCA>
   <IGPM> 0,21 </IGPM>
 </Indice>
 <Indice>
  <ICTI> 183,16 </ICTI>
  <IPCA> 0,84 </IPCA>
  <IGPM> - 0,06 </IGPM>
 </Indice>
 <Indice>
  <ICTI> 183,34 </ICTI>
  <IPCA> 0,76 </IPCA>
  <IGPM> 0,05 </IGPM>
 </Indice>
</Indices>
```

c)
```
<Indices>
 <1>
 <ICTI> 182,34 </ICTI>
 <IPCA> 0,53 </IPCA>
 <IGPM> 0,21 </IGPM>
 <2>
 <ICTI> 183,16 </ICTI>
 <IPCA> 0,84 </IPCA>
```

```
 <IGPM> - 0,06 </IGPM>
 <3>
 <ICTI> 183,34 </ICTI>
 <IPCA> 0,76 </IPCA>
 <IGPM> 0,05 </IGPM>
</Indices>
```

d)

```
<Indices>
  <ICTI> 182,34 </ICTI>
  <IPCA> 0,53 </IPCA>
  <IGPM> 0,21 </IGPM>
</Indices>
<Indices>
  <ICTI> 183,16 </ICTI>
  <IPCA> 0,84 </IPCA>
  <IGPM> - 0,06 </IGPM>
</Indices>
<Indices>
   <ICTI> 183,34 </ICTI>
   <IPCA> 0,76 </IPCA>
   <IGPM> 0,05 </IGPM>
</Indices>
```

e)

```
<Indices>
 <insert>
 <1>
   <ICTI> 182,34 </ICTI>
   <IPCA> 0,53 </IPCA>
   <IGPM> 0,21 </IGPM>
 </1>
 <insert>
 <2>
  <ICTI> 183,16 </ICTI>
  <IPCA> 0,84 </IPCA>
  <IGPM> - 0,06 </IGPM>
 </2>
 <insert>
 <3>
  <ICTI> 183,34 </ICTI>
  <IPCA> 0,76 </IPCA>
  <IGPM> 0,05 </IGPM>
 </3>
```

</Indices>

30. **(CESGRANRIO / TRANSPETRO – 2023)** Um profissional de Informática está trabalhando em um projeto que envolve a manipulação de documentos XML. Ele precisa garantir que os documentos XML estejam bem-formados e válidos, de acordo com as especificações do XML 1.1. Uma das regras que ele deverá seguir para garantir que um documento XML 1.1 seja válido é que o(s):

a) documento pode ter um ou mais elementos raiz.
b) documento deve começar com uma declaração XML.
c) nomes dos elementos são insensíveis a maiúsculas e minúsculas.
d) atributos devem ter o mesmo nome se estiverem no mesmo elemento.
e) comentários XML devem aparecer como atributos de uma etiqueta (tag).

31. **(CESGRANRIO / BASA - 2022)** Um projetista de sistemas está desenvolvendo um sistema e precisou programar um arquivo XSLT. Neste arquivo, ele precisou inserir um elemento para aplicar uma regra de modelo, a partir de uma folha de estilo importada, ao invés de uma regra equivalente, a partir da folha de estilo principal, mas sem que este elemento apareça como o primeiro nó filho de .

Para este caso, o elemento que deve ser inserido para aplicar tal regra nesse arquivo XSLT é o

a) apply-imports
b) apply_templates
c) imports
d) include
e) Template

32. **(CESGRANRIO / BASA – 2022)** Ao desenvolver um sistema de notícias, a empresa X decidiu manter as notícias em um formato XML, como o do exemplo a seguir:

```
<?xml version="1.0"?>
<news>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</news>
```

Mais tarde, entendeu que, para esse formato exemplificado acima, seria melhor definir um esquema em XSD. Que fragmento de código XSD deve conter esse esquema para permitir que o exemplo apresentado seja validado corretamente, quando nele for incluída a referência ao esquema completo?

a) <xs:element name="news">

```
<xs:complexType>
 <xs:sequence> <xs:element name="heading" type="xs:string"/>
  <xs:element name="body" type="xs:string"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

b)
```
<xs:element name="news">
 <xs:sequence>
  <xs:element name="heading" type="xs:string"/>
  <xs:element name="body" type="xs:string"/>
 </xs:sequence>
</xs:element>
```

c)
```
<xs:element name="news">
  <xs:element name="heading" type="xs:string"/>
 <xs:element name="body" type="xs:string"/>
</xs:element>
```

d)
```
<!ELEMENT news (heading, body)>
 <!ELEMENT heading (#PCDATA)>
 <!ELEMENT body (#PCDATA)>
```

e)
```
<!ELEMENT news (heading, body)>
 <!ELEMENT heading (text)>
 <!ELEMENT body (text)>
```

33. **(CESGRANRIO / BASA – 2021)** Ao participar de uma equipe para desenvolvimento de um website para a intranet do banco em que trabalhava, um programador teve como missão criar uma tabela HTML a partir de um arquivo XML que indicava clientes e seus saldos.

O fragmento de XML a seguir é um exemplo da estrutura do XML do arquivo que conterá os dados:

```
<?xml version="1.0" encoding="UTF - 8"?>
<clientes>
  <cliente>
 <nome>Ana Zurique</nome>
  <saldo>3000</saldo>
 </cliente>
 <cliente>
  <nome>Bernardo Washington</nome>
  <saldo>4500</saldo>
</cliente>
```

```
<cliente>
  <nome>Carlos York</nome>
  <saldo>12345</saldo>
</cliente>
</cliente>
```

Para esse arquivo, a tabela gerada deve ter a seguinte aparência:

Cliente	Saldo
Ana Zurique	3000
Bernardo Washington	4500
Carlos York	12345

Inicialmente, o programador construiu o seguinte arquivo em XSLT:

```
<?xml version="1.0" encoding="UTF - 8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
 <table border="1">
<tr>
 <th>Cliente</th>
 <th>Saldo</th>
 </tr>
 <! - - Código Para os Dados - - >
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Que sequência de instruções deve substituir o comentário , de forma a gerar a tabela no formato apresentado?

a)

```
 <xsl:for - each select="clientes/cliente/">
<tr>
<td><xsl:value - of select="nome"/></td>
<td><xsl:value - of select="saldo"/></td>
</tr>
</xsl:for - each>
```

b)
<xsl:for - each select="clientes/cliente">
<tr>
<td><xsl:value - of select="clientes/cliente/nome"/></td>
<td><xsl:value - of select="clientes/cliente/saldo"/></td>
</tr>
</xsl:for - each>

c)
<xsl:for - each select="clientes/cliente">
<tr>
<td><xsl:value - of select="nome"/></td>
<td><xsl:value - of select="saldo"/></td>
</tr>
</xsl:for - each>

d)
<xsl:for - each select="clientes/">
<tr>
<td><xsl:value - of select="cliente/nome"/></td>
<td><xsl:value - of select="cliente/saldo"/></td>
</tr>
</xsl:for - each>

e)
<xsl:for - each select="clientes">
<tr>
<td><xsl:value - of select="cliente/nome"/></td>
<td><xsl:value - of select="cliente/saldo"/></td>
</tr>
</xsl:for - each>

34. **(CESGRANRIO / CEF – 2021)** Um arquivo, contendo um documento XML, contém exatamente a seguinte informação:

<?xml version=”1.0”?>
<PEDIDOS>
<PEDIDO>
<TITULO>Pedido de Empréstimo</TITULO>
<REQUERENTE>José da Silva</REQUERENTE>
<CPF>999.999.999 - 99</CPF>
<VALOR>20000</VALOR>
<PEDIDO>

<PEDIDOS>

A partir desse documento apenas, um processador XML pode garantir que o arquivo é

a) bem - formado, apenas
b) bem - formado e normalizado
c) bem - formado e válido
d) normalizado, apenas
e) válido, apenas

35. **(CESGRANRIO / Caixa – 2021)** As fontes (feed) RSS devem todas fornecer informações em

a) CSS
b) HTML 1.0
c) HTML 1.1
d) SOAP
e) XML

36. **(CESGRANRIO / TRANSPETRO – 2018)** Considerando a linguagem XML, qual é o exemplo correto de uso de um atributo chamado "src" que recebe o valor "computador.gif" em um elemento de nome "img"?

a)
b) <img src="computador.gif "/img>
c) <img> "src=computador.gif " </img>
d) <img> <src> computador.gif </src> </img>
e) <img> src="computador.gif " </img>

37. **(CESGRANRIO / PETROBRAS – 2018)** Qual linguagem de marcação, fundamental para o estabelecimento de serviços Web, que compõe uma Arquitetura Orientada a Serviços, é usada para que dados sejam apresentados, comunicados e armazenados?

a) HTML;
b) XML;
c) JAVA;
d) JAVASCRIPT;
e) C#.

38. **(CESGRANRIO / BASA – 2018)** Considere o esquema XML a seguir:

```
<xs:elemente name="rectangle" type = "area"/>
<xs:complexType name = "area">
 <xs:attribute name="x1" type="xs:decimal" />
 <xs:attribute name="y1" type="xs:decimal" />
```

```
 <xs:attribute name="x2" type="xs:decimal" />
 <xs:attribute name="y2" type="xs:decimal" />
<xs:complexType>
```

Um elemento XML válido, segundo esse esquema, é:

a) <area><x1>1</x1><y1>1</y1><x2>2</x2><y2>3</y2></area>
b) <area x1="1" y1="1" x2="4" y2="5" />
c) <rectangle x1="5" y1="4" x2="1" y2="1"/>
d) <area><x1>4</x1><y1>4</y1><x2>2</x2><y2>3</y2></area>
e) <rectangle><x1>1</x1><y1>1</y1><x2>2</x2><y2>3</y2></rectangle>

39. **(CESGRANRIO / TRANSPETRO – 2018)** Documentos XML são estruturados segundo uma hierarquia de unidades informacionais chamadas de nós. Qual tecnologia XML fornece ao desenvolvedor uma API para adicionar, editar e remover esses nós?

a) XMI
b) XSDL
c) XSLT
d) XML DOM
e) XML Schema

40. **(CESGRANRIO / BASA – 2014)** Sabendo que um arquivo XML está sintaticamente correto e que pode ser consumido ou processado por um parser XML, de acordo com a especificação XML, pode-se afirmar, com certeza, que ele é:

a) autorizado
b) validado
c) certificado
d) compilado
e) bem formado

41. **(CESGRANRIO / AGC – 2014)** Analise o seguinte DTD

```
<?xml version="1.0" ?>
<!DOCTYPE A [
 <!ELEMENT A (B,C)>
 <!ELEMENT B (#PCDATA)>
 <!ELEMENT C (D?,E)>
 <!ELEMENT D (#PCDATA)>
 <!ELEMENT E (#PCDATA)>
]>
```

Segundo o DTD acima, que documento XML NÃO é válido?

a) <A><B>Teste</B> <C><E>Teste</E> </C></A>
b) <A><B>Teste</B> <C><D>Teste</D> <E>Teste</E></C></A>
c) <A><B>Teste</B> <C>Teste<D>Teste</D> <E>Teste</E></C></A>
d) <A><B> </B> <C> <D>Teste</D><E>Teste</E></C></A>
e) <A><B></B><C><D></D><E></E></C></A>

42. (CESGRANRIO / BASA - 2014) Seja o arquivo XML abaixo:

```
<?xml version="1.0" encoding="UTF - 8"?>
<T><P N="1">
<K N="1" M="G">Texto</K>
<K N="2" M="H">Texto</K>
<K></K></P>
<P><F>Texto</F></P>
</T>
```

Que DTD permite que esse arquivo seja considerado válido?

a)
```
<!ATTLIST ( P+ ) >
<!ATTLIST P ( K?, F? ) >
<!ELEMENT P N NMTOKEN #IMPLIED >
<!ATTLIST K ( #PCDATA ) >
<!ELEMENT K M NMTOKEN #IMPLIED >
<!ELEMENT K N NMTOKEN #IMPLIED >
<!ATTLIST F ( #PCDATA ) >
```

b)
```
<!ELEMENT T ( P+ ) >
<!ELEMENT P ( K*, F? ) >
<!ATTLIST P N NMTOKEN #IMPLIED >
<!ELEMENT K ( #PCDATA ) >
<!ATTLIST K M NMTOKEN #IMPLIED >
<!ATTLIST K N NMTOKEN #IMPLIED >
<!ELEMENT F ( #PCDATA ) >
```

c)
```
<!ELEMENT T ( P+ ) >
<!ELEMENT P ( K*, F? ) >
<!ATTLIST P N NMTOKEN #REQUIRED >
<!ELEMENT K ( #PCDATA ) >
<!ATTLIST K M NMTOKEN #IMPLIED >
<!ATTLIST K N NMTOKEN #IMPLIED >
<!ELEMENT F ( #PCDATA ) >
```

```
<!ELEMENT T ( P+ ) >
<!ELEMENT P ( K?, F? ) >
<!ATTLIST P N NMTOKEN #IMPLIED >
<!ELEMENT K ( #PCDATA ) >
<!ATTLIST K M NMTOKEN #IMPLIED >
<!ATTLIST K N NMTOKEN #IMPLIED >
<!ELEMENT F ( #PCDATA ) >
```

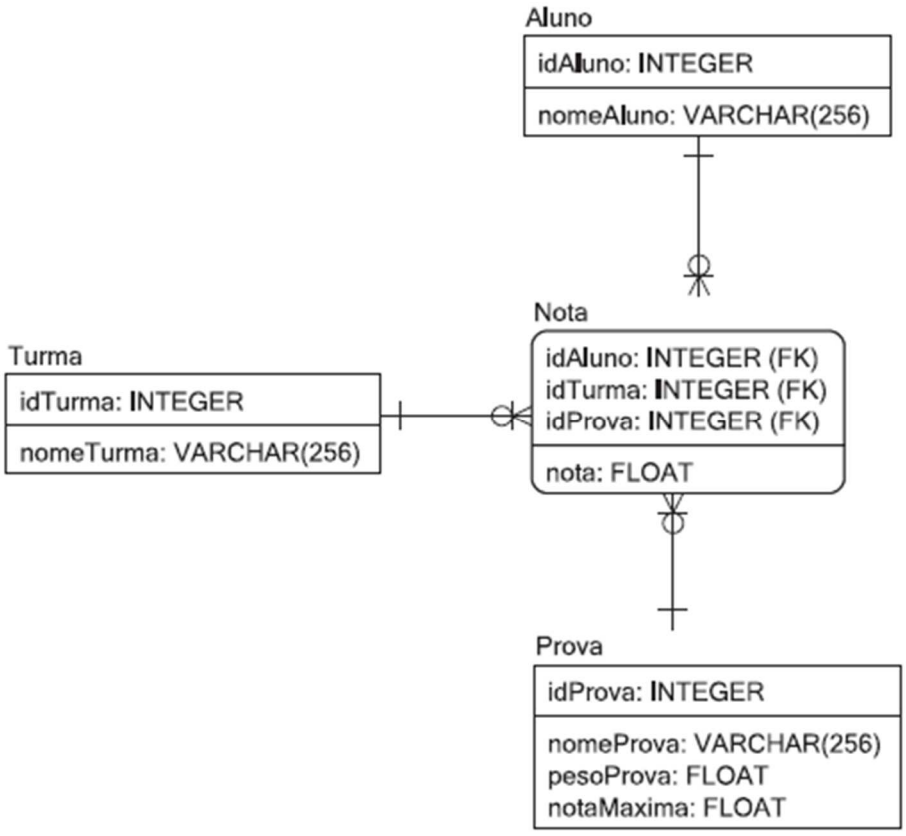
d)

```
<!ATTLIST ( P+ ) >
<!ATTLIST P ( K*, F? ) >
<!ELEMENT P N NMTOKEN #IMPLIED >
<!ATTLIST K ( #PCDATA ) >
<!ELEMENT K M NMTOKEN #IMPLIED >
<!ELEMENT K N NMTOKEN #IMPLIED >
<!ATTLIST F ( #PCDATA ) >
```

e)

43. (CESGRANRIO / CEFET RJ - 2014) Uma universidade decidiu alterar seu sistema acadêmico, atualmente escrito em Delphi, para aceitar uma interface Web. Para isso, decidiu adotar as tecnologias Ajax e PHP.

A primeira parte do trabalho será alterar o subsistema de avaliação, chamado de NOTAS. O modelo de dados atual desse subsistema é bastante simples, e é descrito pelo modelo diagrama a seguir, que usa a notação da Engenharia da Informação.



Que fragmento de código XML o sistema NOTAS pode usar para representar corretamente uma linha da tabela Turma?

a) </TURMA></IDTURMA>1<IDTURMA></NOMETURMA>Cálculo<NOMETURMA><TURMA>
b) <TURMA><IDTURMA><NOMETURMA>1</IDTURMA>Cálculo</NOMETURMA></TURMA>
c) <TURMA><IDTURMA>1</IDTURMA><NOMETURMA>Cálculo</NOMETURMA></TURMA>
d) <TURMA><IDTURMA>1<IDTURMA><NOMETURMA>Cálculo<NOMETURMA><TURMA>
e)
<TURMA/><IDTURMA/>Cálculo</IDTURMA><NOMETURMA/>1</NOMETURMA></TURMA>

44. (CESGRANRIO / IBGE - 2013) O gerente acadêmico de uma universidade solicitou ao setor de tecnologia da informação que fosse desenvolvida uma ferramenta que permitisse a distribuição dos currículos dos professores em diferentes formatos, uma vez que isso é essencial para promover o intercâmbio de informações entre diferentes instituições de ensino do Brasil e do exterior. Sabendo-se que os currículos que estão armazenados na base de dados da universidade são documentos XML válidos, qual tecnologia XML deve ser empregada na construção dessa ferramenta?

a) XSD
b) PDF
c) XSL
d) XKMS
e) HTML

45. (CESGRANRIO / LIQUIGÁS - 2013) Muitas tecnologias usadas pela indústria de software favorecem a implantação de melhorias na gestão de processos integrados de negócios. Um exemplo disso é o uso de:

a) softwares abertos.
b) bancos de dados relacionais.
c) computação móvel em larga escala.
d) orientação a objetos como paradigma de desenvolvimento.
e) XML para a troca de informações entre sistemas.

46. (CESGRANRIO / AGC – 2012) Um documento XML bem formado (well - formed) segue as restrições de sintaxe definidas pela especificação XML.

PORQUE

Um documento XML bem formado deve, necessariamente, estar em conformidade com uma definição em DTD (Document Type Definition) ou em XML Schema.

Analisando - se as afirmações acima, conclui-se que:

a) as duas afirmações são verdadeiras, e a segunda justifica a primeira.
b) as duas afirmações são verdadeiras, e a segunda não justifica a primeira.
c) a primeira afirmação é verdadeira, e a segunda é falsa.
d) a primeira afirmação é falsa, e a segunda é verdadeira.
e) as duas afirmações são falsas.

47. **(CESGRANRIO / Petrobras – 2012)** Solicitado a preparar um arquivo de teste em XML para um sistema de controle de pedidos de uma distribuidora de petróleo, um analista de sistemas gerou o seguinte documento:

```
< ? xml version="1.0" encoding="UTF-8"? >
< ! DOCTYPE cliente SYSTEM "C:\postos.dtd" >
< cliente >
    < posto >
        < cnpj >
        53.726.891/0001-24
< /cnpj >
        < pedidos >
            < pedido >
                < produto >
            Gasolina
< /produto >
                < quantidade >
            10.000
< /quantidade >
            < /pedido >
            < pedido >
                < produto >
            Gasolina
< /produto >
            < /pedido >
        < /pedidos >
    < /posto >
< /cliente >
```

Considere o DTD abaixo, salvo no arquivo C:\postos.dtd.

```
< ? xml version="1.0" encoding="UTF-8"? >
< ! ELEMENT quantidade (#PCDATA) >
< ! ELEMENT produto (#PCDATA) >
< ! ELEMENT posto (cnpj,pedidos*) >
< ! ELEMENT pedidos (pedido*) >
< ! ELEMENT pedido (produto, quantidade)m>
< ! ELEMENT cnpj (#PCDATA) >
```

< ! ELEMENT cliente (posto) >

O arquivo preparado pelo analista está em

a) formato diferente do XML.
b) XML, mas não é válido e não é bem-formado.
c) XML, é bem-formado, mas não é válido.
d) XML, é válido, mas não é bem-formado.
e) XML, é válido e bem-formado.

48.(CESGRANRIO / Liquigás – 2012) Com a proliferação de aplicações e serviços utilizados na Internet, o conjunto geral de marcadores presente na linguagem HTML começou a se tornar restritivo, e a necessidade de extensões para criar novos tipos de marcadores começou a surgir. Uma das soluções adotadas pelo W3C foi padronizar uma nova linguagem com a capacidade de ser extensível, sobre a qual rótulos pudessem ser criados de acordo com a necessidade das aplicações. De fato, tal linguagem é muito mais uma metalinguagem, no sentido de que, a partir dela, outras linguagens (até mesmo a própria HTML) com suas marcações poderiam ser geradas. Essa metalinguagem é conhecida como:

a) UML
b) WML
c) XML
d) VML
e) SVG

49.(CESGRANRIO / Transpetro – 2012) Considere o documento DTD a seguir.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT livros (titulo|autores)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT autores (#PCDATA)>
```

O trecho de documento XML consistente com o DTD acima é:

a)
```
<livros>
   <titulo>Principia Mathematica</titulo>
   <autores>Isaac Newton</autores>
   </livros>
```
b)
```
<livros>
   <autores>Isaac Newton</autores>
   <titulo>Principia Mathematica</titulo>
   </livros>
```

c)
<livros>
<autores>
<autores>Alfred North Whitehead</autores>
<autores>Bertrand Russel</autores>
</autores>
</livros>
d)
<livros>
<titulo>Principia Mathematica</titulo>
<autores>
<autores>Alfred North Whitehead </autores>
<autores>Bertrand Russel</autores>
</autores>
</livros>
e)
<livros>
<titulo>Principia Mathematica</titulo>
</livros>

50. (CESGRANRIO / Petrobras – 2012) Na linguagem XSL,

a) o XSD é o responsável por transformar documentos XML em XHTML.
b) o XSL-FO é o componente que permite a navegação através de um documento XML.
c) o SVG é o componente responsável por descrever gráficos vetoriais bidimensionais.
d) as regras de transformação residem em um arquivo DTD.
e) as transformações podem ocorrer tanto no servidor como no cliente.

51. (CESGRANRIO / Petrobras – 2012) Sobre o XML DOM, que define uma forma padrão para acessar e manipular documentos XML, considere as afirmativas a seguir.

I - Utiliza um modelo dirigido por eventos para ler documentos XML.

II - Por ser uma API definida através de uma linguagem de definição de interface (IDL), é independente em relação a plataformas e linguagens de programação.

III - É bastante eficiente em relação ao consumo de memória, mesmo no caso de grandes documentos XML.

É correto APENAS o que se afirma em:

a) I
b) II
c) III

d) I e II
e) I e IIII

LISTA DE QUESTÕES – DIVERSAS BANCAS

52. (VUNESP / TJM-SP – 2021) No XML, os nomes de elementos:

a) não diferenciam maiúsculas de minúsculas.
b) devem ser iniciados com um caractere letra ou sublinhado.
c) podem conter letras, números, hifens, sublinhados, pontos ou espaços.
d) não podem conter caracteres acentuados.
e) não podem fazer uso de nomes existentes no HTML.

53. (VUNESP / SEMAE DE PIRACICABA – 2021) A opção que representa a forma correta de inserção de um comentário dentro de um arquivo XML é:

a) // meu comentário
b) <!-- meu comentário -->
c) /* meu comentário */
d) # meu comentário
e) ** meu comentário

54. (IDIB / CRF-MS – 2021) Em relação ao XML, analise as afirmativas a seguir:

(I) O XML é uma linguagem de marcação como o HTML.
(II) O XML é uma linguagem de programação para ser compilada.
(III) O XML é utilizado para armazenar e transportar dados.

É correto o que se afirma:

a) apenas em I e II..
b) apenas em II e III.
c) apenas em I.
d) apenas em I e III.

55. (VUNESP / Prefeitura de Presidente Prudente-SP – 2021) Segundo a especificação do XML, se um documento XML é considerado válido, então, é correto afirmar que:

a) ele também é bem-formado.
b) ele possui uma Definição de Tipo de Documento (DTD), mas a sintaxe não necessariamente está correta.
c) a sintaxe dele está correta, mas ele não possui uma Definição de Tipo de Documento (DTD).
d) todos os elementos que compõem o documento possuem, no máximo, um único elemento filho.
e) todos os elementos possuem a propriedade "id" e estão corretamente identificados.

56. **(COMPERVE / TJ-RN – 2020)** Alguns caracteres causam problemas quando são colocados dentro de conteúdo ou como valores de atributos no XML. Por isso, certos caracteres são proibidos na linguagem, tais como:

a) = e “
b) < e !
c) ! e =
d) " e <

57. **(AOCP / UFPB – 2019)** O XML não é uma linguagem de programação, mas, sim, de marcação. Sobre XML, é correto afirmar que:

a) o XML é utilizado para aumentar a velocidade na troca de informação.
b) o XML pode ser definido como uma linguagem de metamarcação.
c) o XML nada mais é do que um arquivo que contém a codificação de exibição de uma página web.
d) o XML é uma linguagem de orientação a objetos.
e) o XML pode ser conceituado como uma linguagem de metaprogramação.

58. **(CVV / UFC – 2019)** Sobre as características da linguagem XML (eXtensible Markup Language), é correto afirmar:

a) o usuário da linguagem XML pode definir novas tags para melhor estruturar a informação contida no arquivo.

b) a XML é uma linguagem de programação que necessita de um compilador específico para gerar o arquivo binário a ser executado em algum sistema.

c) a desvantagem da XML é a dependência da plataforma sobre a qual está executando, sendo necessárias adequações para cada tipo de sistema.

d) a característica de extensibilidade da linguagem está relacionada ao fato de ser possível criar funções a partir de um conjunto fixo de tags fornecido pela linguagem.

e) a linguagem XML fornece o recurso de tipagem dos dados, de forma que, por exemplo, é possível definir números inteiros e realizar operações sobre eles no programa XML.

59. **(CCV / UFC – 2018)** Qual dos seguintes fragmentos representa um fragmento XML bem formado?

a) <myElement myAttribute="someValue"/>
b) <myElement myAttribute="someValue’/>
c) <myElement myAttribute=’someValue’>

d) <myElement myAttribute=someValue/>
e) <myElement myAttribute=someValue>

60. **(QUADRIX / CRQ4-SP – 2018)** Uma vantagem do DTD é que ele é escrito em linguagem XML, enquanto o XML-Schema possui outra sintaxe de programação.

61. **(QUADRIX / CRQ4-SP – 2018)** Para a descrição de um XML, tanto o XML-Schema quanto a DTD (Document Type Definition) podem definir os elementos e atributos que podem aparecer em um documento, os tipos de dados para elementos e atributos e os valores-padrão e fixos para elementos e atributos.

62. **(CONSUPLAN / TRE-RJ – 2017)** A respeito de XML, é INCORRETO afirmar que:

a) Processar um documento XML requer um software parser XML (ou processador XML).
b) Todo documento XML deve ter exatamente um elemento-raiz que contém todos os outros elementos.
c) Apesar de documentos XML serem altamente portáveis, visualizar ou modificar documentos XML requer softwares especializados.
d) Um documento XML pode referenciar uma Definição de Tipo de Documento (DTD) ou um esquema que define a estrutura adequada do documento XML.

63. **(UFMT / UFSBA – 2017)** Sobre XML (eXtended Markup Language), assinale a afirmativa correta.

a) Possui tecnologias para auxílio na execução de seu código, como DTD (Document Type Definition) e XML Schema.
b) É uma tecnologia recomendada pela W3C, projetada para armazenar e transportar dados.
c) É uma evolução do HTML, por isso páginas HTML migraram para páginas XHTML.
d) É estruturada na forma de árvore, mas permite a existência de mais de um nó raiz no documento.

64. **(VUNESP / Prefeitura de Presidente Prudente-SP – 2016)** A Definição de Tipo de Documento (DTD) é utilizada no XML para:

a) especificar as transformações a serem aplicadas para reestruturar o documento em um novo formato.
b) validar a estrutura do documento.
c) armazenar valores com mais de 65535 bytes.
d) interligar múltiplos documentos.
e) associar código JavaScript ao documento.

65. **(CCV / UFC – 2016)** Em um documento XML, o que a DTD representa?

a) Direct Type Definition.
b) Direct Type Document.

c) Dynamic Type Definition.
d) Document Type Definition.
e) Dynamic Type Document.

66. **(FUNRIO / IF-PA – 2016)** Em relação as regras de sintaxe do XML são apresentadas as seguintes proposições:

I – Todo documento XML deve conter um elemento raiz que é o pai de todos os outros elementos.
II – Os elementos do XML não precisam estar devidamente aninhados.
III – Os valores dos atributos devem sempre estar entre aspas.

É correto apenas o que se afirma em

a) I.
b) II.
c) III.
d) I e II.
e) I e III.

GABARITO

1.	LETRA B	23.	LETRA E	45.	LETRA E
2.	CORRETO	24.	LETRA D	46.	LETRA C
3.	ERRADO	25.	LETRA E	47.	LETRA C
4.	ERRADO	26.	LETRA A	48.	LETRA C
5.	CORRETO	27.	LETRA D	49.	LETRA E
6.	ERRADO	28.	ERRADO	50.	LETRA E
7.	ERRADO	29.	LETRA B	51.	LETRA B
8.	LETRA B	30.	LETRA B	52.	LETRA B
9.	LETRA E	31.	LETRA A	53.	LETRA B
10.	LETRA E	32.	LETRA A	54.	LETRA D
11.	LETRA E	33.	LETRA C	55.	LETRA A
12.	LETRA B	34.	LETRA A	56.	LETRA D
13.	LETRA A	35.	LETRA E	57.	LETRA B
14.	LETRA D	36.	LETRA A	58.	LETRA A
15.	LETRA D	37.	LETRA B	59.	LETRA A
16.	LETRA D	38.	LETRA C	60.	ERRADO
17.	LETRA C	39.	LETRA D	61.	CORRETO
18.	LETRA A	40.	LETRA E	62.	LETRA C
19.	LETRA B	41.	LETRA C	63.	LETRA B
20.	LETRA D	42.	LETRA B	64.	LETRA B
21.	LETRA D	43.	LETRA C	65.	LETRA D
22.	LETRA D	44.	LETRA C	66.	LETRA E

XSLT

Galera, XSL (eXtensible Stylesheet Language) é uma linguagem de folha de estilo para Documentos XML. Calma, professor! O que é uma folha de estilo? Resumindo, trata-se de padrão que define a apresentação de um documento! Imaginem o seguinte: você tem que escrever a sua dissertação de graduação da faculdade. Existem duas abordagens para escrevê-la!

Você pode ir escrevendo o conteúdo e já ir mexendo no layout, i.e., a cada parágrafo, você modifica o tamanho da letra, modifica a cor, define um espaçamento, coloca uma figura, modifica sua posição, e assim por diante. Outra maneira, é construir uma folha de estilo já com todas essas informações de layout (fonte, tamanho, cor, etc) e depois somente aplicar ao texto de sua monografia.

Vocês percebem a diferença? Se eu quiser fazer depois a minha dissertação de mestrado, basta eu aplicar essa mesma folha de estilo! Se eu utilizasse a outra abordagem, eu teria que fazer novamente a mesma configuração da apresentação, layout, etc. Portanto, as folhas de estilo existem para tornar a apresentação independente do conteúdo e facilitar nossa vida. Vamos ver um exemplo:

Código XML: apenas com conteúdo (catalogocd.xml)

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

<?xml-stylesheet type=”text/xsl” href=”catalogocd.xsl”?>


<catalogo>

   <cd>

      <titulo> Sgt. Peppers </titulo>

      <artista> The Beatles </artista>

   </cd>

   <cd>

      <titulo> Ride the Lightning </titulo>

      <artista> Metallica </artista>

   </cd>

</catalogo>
```

Código XSL: apenas com apresentação (catalogocd.xsl)

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


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


<xsl:template match="/">
 <html>
 <body>
 <h2>Minha Coleção de CD</h2>
  <table border="1">
   <tr bgcolor="#9acd32">
    <th style="text-align:left">Titulo</th>
    <th style="text-align:left">Artista</th>
   </tr>
   <xsl:for-each select="catalogo/cd">
   <tr>
    <td><xsl:value-of select="titulo"/></td>
    <td><xsl:value-of select="artista"/></td>
   </tr>
   </xsl:for-each>
  </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

Resultado Final: aplicação do XSL no XML!

Minha Coleção de CD

Titulo	Artista
Sgt. Peppers	The Beatles
Ride the Lightning	Metallica

XSL é uma linguagem de folha de estilo para Documentos XML (assim como o CSS é uma linguagem de folha de estilo para Documentos HTML). Ela se divide em XSLT (Transformação XML), XPath (Navegação XML) e XSL-FO (Formatação XML). Agora voltando para nosso papo principal... o que seria o eXtensible Stylesheet Language for Transformation (XSLT)?

Trata-se de uma linguagem para transformação de Documentos XML em outros formatos reconhecidos por um navegador web (XML, XHTML, HTML e outros). Em geral, ele faz isso ao transformar cada Elemento XML em Elemento (X)HTML. É possível adicionar ou remover elementos e atributos de/para um arquivo de saída, ou mesmo reorganizar elementos, executar testes e muito mais.

Galera, vocês sabem que Documentos XML podem ser representados como uma árvore. Dessa forma, podemos dizer que o XSLT transforma uma Árvore-Fonte XML em uma Árvore-Resultado XML. Ademais, é importante ressaltar que ela utiliza o XPath para encontrar informações em um Documento XML. E como se busca informações nessa árvore? Por meio do XPath!

XSLT é suportado por diversos navegadores, portanto fiquem tranquilos. Professor, como eu faço para declarar uma folha de estilo? Podemos fazer de duas maneiras:

Utilizando <xsl:stylesheet> - raiz do Documento XML;

```
<xsl:stylesheet version="1.0"

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

Ou utilizando <xsl:transform> - raiz do Documento XML;

```
<xsl:transform version="1.0"

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

Observem que o atributo XMLNS possui um prefixo chamado XSL, seguida de uma URI. Em outras palavras, aponta para um namespace que contém um XML Schema que diz se o documento é válido e bem formado. Percebam que, para acessar os elementos, atributos, entre outros devemos declarar o namespace do XSLT no início do documento.

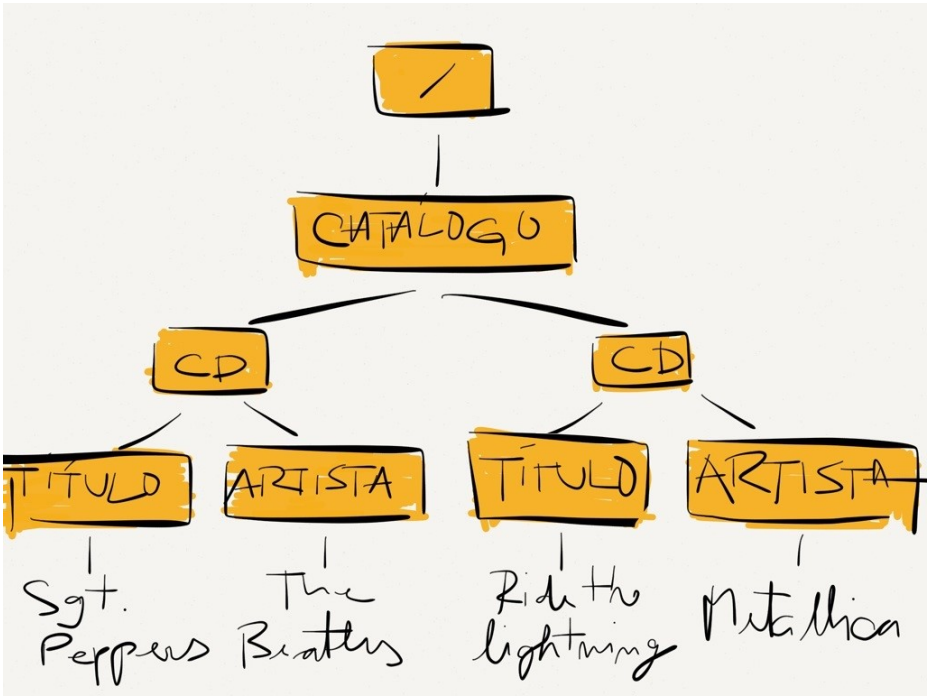
Galera, nós também podemos declarar um template através do elemento <xsl:template> – um template é um conjunto de uma ou mais regras aplicadas ao Documento XML. Esse elemento

possui um atributo chamado MATCH, que é utilizado para associar um template com um elemento XML ou com o documento XML inteiro. Lembram do exemplo lá de cima? O que significa essa barra?

```
<xsl:template match="/">
```

Vocês se lembram que eu disse que se utiliza o XPath para navegar pelo Documento XML? Pois é, ele trata o documento como uma árvore cheia de nós, como mostra a imagem abaixo. Se o código diz que é a partir do "/", isso significa que se trata do documento inteiro. Observem também que o código está transformando um Documento XML em um Documento HTML.

O Elemento <xsl:for-each> seleciona elementos XML de um conjunto de nós. O atributo SELECT seleciona o nó por meio do XPath. No nosso caso, ele manda selecionar "catalogo/cd/artista" e "catalogo/cd/titulo" e monta uma tabela mostrada ao final! É tranquilo de entender, concordam? Por fim, o elemento <xsl:value-of> extrai o valor de um elemento e adicioná-lo à saída da transformação.



Galera, como fazer o browser aplicar automaticamente uma folha de estilo ao XML? Adiciona-se o comando abaixo para linkar o Arquivo XML ao Arquivo XSL e realizar a transformação no navegador especificado:

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

QUESTÕES COMENTADAS - XSLT - MULTIBANCAS

1. (CESGRANRIO - 2010 – PETROBRÁS – Analista de Sistemas – B) No contexto de linguagens de marcação, transformação e apresentação, tem-se que uma transformação expressa em XSLT descreve regras para transformar uma árvore fonte em uma árvore resultado.

Comentários:

Galera, vocês sabem que Documentos XML podem ser representados como uma árvore. Dessa forma, podemos dizer que o XSLT transforma uma Árvore-Fonte XML em uma Árvore-Resultado XML. Ademais, é importante ressaltar que ela utiliza o XPath para encontrar informações em um Documento XML. E como se busca informações nessa árvore? Por meio do XPath! Conforme discutido em aula. Gabarito: C

2. (CESGRANRIO - 2012 – PETROBRÁS – Analista de Sistemas) SOA e Web Services utilizam interfaces de serviço para definir o que será solicitado e o que deve ser retornado como resultado do processamento do serviço. No entanto, problemas surgem quando a SOA e os consumidores de Web Services se baseiam em estruturas de dados que possuem certas discrepâncias. Qual a tecnologia usada para resolver esse tipo de problema?

a) DTD

b) XSLT

c) XQuery

d) XLink

e) XSL-FO

Comentários:

Questão para pensar:

Que tecnologia é responsável por resolver o problema de discrepância na troca de dados entre consumidores de Web Services? Que tecnologia pode transformar um arquivo em outro para resolver o problema? XSLT! Gabarito: B

3. (CESPE - 2009 – TRE/MA – Analista de Sistemas) Considerando o trecho de código acima apresentado, assinale a opção correta.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
    <html>
    <body>
        <h2>Meus Filmes</h2>
        <table border="1">
        <tr bgcolor="#9acd32">
        <th>Título</th>
        <th>Ator Principal</th>
        </tr>
        <xsl:for-each select="filmes/dados">
        <tr>
        <td><xsl:value-of select="titulo"/></td>
        <td><xsl:value-of
        select="atorprincipal"/></td>
        </tr>
        </xsl:for-each>
        </table>
    </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```

a) O código, escrito em XSLT, necessita de um arquivo CSS que contenha, no mínimo, um javascript que modifique a tabela com as tags titulo e atorprincipal para gerar uma tabela de saída informando o conteúdo das tags processadas.

b) Para funcionar corretamente, esse código, escrito em XML, necessita de um arquivo XSTL que contenha, no mínimo, as tags filmes e dados. Por sua vez, na tag de dados, devem existir tags de titulo e atorprincipal em CSS para gerar uma tabela de saída informando o conteúdo das tags processadas.

c) Para funcionar corretamente, esse código, escrito em HTML, necessita de um arquivo XML que contenha, no mínimo, as tags XSLT filmes e dados. Por sua vez, na tag de dados, devem existir tags de titulo e atorprincipal em CSS para gerar uma tabela de saída informando o conteúdo das tags processadas.

d) Para funcionar corretamente, esse código, escrito em XSLT, necessita de um arquivo XML que contenha, no mínimo, as tags XML filmes e dados. Por sua vez, na tag de dados, devem existir tags de titulo e atorprincipal para gerar uma tabela de saída informando o conteúdo das tags processadas.

e) O código, escrito em XSLT/javascript, necessita, para funcionar corretamente, de um arquivo HTML que contenha, no mínimo, as linhas filmes e dados. Por sua vez, na linha de dados, devem existir variáveis com o nome titulo e atorprincipal para gerar uma tabela de saída informando o conteúdo das tags processadas.

Comentários:

(a) Não, ele não necessita de arquivo CSS algum, nem comando Javascript; (b) Não, nada de CSS; (c) Não, escrito em XML ou XSLT; as tags XML filmes e dados; e nada de CSS; (d) Sim, agora está correto; (e) Não, escrito em XSLT ou XML; nada de HTML; nada de variáveis, são tags. **Gabarito: D**

4. (CESPE - 2010 – INMETRO – Analista de Sistemas – C) A XSLT permite transformar um documento XML em HTML, texto simples ou qualquer outro documento embasado em texto.

Comentários:

Perfeita definição de XSLT! Gabarito: C

5. (CESPE - 2010 – INMETRO – Analista de Sistemas – D) O código abaixo descreve corretamente uma aplicação em XSLT.

```
<?xml version ="1.0" ?>
<xsl:template match="myMessage">
   <html>
        <body> <xsl: value-of- select="Message">
        </body>
        </html>
</xsl:template>
```

Comentários:

Não, falta a declaração da Folha de estilos e seu namespace. Gabarito: E

6. (CESPE - 2010 – MPU – Analista de Sistemas) Um arquivo XSLT (Extensible Stylesheet Language Transformation) permite transformar os dados de um arquivo XML. A maneira correta de se referir a um arquivo de estilo denominado mpuestilo.xml em um arquivo XML é mostrada a seguir.

<stylesheet type="text/xsl" href="mpuestilo.xsl">

Comentários:

Não, seria assim:

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

Gabarito: E

7. (CESPE - 2010 – TRE/BA – Analista de Sistemas) O documento XSLT é necessário para a definição da estrutura de um documento XML.

Comentários:

Não, quem define essa estrutura é o XML Schema. Gabarito: E

8. (CESPE - 2010 – TRE/MT – Analista de Sistemas) A respeito de XSLT (eXtensible Stylesheet Language Transformation), assinale a opção correta.

a) Uma transformação na linguagem XSLT é expressa na forma de uma folha de estilo, cuja sintaxe utiliza XML.

b) XSLT é uma linguagem para transformar somente documentos XHTML em documentos HTML.

c) A transformação XSLT deve respeitar a estrutura da árvore de origem, ou seja, a árvore de destino não pode ter uma estrutura diferente da árvore de origem.

d) Uma transformação expressa em XSLT descreve regras para transformar uma ou mais árvores de origem em uma e somente uma árvore de destino.

e) O seguinte trecho é correto.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template for-each="/">
 <html>
   <body>
      <p>Biblioteca</p>
      <table border="1">
         <tr>
          <th>Titulo</th>
          <th>Autor</th>
         </tr>
        <xsl:match select="biblioteca/livro">
        <tr>
          <td><xsl:value-of select="titulo"/></td>
          <td><xsl:value-of select="autor"/></td>
        </tr>
        </xsl:for-each>
      </table>
   </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

Comentários:

(a) Sim, é exatamente isso; (b) Não, para transformar XML em outros documentos baseados em texto; (c) Não, a árvore de destino pode ser diferente; (d) Não, é uma relação N:1, i.e., uma ou mais árvores de origem em uma única árvore de destino; (e) Não! Não existe <xsl:template for-each="/", mas sim <xsl:template match="/". Além disso, está errado <xsl:match select...>, seria <xsl:for-each select...> - enfim, não faz sentido algum. Gabarito: A

9. (CESPE - 2011 – MEC – Analista de Sistemas) Referenciada na e-ping, a XSLT é uma linguagem que transforma documentos XML em outros documentos XML, o que permite o intercâmbio de informações e a interoperabilidade entre sistemas.

Comentários:

Perfeito, nada a acrescentar. Gabarito: C

10.(CESPE - 2011 – TJ/ES – Analista de Sistemas) XSLT é um subconjunto do XML Schema que permite transformar documentos XML em outros formatos como PDF, HTML ou mesmo outro XML. Para tanto, o XSLT define, entre outros aspectos, a forma como os documentos XML são acessados.

Comentários:

Na verdade, é um subconjunto do XSL. Ademais, a forma de acesso é dada pelo XPath; o XSLT trata da transformação. Gabarito: E

11.(CESPE - 2012 – TJ/RO – Analista de Sistemas – B) <?xml> é exemplo de tag do tipo root, necessária para identificar o documento como XSL na linguagem XSLT 2.0.

Comentários:

Não, o elemento raiz seria assim:

Utilizando <xsl:stylesheet> - raiz do Documento XML;

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

Ou utilizando <xsl:transform> - raiz do Documento XML;

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

Gabarito: E

12.(CESPE - 2012 – TJ/RO – Analista de Sistemas – C) Na linguagem XSLT 2.0, o elemento <xsl:value-of> pode ser utilizado para extrair o valor de um elemento XML e adicioná-lo ao documento de saída, resultado da transformação.

Comentários:

O Elemento <xsl:for-each> seleciona elementos XML de um conjunto de nós. O atributo SELECT seleciona o nó por meio do XPath. No nosso caso, ele manda selecionar "catalogo/cd/artista" e "catalogo/cd/titulo" e monta uma tabela mostrada ao final! É tranquilo de entender, concordam? Por fim, o elemento <xsl:value-of> extrai o valor de um elemento e adicioná-lo à saída da transformação.

Conforme visto em aula, está perfeito! Gabarito: C

13.(CESPE - 2013 – TCE/ES – Analista de Sistemas – B) No processo de transformação realizado pelo XSLT, os templates de regras devem ser construídos de forma a não existirem conflitos de regras, visto que esses conflitos não podem ser automaticamente resolvidos.

Comentários:

Podem, sim – por meio de namespaces. Gabarito: E

14.(FCC - 2006 – TRT/MS – Analista de Sistemas) A linguagem ..... especifica a utilização de um documento ....., utilizando a linguagem ..... para descrever como o documento será transformado em outro documento ..... .

As lacunas acima serão corretamente preenchidas por, respectivamente,

a) XML, XML, XSLT e XSL.
b) XML, XML, XSL e XSLT.
c) XSL, XML, XSLT e XML.
d) XSLT, XML, XSL e XML.
e) XSL, XML, XSLT e XSL.

Comentários:

A linguagem XSL especifica a utilização de um documento XML, utilizando a linguagem XSLT para descrever como o documento será transformado em outro documento XML. Gabarito: C

15.(FGV - 2009 – MEC – Analista de Sistemas) A XSLT (eXtensible Stylesheet Language: Transformations) é uma linguagem usada para transformar a estrutura de um documento XML. Essa transformação é realizada por um processador XSLT. O papel principal de um processador XSLT é aplicar uma folha de estilo XSLT em um documento fonte XML e produzir um documento resultante.

Assinale a alternativa que não apresente um processador XSLT:

a) Xalan.
b) Saxon.
c) XTLTX.
d) MSXML3.
e) Sablotron.

Comentários:

Não coloquei na teoria porque é um tema muito específico e quase não cai em prova. Os processadores XSLT mais famosos são: Xalan, Saxon, MSXML e Sablotron. Eles são os responsáveis de fato por processar os arquivos de origem no arquivo de destino. Gabarito: C

16.(FGV - 2013 – AL/MA – Analista de Sistemas) Com relação as folhas de estilo XSLT, analise as afirmativas a seguir.

I. Namespaces podem ser usados em folhas de estilo XSLT.

II. Folhas de estilo XSLT devem ser documentos XML bem formados.

III. Somente documentos XML podem ser gerados como saída do processamento de folhas de estilo XSLT.

Assinale:

a) se somente a afirmativa I estive correta.

b) se somente a afirmativa II estiver correta.

c) se somente a afirmativa III estiver correta.

d) se somente as afirmativas I e II estiverem corretas

e) se todas as afirmativas estiverem corretas.

Comentários:

(I) Sim, claro que podem; (II) Sim, eles devem ser bem formados; (III) Não, qualquer documento baseado em texto. Gabarito: D

17.(CESPE - 2004 – ABIN – Analista de Sistemas) As folhas de estilo escritas em linguagem XSL (extensible stylesheet language) permitem definir várias formatações alternativas para documentos escritos em linguagem XML (extensible mark-up language).

Comentários:

XSL é uma linguagem de folha de estilo para Documentos XML (assim como o CSS é uma linguagem de folha de estilo para Documentos HTML). Ela se divide em XSLT (Transformação XML), XPath (Navegação XML) e XSL-FO (Formatação XML). Agora voltando para nosso papo principal... o que seria o eXtensible Stylesheet Language for Transformation (XSLT)?

Trata-se de uma linguagem para transformação de Documentos XML em outros formatos reconhecidos por um navegador web (XML, XHTML, HTML e outros). Em geral, ele faz isso ao transformar cada Elemento XML em Elemento (X)HTML. É possível adicionar ou remover elementos e atributos de/para um arquivo de saída, ou mesmo reorganizar elementos, executar testes e muito mais.

Perfeito, conforme visto em aula. Gabarito: C

18.(FCC - 2011 – TCE/PR – Analista de Sistemas) Em padrões XML, style sheets são ferramentas utilizadas nos padrões:

a) XSLT e XBRL.

b) WSDL e SOAP.

c) XSLT e SOAP.

d) XBRL e WSDL.

e) XSLT e WSDL.

Comentários:

Galera, XSLT é a resposta óbvia, portanto eliminamos os itens B e D. Vocês sabem o que é XBRL? Trata-se de Extensible Business Reporting Language! Professor, você não falou disso em aula! Galera, é a única vez que vi isso em prova – preferi falar disso só nessa questão. XBRL é um padrão baseado em XML para definir a informação financeira – nós usamos bastante aqui nos sistemas da Secretaria do Tesouro Nacional (STN). Logo, se é XML, pode receber definições de estilo.
Gabarito: A

19.(CESPE - 2016 – TCE/SC – Analista de Sistemas) O XSLT é utilizado para adicionar e(ou) remover elementos e atributos do arquivo de saída e para transformar um documento XML em um documento HTML ou XHTML, ou, ainda, em outro documento XML.

Comentários:

XSLT é uma linguagem para transformação de Documentos XML em outros formatos reconhecidos por um navegador web (XML, XHTML, HTML e outros). Em geral, ele faz isso ao transformar cada Elemento XML em Elemento (X)HTML. É possível adicionar ou remover elementos e atributos de/para um arquivo de saída, ou mesmo reorganizar elementos, executar testes, entre outros.
Gabarito: C

LISTA DE QUESTÕES - XSLT - MULTIBANCAS

1. (CESGRANRIO - 2010 – PETROBRÁS – Analista de Sistemas – B) No contexto de linguagens de marcação, transformação e apresentação, tem-se que uma transformação expressa em XSLT descreve regras para transformar uma árvore fonte em uma árvore resultado.

2. (CESGRANRIO - 2012 – PETROBRÁS – Analista de Sistemas) SOA e Web Services utilizam interfaces de serviço para definir o que será solicitado e o que deve ser retornado como resultado do processamento do serviço. No entanto, problemas surgem quando a SOA e os consumidores de Web Services se baseiam em estruturas de dados que possuem certas discrepâncias. Qual a tecnologia usada para resolver esse tipo de problema?

   a) DTD

   b) XSLT

   c) XQuery

   d) XLink

   e) XSL-FO

3. (CESPE - 2009 – TRE/MA – Analista de Sistemas) Considerando o trecho de código acima apresentado, assinale a opção correta.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
    <html>
    <body>
        <h2>Meus Filmes</h2>
        <table border="1">
        <tr bgcolor="#9acd32">
        <th>Título</th>
        <th>Ator Principal</th>
        </tr>
        <xsl:for-each select="filmes/dados">
        <tr>
        <td><xsl:value-of select="titulo"/></td>
        <td><xsl:value-of
        select="atorprincipal"/></td>
        </tr>
        </xsl:for-each>
        </table>
    </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```

a) O código, escrito em XSLT, necessita de um arquivo CSS que contenha, no mínimo, um javascript que modifique a tabela com as tags titulo e atorprincipal para gerar uma tabela de saída informando o conteúdo das tags processadas.

b) Para funcionar corretamente, esse código, escrito em XML, necessita de um arquivo XSTL que contenha, no mínimo, as tags filmes e dados. Por sua vez, na tag de dados, devem existir tags de titulo e atorprincipal em CSS para gerar uma tabela de saída informando o conteúdo das tags processadas.

c) Para funcionar corretamente, esse código, escrito em HTML, necessita de um arquivo XML que contenha, no mínimo, as tags XSLT filmes e dados. Por sua vez, na tag de dados, devem existir tags de titulo e atorprincipal em CSS para gerar uma tabela de saída informando o conteúdo das tags processadas.

d) Para funcionar corretamente, esse código, escrito em XSLT, necessita de um arquivo XML que contenha, no mínimo, as tags XML filmes e dados. Por sua vez, na tag de dados, devem existir tags de titulo e atorprincipal para gerar uma tabela de saída informando o conteúdo das tags processadas.

e) O código, escrito em XSLT/javascript, necessita, para funcionar corretamente, de um arquivo HTML que contenha, no mínimo, as linhas filmes e dados. Por sua vez, na linha de dados, devem existir variáveis com o nome titulo e atorprincipal para gerar uma tabela de saída informando o conteúdo das tags processadas.

4. (CESPE - 2010 – INMETRO – Analista de Sistemas – C) A XSLT permite transformar um documento XML em HTML, texto simples ou qualquer outro documento embasado em texto.

5. (CESPE - 2010 – INMETRO – Analista de Sistemas – D) O código abaixo descreve corretamente uma aplicação em XSLT.

```
<?xml version ="1.0" ?>
<xsl:template match="myMessage">
   <html>
        <body> <xsl: value-of- select="Message">
       </body>
       </html>
</xsl:template>
```

6. (CESPE - 2010 – MPU – Analista de Sistemas) Um arquivo XSLT (Extensible Stylesheet Language Transformation) permite transformar os dados de um arquivo XML. A maneira correta de se referir a um arquivo de estilo denominado mpuestilo.xml em um arquivo XML é mostrada a seguir.

<stylesheet type="text/xsl" href="mpuestilo.xsl">

7. (CESPE - 2010 – TRE/BA – Analista de Sistemas) O documento XSLT é necessário para a definição da estrutura de um documento XML.

8. (CESPE - 2010 – TRE/MT – Analista de Sistemas) A respeito de XSLT (eXtensible Stylesheet Language Transformation), assinale a opção correta.

a) Uma transformação na linguagem XSLT é expressa na forma de uma folha de estilo, cuja sintaxe utiliza XML.

b) XSLT é uma linguagem para transformar somente documentos XHTML em documentos HTML.

c) A transformação XSLT deve respeitar a estrutura da árvore de origem, ou seja, a árvore de destino não pode ter uma estrutura diferente da árvore de origem.

d) Uma transformação expressa em XSLT descreve regras para transformar uma ou mais árvores de origem em uma e somente uma árvore de destino.

e) O seguinte trecho é correto.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template for-each="/">
 <html>
   <body>
      <p>Biblioteca</p>
      <table border="1">
         <tr>
          <th>Titulo</th>
          <th>Autor</th>
         </tr>
        <xsl:match select="biblioteca/livro">
        <tr>
          <td><xsl:value-of select="titulo"/></td>
          <td><xsl:value-of select="autor"/></td>
        </tr>
        </xsl:for-each>
       </table>
  </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

9. (CESPE - 2011 – MEC – Analista de Sistemas) Referenciada na e-ping, a XSLT é uma linguagem que transforma documentos XML em outros documentos XML, o que permite o intercâmbio de informações e a interoperabilidade entre sistemas.

10. (CESPE - 2011 – TJ/ES – Analista de Sistemas) XSLT é um subconjunto do XML Schema que permite transformar documentos XML em outros formatos como PDF, HTML ou mesmo outro XML. Para tanto, o XSLT define, entre outros aspectos, a forma como os documentos XML são acessados.

11. (CESPE - 2012 – TJ/RO – Analista de Sistemas – B) <?xml> é exemplo de tag do tipo root, necessária para identificar o documento como XSL na linguagem XSLT 2.0.

12.(CESPE - 2012 – TJ/RO – Analista de Sistemas – C) Na linguagem XSLT 2.0, o elemento <xsl:value-of> pode ser utilizado para extrair o valor de um elemento XML e adicioná-lo ao documento de saída, resultado da transformação.

13.(CESPE - 2013 – TCE/ES – Analista de Sistemas – B) No processo de transformação realizado pelo XSLT, os templates de regras devem ser construídos de forma a não existirem conflitos de regras, visto que esses conflitos não podem ser automaticamente resolvidos.

14.(FCC - 2006 – TRT/MS – Analista de Sistemas) A linguagem ..... especifica a utilização de um documento ....., utilizando a linguagem ..... para descrever como o documento será transformado em outro documento ..... .

As lacunas acima serão corretamente preenchidas por, respectivamente,

a) XML, XML, XSLT e XSL.

b) XML, XML, XSL e XSLT.

c) XSL, XML, XSLT e XML.

d) XSLT, XML, XSL e XML.

e) XSL, XML, XSLT e XSL.

15.(FGV - 2009 – MEC – Analista de Sistemas) A XSLT (eXtensible Stylesheet Language: Transformations) é uma linguagem usada para transformar a estrutura de um documento XML. Essa transformação é realizada por um processador XSLT. O papel principal de um processador XSLT é aplicar uma folha de estilo XSLT em um documento fonte XML e produzir um documento resultante.

Assinale a alternativa que não apresente um processador XSLT:

a) Xalan.

b) Saxon.

c) XTLTX.

d) MSXML3.

e) Sablotron.

16.(FGV - 2013 – AL/MA – Analista de Sistemas) Com relação as folhas de estilo XSLT, analise as afirmativas a seguir.

I. Namespaces podem ser usados em folhas de estilo XSLT.

II. Folhas de estilo XSLT devem ser documentos XML bem formados.

III. Somente documentos XML podem ser gerados como saída do processamento de folhas de estilo XSLT.

Assinale:

a) se somente a afirmativa I estive correta.

b) se somente a afirmativa II estiver correta.

c) se somente a afirmativa III estiver correta.

d) se somente as afirmativas I e II estiverem corretas

e) se todas as afirmativas estiverem corretas.

17. (CESPE - 2004 – ABIN – Analista de Sistemas) As folhas de estilo escritas em linguagem XSL (extensible stylesheet language) permitem definir várias formatações alternativas para documentos escritos em linguagem XML (extensible mark-up language).

18. (FCC - 2011 – TCE/PR – Analista de Sistemas) Em padrões XML, style sheets são ferramentas utilizadas nos padrões:

a) XSLT e XBRL.

b) WSDL e SOAP.

c) XSLT e SOAP.

d) XBRL e WSDL.

e) XSLT e WSDL.

19. (CESPE - 2016 – TCE/SC – Analista de Sistemas) O XSLT é utilizado para adicionar e(ou) remover elementos e atributos do arquivo de saída e para transformar um documento XML em um documento HTML ou XHTML, ou, ainda, em outro documento XML.

GABARITO

1. C
2. B
3. D
4. C
5. E
6. E
7. E
8. A
9. C
10. E
11. E
12. C
13. E
14. C
15. C
16. D
17. C
18. A
19. C

JSON

Conceitos Básicos

INCIDÊNCIA EM PROVA: MÉDIA

Hoje vamos falar sobre ele...

O JSON (lê-se Jason, como o protagonista do filme Sexta-Feira 13) é o acrônimo para JavaScript Object Notation. Trata-se basicamente de um formato leve, compacto, de padrão aberto, baseado em texto, auto-descritível, fácil de entender (por máquinas e por humanos) e independente de linguagem de programação para armazenamento, transporte, compartilhamento e intercâmbio de dados estruturados entre sistemas.

Como podemos ver pelo próprio acrônimo, trata-se de uma Notação de Objetos JavaScript. Galera, JavaScript é uma linguagem de programação utilizada na web para tornar páginas web mais dinâmicas. *Sabe quando uma página web possui conteúdos dinâmicos, áudio, vídeo, imagens animadas, entre outros?* Pois é, quem faz isso é o JavaScript! Hoje em dia, páginas web mal funcionam sem JavaScript. *Professor, preciso entender essa linguagem de programação?* Não!

Eu só falei isso porque é interessante saber que o JSON surgiu a partir do JavaScript/ECMAScript. Essa linguagem de programação possuía uma sintaxe bem leve e tranquila para criação de objetos. Alguém observou essa característica e pensou: *e se utilizássemos essa sintaxe para intercâmbio de dados de propósito geral?* Foi aí que ele se tornou um formato simples e autônomo para intercâmbio de dados completamente independente de linguagem de programação (e outras tecnologias).

JSON é apenas um formato de texto leve e compacto, logo é fácil de ser enviado e transportado por computadores. Ele não tenta impor representações de dados para outras linguagens de programação. Em vez disso, ele compartilha um pequeno subconjunto da sintaxe original do JavaScript/ECMAScript com outras linguagens de programação e tecnologias associadas seguindo um conjunto de regras de estruturação para a representação portátil de dados estruturados

A partir de 2017, muitas linguagens de programação incluíram nativamente códigos para ler, gerar, analisar sintaticamente e realizar conversões de dados em JSON.

JSON x XML

INCIDÊNCIA EM PROVA: MÉDIA

Existem diversos formatos para intercâmbio de dados, sendo que o mais famoso até a década passada era o XML (eXtensible Markup Language). Vamos ver um exemplo de cada:

CÓDIGO EM XML

```
<servidores>
  <servidor>
    <primeiroNome>Diego</primeiroNome> <ultimoNome>Carvalho</ultimoNome>
  </servidor>
  <servidor>
    <primeiroNome>Renato</primeiroNome> <ultimoNome>da Costa</ultimoNome>
  </servidor>
  <servidor>
    <primeiroNome>Thiago</primeiroNome> <ultimoNome>Cavalcanti</ultimoNome>
  </servidor>
</servidores>
```

CÓDIGO EM JSON

```
{"servidores":[
  { "primeiroNome":"Diego", "ultimoNome":"Carvalho" },
  { "primeiroNome ":"Renato", " ultimoNome":"da Costa" },
  { "primeiroNome ":"Thiago", " ultimoNome":"Cavalcanti" }
]}
```

JSON e XML são considerados concorrentes – eles possuem diversas semelhanças e diferenças listadas nas tabelas a seguir:

SEMELHANÇAS
Representam informações no formato texto com uma natureza auto-descritiva.
Ambos são capazes de representar informações complexas difíceis de representar no formato tabular como relações de hierarquia, dados ausentes, entre outros.
Ambos podem ser considerados padrões para representação de dados. XML é um padrão W3C, enquanto JSON foi formalizado na RFC 4627.
Ambos são independentes de linguagem. Dados representados em XML e JSON podem ser acessados por qualquer linguagem de programação, através de API's específicas.

DIFERENÇAS
JSON não é uma linguagem de marcação – não possui tags de abertura e de fechamento.
JSON representa as informações de forma mais compacta – basta visualizar os códigos acima.
JSON não permite a execução de instruções de processamento, enquanto isso é possível em XML por meio do prolog.
JSON é tipicamente destinado para a troca de informações, enquanto XML possui mais aplicações. Por exemplo: existem bancos de dados no formato XML e estruturados em SGBD XML nativo.

Enfatizando mais uma vez: notem que a simplicidade do JSON ajudou a popularizar seu uso – especialmente como uma alternativa para XML – conforme podemos ver pelos códigos seguintes.

CÓDIGO EM XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <id>1</id>
  <nome>Diego Carvalho</nome>
  <endereco>Asa Norte</endereco>
```

CÓDIGO EM JSON

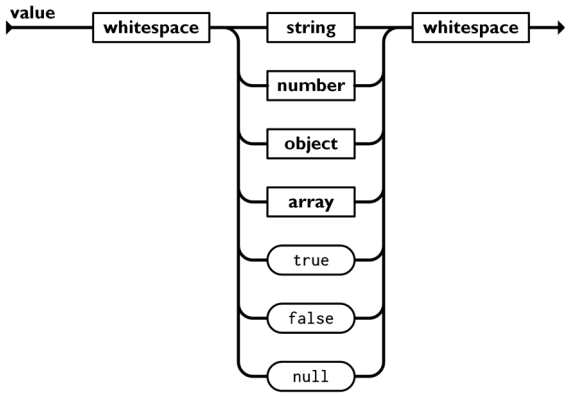
```json
{"id":1,"nome":"Diego Carvalho", "endereco":"Asa Norte"}
```

Sintaxe

INCIDÊNCIA EM PROVA: MÉDIA

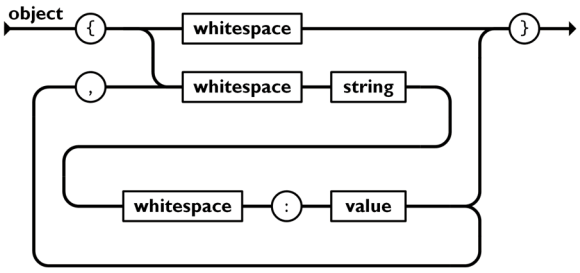
A sintaxe do JSON é extremamente, extremamente, extremamente simples! Ela obedece a um formato de **atributo:valor** em que, para cada valor representado, atribui-se um nome (ou rótulo) que descreve o seu significado. O nome do atributo deve vir sempre entre aspas duplas, seguido de dois-pontos, seguido do valor. Além disso, dados vêm separados por vírgulas, chaves representam objetos e colchetes representam arrays. Vejamos um formato básico a seguir:

```
{
    "nome": "Oscar Schimidt",
    "altura": 2.06
}
```



Dito isso, vamos falar agora sobre os tipos de valores. JSON possui seis tipos básicos de dados: **object**, **array**, **number**, **string**, **true**, **false** e **null**. Vamos detalhá-los a seguir...

Objetos

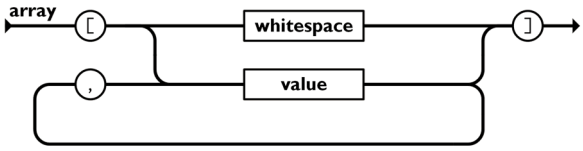


Um objeto é um conjunto desordenado de pares nome/valor (também chamado de chave/valor). Conforme podemos ver no diagrama anterior, a sintaxe de um objeto começa com uma chave de abertura (**{**) e termina com uma chave de fechamento (**}**), sendo que cada nome é seguido por um sinal de dois-pontos (**:**) e os pares nome/valor são seguidos por um sinal de vírgula (**,**). Além disso, um nome ou chave é uma string (sequência de caracteres).

Observe que a sintaxe do JSON não impõe nenhuma restrição às strings usadas como nomes, não exige que elas sejam exclusivas (apesar de ser uma boa prática) e não atribui qualquer significado à ordem dos pares nome/valor. Os objetos podem ser compostos por múltiplos pares nome/valor, por arrays e também por outros objetos. Dessa forma, um objeto pode representar, virtualmente, qualquer tipo de informação. Vejamos um objeto com cinco pares de chave/valor:

```
{
    "texto" : "Brasil",
    "numero" : 23,
    "numeroReal" : 54.87,
    "booleano": true,
    "nulo": null
}
```

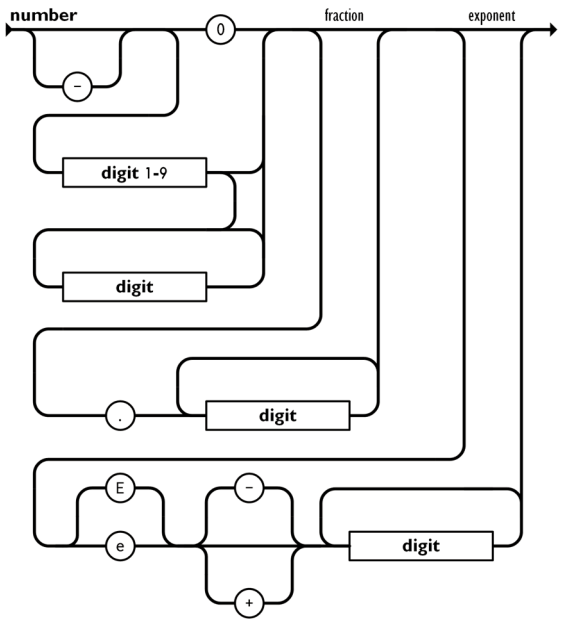
Arrays



Uma array (ou vetor) é uma coleção de zero ou mais valores ordenados! Ele começa com um colchete de abertura ([) e termina com um colchete de fechamento (]), sendo que os valores são separados por vírgula (,). No contexto de um array, o primeiro elemento é considerado como sendo de índice `0` e, não, `1`. No exemplo a seguir, temos um array com três valores: o primeiro tem o índice `0`, o segundo tem o índice `1` e o terceiro tem o índice `2`.

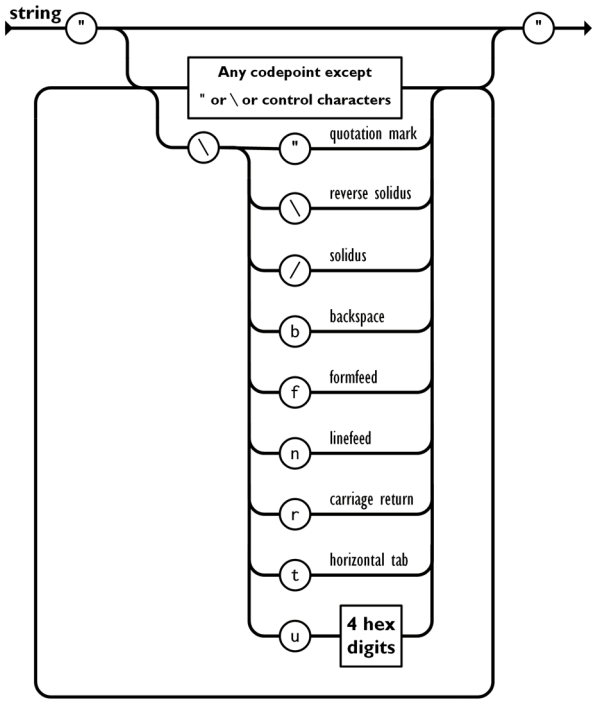
```
{
"servidores":["Diego", "Renato", "Thiago"]
}
```

Números



Um número é uma sequência de dígitos em formato decimal que não diferencia números inteiros de ponto flutuante. Pode conter sinal de negativo, parte fracionária, expoentes/potências, etc.

String



Uma string é uma cadeia de zero ou mais caracteres Unicode – elas são delimitadas por aspas duplas (") e suportam a barra inversa (\) como caractere de escape.

False, True e Null

JSON suporta valores booleanos: **true** e **false** – eles correspondem respectivamente aos valores lógicos verdadeiro e falso e não vêm entre aspas. E o valor **null** corresponde ao valor nulo, que indica a ausência de um valor. Como é obrigatório ter um par nome/valor, quando não existe um valor associado a um nome, não se pode deixá-lo em branco. Logo, o valor é indicado como **null** (lembrando que ele não vem entre aspas).

	A	B	C	D	E
1	Nome	Instrumentos	Idade	Cidade	
2	Syd Barret	Vocal, Guitarra		Cambridge	
3	David Gilmour	Guitarra, Saxofone, Bandolim	75	Cambridge	
4	Roger Waters	Baixo, Sintetizador	78	Surrey	
5	Richard Wright	Piano, Teclado, Hammond		Londres	
6	Nick Mason	Bateria, Bongo	77	Birmingham	
7					

Agora vamos fazer uma parada legal! Eu criei a planilha apresentada a seguir utilizando Microsoft Excel. Observe que ela contém dados sobre a maior banda de todos os tempos: **Pink Floyd**. Temos quatro colunas para representar o nome, instrumentos, idade e cidade natal dos cinco integrantes que já passaram pela banda. Em seguida, eu salve essa planilha como **.csv** e o converti tanto para **.xml** quanto para **.json**. Vamos ver os resultados:

CÓDIGO EM CSV

```
Nome;Instrumentos;Idade;Cidade
Syd Barret;Vocal, Guitarra;;Cambridge
David Gilmour;Guitarra, Saxofone, Bandolim;75;Cambridge
Roger Waters;Baixo, Sintetizador;78;Surrey
Richard Wright;Piano, Teclado, Hammond;;Londres
Nick Mason;Bateria, Bongo;77;Birmingham
```

CÓDIGO EM XML

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <row>
    <Nome>Syd Barret</Nome>
    <Instrumentos>Vocal, Guitarra</Instrumentos>
    <Idade></Idade>
    <Cidade>Cambridge</Cidade>
  </row>
  <row>
    <Nome>David Gilmour</Nome>
    <Instrumentos>Guitarra, Saxofone, Bandolim</Instrumentos>
    <Idade>75</Idade>
    <Cidade>Cambridge</Cidade>
  </row>
  <row>
```

```
    <Nome>Roger Waters</Nome>
    <Instrumentos>Baixo, Sintetizador</Instrumentos>
    <Idade>78</Idade>
    <Cidade>Surrey</Cidade>
  </row>
  <row>
    <Nome>Richard Wright</Nome>
    <Instrumentos>Piano, Teclado, Hammond</Instrumentos>
    <Idade></Idade>
    <Cidade>Londres</Cidade>
  </row>
  <row>
    <Nome>Nick Mason</Nome>
    <Instrumentos>Bateria, Bongo</Instrumentos>
    <Idade>77</Idade>
    <Cidade>Birmingham</Cidade>
  </row>
</root>
```

CÓDIGO EM JSON

```
[
 {
   "Nome": "Syd Barret",
   "Instrumentos": "Vocal, Guitarra",
   "Idade": null,
   "Cidade": "Cambridge"
 },
 {
   "Nome": "David Gilmour",
   "Instrumentos": "Guitarra, Saxofone, Bandolim",
   "Idade": 75,
   "Cidade": "Cambridge"
 },
 {
   "Nome": "Roger Waters",
   "Instrumentos": "Baixo, Sintetizador",
   "Idade": 78,
   "Cidade": "Surrey"
 },
 {
   "Nome": "Richard Wright",
   "Instrumentos": "Piano, Teclado, Hammond",
   "Idade": null,
   "Cidade": "Londres"
 },
 {
   "Nome": "Nick Mason",
   "Instrumentos": "Bateria, Bongo",
   "Idade": 77,
   "Cidade": "Birmingham"
 }
]
```

Vamos comentar? Note que temos colchetes, logo se trata de um array. Dentro dos colchetes, nós temos os dados dos músicos entre chaves, logo se trata de objetos. Dentro de cada objeto, temos pares de nome e valor. Podemos observar tipos string (Ex: Nome); tipos numéricos (Ex: Idade) e

tipos nulos (Ex: Idade). Nesse último caso, como os músicos já faleceram, não há uma idade, portanto se entende como nulo. *Fechado?*

RESUMO

SEMELHANÇAS	Representam informações no formato texto com uma natureza auto-descritiva.
	Ambos são capazes de representar informações complexas difíceis de representar no formato tabular como relações de hierarquia, dados ausentes, entre outros.
	Ambos podem ser considerados padrões para representação de dados. XML é um padrão W3C, enquanto JSON foi formalizado na RFC 4627.
	Ambos são independentes de linguagem. Dados representados em XML e JSON podem ser acessados por qualquer linguagem de programação, através de API's específicas.

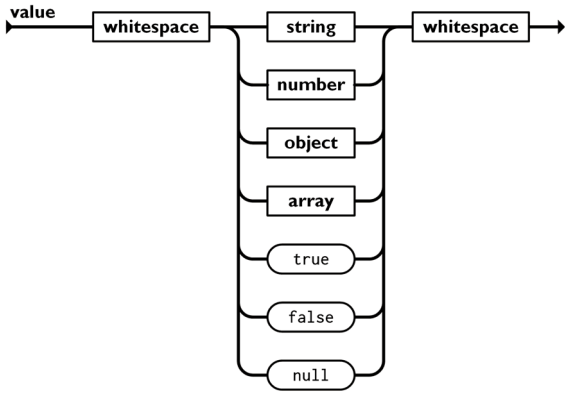
DIFERENÇAS	JSON não é uma linguagem de marcação – não possui tags de abertura e de fechamento.
	JSON representa as informações de forma mais compacta – basta visualizar os códigos acima.
	JSON não permite a execução de instruções de processamento, enquanto isso é possível em XML por meio do prolog.
	JSON é tipicamente destinado para a troca de informações, enquanto XML possui mais aplicações. Por exemplo: existem bancos de dados no formato XML e estruturados em SGBD XML nativo.

CÓDIGO EM XML

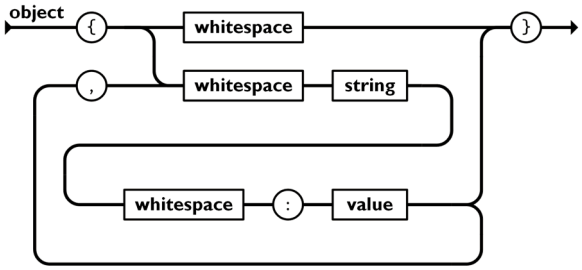
```xml
<servidores>
  <servidor>
    <primeiroNome>Diego</primeiroNome> <ultimoNome>Carvalho</ultimoNome>
  </servidor>
  <servidor>
    <primeiroNome>Renato</primeiroNome> <ultimoNome>da Costa</ultimoNome>
  </servidor>
  <servidor>
    <primeiroNome>Thiago</primeiroNome> <ultimoNome>Cavalcanti</ultimoNome>
  </servidor>
</servidores>
```

CÓDIGO EM JSON

```json
{"servidores":[
  { "primeiroNome":"Diego", "ultimoNome":"Carvalho" },
  { "primeiroNome ":"Renato", " ultimoNome":"da Costa" },
  { "primeiroNome ":"Thiago", " ultimoNome":"Cavalcanti" }
]}
```



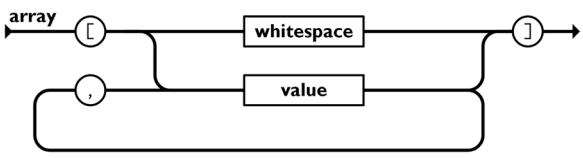
OBJETOS



Um objeto é um conjunto desordenado de pares nome/valor (também chamado de chave/valor). Conforme podemos ver no diagrama anterior, a sintaxe de um objeto começa com uma chave de abertura ({) e termina com uma chave de fechamento (}), sendo que cada nome é seguido por um sinal de dois-pontos (:) e os pares nome/valor são seguidos por um sinal de vírgula (,). Além disso, um nome ou chave é uma string (sequência de caracteres).

```
{
    "texto" : "Brasil",
    "numero" : 23,
    "numeroReal" : 54.87,
    "booleano": true,
    "nulo": null
}
```

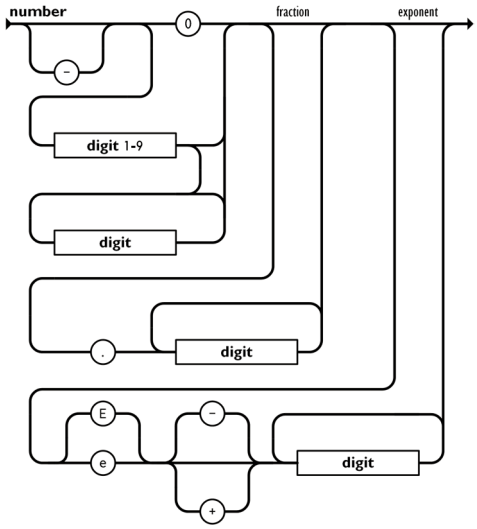
OBJETOS



Uma array (ou vetor) é uma coleção de zero ou mais valores ordenados! Ele começa com um colchete de abertura ([) e termina com um colchete de fechamento (]), sendo que os valores são separados por vírgula (,). No contexto de um array, o primeiro elemento é considerado como sendo de índice `0` e, não, `1`. No exemplo a seguir, temos um array com três valores: o primeiro tem o índice `0`, o segundo tem o índice `1` e o terceiro tem o índice `2`.

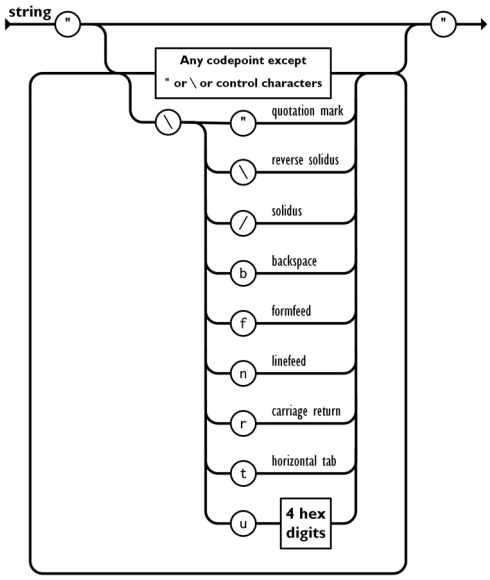
```
{
"servidores":["Diego", "Renato", "Thiago"]
}
```

NÚMEROS



Um número é uma sequência de dígitos em formato decimal que não diferencia números inteiros de ponto flutuante. Pode conter sinal de negativo, parte fracionária, expoentes/potências, etc.

STRING



Uma string é uma cadeia de zero ou mais caracteres Unicode – elas são delimitadas por aspas duplas (") e suportam a barra inversa (\) como caractere de escape.

QUESTÕES COMENTADAS – DIVERSAS BANCAS

1. **(CESPE / Ministério de Economia – 2020)** O JavaScript Object Notation (JSON) é um formato de intercâmbio de dados baseado em texto. De acordo com a gramática JSON, especificada na RFC 8259, é correto afirmar que:

   a) um array é representado como colchetes em torno de zero ou mais elementos.
   b) uma string começa e termina com crases.
   c) nomes literais devem ser utilizados em caixa alta.
   d) um número é representado na base 16 usando símbolos alfanuméricos.
   e) um objeto é representado como um par de parênteses em torno de zero ou mais pares nome/valor.

Comentários:

(a) Correto; (b) Errado, utilizam-se aspas; (c) Errado, não existe essa obrigação; (d) Errado, ele não suporta números em octal ou hexadecimal; (e) Errado, é representado como um par de chaves.

Gabarito: Letra A

---

2. **(CESPE / TRT8 - 2022)** Assinale a opção que apresenta a notação que representa corretamente em JSON a propriedade de `Nome` para as Pessoas `João` e `Maria`.

   a) `{"Pessoas": {"Nome":"João" }, {"Nome":"Maria"}}`
   b) `{"Pessoas": [{"Nome":"João" } {"Nome":"Maria"}]}`
   c) `{"Pessoas": ["Nome":"João", "Nome":"Maria"]}`
   d) `{"Pessoas" [{"Nome":"João"}, {"Nome":"Maria"}]}`
   e) `{"Pessoas": [{"Nome":"João"}, {"Nome":"Maria"}]}`

Comentários:

Em JSON, um objeto segue a sintaxe: {**propriedade:valor**}. A propriedade "Pessoas" tem como valor uma coleção (array) de outros objetos. Para representar um array, utilizamos os colchetes e os objetos internos também segue a sintaxe {propriedade:valor} separados por vírgula. Nesse caso, a propriedade é o "Nome" e os valores são "João" e "Maria". Logo, temos:

{"Pessoas": [{"Nome":"João"},{"Nome":"Maria"}]}

Gabarito: Letra E

---

3. **(CESGRANRIO / CEF – 2021)** A seguir, é apresentada parte de um arquivo em JSON, em que foram inseridos números seguidos de ponto (.) apenas para indicar as linhas.

```
1. [
2.  {
3.      "Fabrica": "FabricaB",
4.      "Pais": "PaisB",
5.      "Carros": [
6.         "CarroA",
7.         "CarroB"
8.      ]
9.  },
10.   {
11.      "Fabrica": "FabricaC",
12.      "Pais": "PaisC",
13.      "Carros": [
14.         "CarroC",
15.         "CarroD"
16.     ]
17.   }
18.]
```

Tendo como referência as informações precedentes, julgue os itens subsecutivos, a respeito de JSON.

A sintaxe das linhas 1, 5, 8, 13, 16 e 18 está errada, pois, em JSON, os vários valores em um array não devem ser listados entre colchetes, como ocorre nas linhas mencionadas, mas entre parênteses.

Comentários:

Pelo contrário, valores em um array devem ser listados entre colchetes.

Gabarito: Errado

---

4. **(CESPE / BANESE – 2021)** Em JSON cada objeto é representado por uma lista de nomes e valores apresentados entre chaves e agrupados por colchetes.

Comentários:

Objetos são representados por uma lista e nomes/valores entre chaves, mas não têm que vir entre colchetes – não entendi por que essa questão foi considerada correta pela banca.

Gabarito: Correto

---

5. **(CESGRANRIO / Banco da Amazônia – 2021)** Para transportar os dados de um sistema para outro, um programador recebeu a tarefa de transformar um arquivo CSV, gerado no primeiro sistema, para o formato JSON, suportado pelo segundo sistema. Nesse contexto, considere o seguinte arquivo CSV:

```
nome;saldo Ana Zurique;3000 Bernardo Washington;4500 Carlos York;12345
```

O fragmento de arquivo JSON válido que possui a mesma informação que o arquivo CSV apresentado acima é:

a)
```
[
 {
   nome: "Ana Zurique",
   saldo: 3000
 },
 {
   nome: "Bernardo Washington",
   saldo: 4500
 },
 {
   nome: "Carlos York",
   saldo: 12345
 }
]
```

b)
```
[
 {
   "nome": "Ana Zurique",
   "saldo": 3000
 },
 {
   "nome": "Bernardo Washington",
   "saldo": 4500
 },
 {
   "nome": "Carlos York",
   "saldo": 12345
 }
]
```

c)
```
{
   "Ana Zurique": {
      saldo: 3000
   },
   "Bernardo Washington": {
      saldo: 4500
   },
   "Carlos York": {
      saldo: 12345
   }
}
```

d)
```
 {
    nome:[Ana Zurique,Bernardo Washington,Carlos York],
    saldo:[3000,4500,12345]
 }
```

e)
```
[
  [Ana Zurique,3000  ],
  [Bernardo Washington,4500  ],
  [Carlos York,12345  ]
]
```

Comentários:

Há outros erros em cada item, mas bastava lembrar que o nome em um par nome/valor deve vir entre aspas. Dessa forma, podemos ver que todos os itens estão errados, exceto o segundo.

Gabarito: Letra B

6. **(IDIB / Prefeitura de Xinguara-PA – 2020)** JSON e XML são exemplos de dois padrões para estruturação e representação de dados. Ambos são amplamente utilizados na Internet em atividades que envolvem a integração entre sistemas. A respeito destes dois importantes padrões, analise as afirmativas abaixo.

I. Comparado ao XML, JSON possui a vantagem de apresentar a informação de forma mais compacta.
II. Enquanto JSON apresenta as informações em formato texto, em XML as informações são apresentadas em formato binário.
III. Ambos são capazes de representar relações de hierarquia. Já em relação a atributos multivalorados, apenas o padrão XML é capaz de suportar este tipo de representação.

É correto o que se afirma:

a) apenas em I.
b) apenas em II.
c) apenas em II e III.
d) em I, II e III.

Comentários:

(I) Correto; (II) Errado, ambos apresentam informações em formato texto; (III) Errado, XML não suporta atributos multivalorados.

Gabarito: Letra A

---

7. **(VUNESP / EBSERH - 2020)** Deseja-se representar, em formato JSON (JavaScript Object Notation), um objeto contendo notas de dois alunos. Uma representação correta de um possível objeto para este fim é:

a)
```
[{"nome": "José",
"notas": [5.5, 8, 7]},
{"nome": "Maria",
"notas": [8, 9, 7.5]}]
```

b)
```
{"Nomes": ["José", "Maria"]}, {
"Notas": [[5.5, 8, 7], [8, 9, 7.5]]}
```

c)
```
{"Alunos": [ {"nome": "José", "notas": [5.5, 8, 7]}, {"nome": "Maria", "notas": [8, 9, 7.5]}]}
```

d)
```
{"Alunos": ["José" = [5.5, 8, 7], "Maria" = [8, 9, 7.5]]}
```

e) {"Alunos" = [{"nome" = "José", "notas" = [5.5, 8, 7]}, {"nome" = "Maria", "notas" = [8, 9, 7.5]}]}

Comentários:

(a) Errado, objetos vêm entre chaves e, não, colchetes; (b) Errado, são dois objetos diferentes; (c) Correto; (d) Errado, a questão trocou chaves por colchetes e não se usa sinal de igualdade, usa-se dois-pontos; (e) Errado, não se usa sinal de igualdade, usa-se dois-pontos.

Gabarito: Letra C

---

8. **(CESPE / TJ-AM – 2019)** Um objeto representado em JSON é constituído por um nome e por um valor, de modo que cada ocorrência dessa dupla (nome/valor) está restrita a dados em formato de texto.

Comentários:

JSON aceita diferentes tipos de dados (string, números, booleanos, etc), mas todos eles possuem o formato de texto – não confundam!

Gabarito: Correto

---

9. **(FCC / Prefeitura de Manaus-AM – 2019)** Na sintaxe do padrão de troca de dados JSON, o tipo de dados que um campo pode conter é, dentre outros:

a) float.
b) vector.
c) date.
d) boolean.
e) function.

Comentários:

Dentre as opções, a única que contém um tipo de dados é o boolean.

Gabarito: Letra D

---

10. **(CCV / UFC – 2019)** Uma das formas de realizar a troca de informações entre dispositivos é através de mensagens com conteúdo JSON (Javascript Object Notation). Sobre o JSON, é correto afirmar:

a) Uma semelhança entre o XML e o JSON é a categorização das duas como sendo linguagens de marcação.

b) É uma notação textual baseada na especificação de um conjunto de elementos organizados na forma chave-valor.

c) Assim como o XML, o JSON não permite que seja especificado o tipo dos elementos que estão sendo especificados devido o seu formato textual.

d) É uma especificação onde os dados do arquivo são compilados e processados gerando o arquivo binário que será utilizado na troca de informações.

e) Uma da desvantagem no uso do JSON é a sua dependência da plataforma sobre a qual está executando, devido à necessidade do arquivo ser compilado para cada plataforma diferente.

Comentários:

(a) Errado, JSON não é uma linguagem de marcação; (b) Correto; (c) Errado, é permitido especificar o tipo em ambos; (d) Errado, nada faz sentido nesse item; (e) Errado, ele é completamente independente de plataforma.

Gabarito: Letra B

---

11. (CCV / UFC – 2019) Utilizando JSON, na notação abaixo, Equipe é de qual tipo JSON?

```
{ "Equipe": [ "Carlos", "Pedro", "Francisco" ] }
```

a) Um array.
b) Um object.
c) Um struct.
d) Uma classe.
e) Uma string JSON inválida.

Comentários:

Note que a sintaxe possui colchetes, logo se trata de um array.

Gabarito: Letra A

---

12. (CESPE / CGE-CE – 2019) As estruturas básicas que constituem um JSON são:

a) uma lista de valores e uma coleção de strings/inteiros.
b) uma lista de inteiros e uma coleção de pares de strings.
c) uma lista de valores ordenados e uma coleção de pares nome/valor.
d) uma lista de strings e uma coleção de pares de booleanos.
e) uma lista de booleanos e uma coleção de listas de strings.

Comentários:

As estruturas básicas que constituem um JSON são uma lista de valores e uma coleção de pares nome/valor. Não é necessariamente uma lista de valores ordenados, mas é o item menos errado.

Gabarito: Letra C

---

13. (UECE / Prefeitura de Sobral-CE – 2019) Assinale a opção que traz um exemplo correto de dados escritos no formato JSON (JavaScript Object Notation):

a)

```
{ "pessoa" :
    [ { "nome" : "Claudia Rosa",
      "salario" : "40",
      "aluno" : false },
     { "nome" : "Silva Neto",
      "salario" : "33",
      "aluno" : true } ]
     }
```

b)

```
pessoas
    pessoa
      nome = "Claudia Rosa"
      salario = "40"
      aluno = FALSE
    pessoa
      nome = "Silva Neto"
      salario = 33
      aluno = TRUE
```

c)

```
<JSON id="Pessoa">
    <TH><TD>Nome</TD>
      <TD>salario</TD>
      <TD>aluno</TD></TH>
    <TR><TD>Claudia Rosa</TD>
      <TD>40</TD>
        <TD>false</TD></TR>
    <TR><TD>Silva Neto</TD>
        <TD>3340</TD>
        <TD>true</TD></TR>
     </JSON>
```

d)

```
<Pessoas>
    <Pessoa>
      <nome>Claudia Rosa</nome>
      <salario>40</salario>
      <aluno>false</aluno></Pessoa>
    <Pessoa>
      <nome>Silva Neto</nome>
      <salario>3340</salario>
      <aluno>true</aluno></Pessoa>
     </Pessoas>
```

Comentários:

(a) Correto; (b) Errado, está tudo errado – o código está simplesmente indentado; (c) Errado, JSON não possui tags; (d) Errado, JSON não possui tags.

Gabarito: Letra A

---

14. (FUNRIO / Câmara de São João de Mariti-RJ – 2018) Avalie se as seguintes afirmativas, relativas ao JSON, são falsas (F) ou verdadeiras (V):

( ) JSON é uma linguagem de marcação, possuindo tag de abertura e de fechamento.
( ) JSON é um modelo para armazenamento e transmissão de informações no formato texto.
( ) JSON é independente de linguagem, sendo possível o acesso aos dados por qualquer linguagem de programação, por meio de API's específicas.

As afirmativas são respectivamente:

a) V, F e F.
b) V, V e F.
c) F, V e V.
d) F, F e V.
e) F, V e F.

Comentários:

(F) Errado, não é uma linguagem de marcação; (V) Correto; (V) Correto.

Gabarito: Letra C

---

15. (QUADRIX / CRM - PR - 2018) JSON (JavaScript Object Notation) é um formato de arquivo de padrão aberto para uso exclusivo em programação JavaScript.

Comentários:

Não é de uso exclusivo em programação JavaScript – é independente de linguagem de programação.

Gabarito: Errado

---

16. (CESGRANRIO / LIQUIGÁS – 2018) Interoperabilidade refere-se ao processo de comunicação entre sistemas sem que seja criada uma dependência tecnológica entre eles. Uma maneira de implementar a interoperabilidade é pela transferência de dados pelo uso de uma linguagem de intercâmbio de dados como:

a) JSON.
b) GTFS.

c) SPARQL.
d) XML SCHEMA.
e) XSLT.

Comentários:

Uma maneira de implementar a interoperabilidade é pela transferência de dados pelo uso de uma linguagem de intercâmbio de dados como JSON.

Gabarito: Letra A

---

17. (UECE / FUNCEME – 2018) Atente ao que se diz a seguir sobre o JSON (JavaScript Object Notation) e assinale a afirmação verdadeira.

a) É uma notação em formato texto criada para programas web desenvolvidos em JavaScript.
b) Apesar de ser fácil de ler e escrever para humanos, é difícil de gerar e interpretar para máquinas.
c) Diferentemente de XML, não pode ser utilizado como formato de representação de dados em web services.
d) Sua estrutura é constituída por um conjunto de pares chave/valor, o que o torna um formato propício para troca de dados.

Comentários:

(a) Errado, surgiu a partir do JavaScript, mas é independente de linguagem de programação; (b) Errado, é fácil de ler escrever para ambos; (c) Errado, não só pode como é utilizada como formato de representação de dados em serviços web; (d) Correto.

Gabarito: Letra D

---

18. (IF-RS / IF-RS – 2018) JSON (JavaScript Object Notation) é um formato para intercâmbio de dados baseado em texto e independente de linguagem. A respeito das características que apresenta este formato de acordo com o padrão ECMA-404, classifique cada uma das afirmativas abaixo como verdadeira (V) ou falsa (F), e assinale a alternativa que apresenta a sequência CORRETA, de cima para baixo:

( ) Suporta os seguintes tipos de dados: object, array, number, string, true, false e null.
( ) É indicado para aplicações que requerem transferência de dados binários.
( ) Não pode ser considerado uma especificação de intercâmbio completo de dados.
( ) Sua sintaxe é formada por chaves, colchetes, dois pontos, vírgulas e cifrões.

a) V – F – F – V.
b) F – V – F – F.

c) V – F – V – V.
d) V – V – V – V.
e) V – F – V – F

Comentários:

(V) Correto; (F) Errado, é indicado para aplicações que requerem transferência de dados textuais; (V) Correto. De acordo com a especificação, a sintaxe não é uma especificação de um intercâmbio completo de dados. O intercâmbio de dados significativo requer acordo entre um produtor e um consumidor sobre a semântica anexada a um uso específico da sintaxe JSON – o que JSON fornece é a estrutura sintática à qual essa semântica pode ser anexada; (F) Errado, não há cifrões.

Gabarito: Letra E

---

19. (SUGEP / UFRPE – 2018) Sobre JSON, analise as afirmativas abaixo:

1) É um formato de arquivo de texto para troca de dados em que um objeto é um conjunto de pares nome/valor.
2) A tecnologia JSON é uma especialização do formato XML para representação de dados.
3) Para lidar com um conjunto de objetos em JSON, é necessário utilizar um array que permita realizar, em uma única operação, a carga de todos os objetos.

- Está(ão) correta(s), apenas:

a) 1.
b) 2.
c) 1 e 2.
d) 1 e 3.
e) 3.

Comentários:

(1) Correto; (2) Errado, não se trata de uma especialização do XML – são representações diferentes; (3) Errado, não é obrigatório utilizar um array – há diversas outras maneiras de realizar cargas.

Gabarito: Letra A

---

20. (CESGRANRIO / BANCO DO BRASIL - 2018) O seguinte item apresenta uma estrutura em formato JSON válido:

```
{
    nomeCompleto: {
            nome: Fulano,
            sobrenome: de Tal},
    idade: 26,
    email: {
            fulano@aqui.com.br,
            fulano@ali.com.br,
            fulano@la.com.br
    },
    endereco: {},
    preferencias: null
}
```

a)

```
{
    nomeCompleto: {
            nome: Fulano,
            sobrenome: de Tal},
    idade: 26,
    email: [
            fulano@aqui.com.br,
            fulano@ali.com.br,
            fulano@la.com.br
    ],
    endereco: {},
    preferencias: null
}
```

b)

```
{
    "nomeCompleto": {
            "nome": "Fulano",
            "sobrenome": "de Tal"},
    "idade": 26,
    "email": [
            "fulano@aqui.com.br",
            "fulano@ali.com.br",
            "fulano@la.com.br"
    ],
    "endereco": {},
    "preferencias": null
}
```

c)

```
{
    "nomeCompleto": {
            "nome": "Fulano",
            "sobrenome": "de Tal"},
    "idade": 26,
    "email": {
            "fulano@aqui.com.br",
            "fulano@ali.com.br",
            "fulano@la.com.br"
    },
    "endereco": {},
    "preferencias": null
}
```

d)

e)
```
{
    "nomeCompleto": {
            "nome": Fulano,
            "sobrenome": de Tal},
    "idade": 26,
    "email": {
            fulano@aqui.com.br,
            fulano@ali.com.br,
            fulano@la.com.br
    },
    "endereco": {},
    "preferencias": null
}
```

Comentários:

(a) Errado, faltas as aspas nos nomes; (b) Errado, faltas as aspas nos nomes; (c) Correto; (d) Errado, e-mail é um array, logo deveria vir entre colchetes; (e) Errado, faltam as aspas nas strings.

Gabarito: Letra C

---

21. (CESPE / STM – 2018) O protocolo JSON é derivado da linguagem de programação Java e sua utilização é restrita a sistemas desenvolvidos em Java ou JavaScript.

Comentários:

JSON não é um protocolo; é derivado da linguagem de programação JavaScript e, não, Java; e sua utilização não é restrita a sistemas desenvolvidos nessas linguagens – ele é independente de linguagens de programação.

Gabarito: Errado

---

22. (COMPERVE / UFRN – 2016) A formatação JSON (JavaScript Object Notation) é:

a) uma linguagem de marcação, pois possui tags de abertura e de fechamento.
b) um modelo independente de linguagem, pois usa convenções que são familiares às linguagens C, C++, C#, Java, JavaScript, Perl, Python e muitas outras.
c) um modelo de banco de dados relacional robusto para armazenamento e transmissão de informações.
d) uma linguagem que permite a execução de instruções de processamento.

Comentários:

(a) Errado, não se trata de uma linguagem de marcação; (b) Correto; (c) Errado, não se trata de um modelo de banco de dados relacional; (d) Errado, ela não permite a execução de instruções de processamento.

Gabarito: Letra B

23. (VUNESP / Prefeitura de Presidente Prudente-SP – 2016) Segundo a especificação ECMA-404 que descreve o formato JSON, é válido o documento:

a)
```
{
  'id': '1',
  'nome': 'João',
  'irmão': 'Mário',
  'irmã': 'Joana'
}
```

b)
```
{
  "id": 1,
  "nome": "João",
  "irmãos": [
    "Mário": 2,
    "Pedro": 3
  ],
  "irmãs": [
    "Joana": 4,
    "Rose": 5
  ]
}
```

c)
```
{
  'id': 1,
  'nome': 'João',
  'irmãos': [
    'Mário',
    'Pedro'
  ],
  'irmãs': [
    'Joana',
    'Rose'
  ]
}
```

d)
```
{
  "id": 1,
  "nome": "João",
  "irmãos": [
    "Mário",
    "Pedro"
  ],
  "irmãs": [
    "Joana",
    "Rose"
  ]
}
```

e)
```
{
  "id": 1,
  "nome": "João",
  "irmãos": {
    "Mário",
    "Pedro"
  },
  "irmãs": {
    "Joana",
    "Rose"
  }
}
```

Comentários:

(a) Errado, utilizam-se aspas duplas e, não, simples; (b) Errado, o correto seria utilizar `"irmãos": {"Mário":2,"Pedro":3}`; (c) Errado, utilizam-se aspas duplas e, não, simples; (d) Correto; (e) Errado, o correto seria `"irmãos":["Mário","Pedro"]`.

Gabarito: Letra D

---

24. (CCV / UFC - 2016) Considerando a estrutura JSON (JavaScript Object Notation), marque o item correto.

a) Segundo a especificação do JSON, não é possível a utilização de caracteres de escape, sendo uma limitação na troca de dados.

b) Por sua notação simplista, somente é possível utilizar tipos de dados primitivos: caractere, números e valores lógicos (Verdadeiro ou Falso).

c) Na especificação dos dados no formato JSON, os valores são especificados em pares (chave:valor), onde a chave é identificada por números sequenciais.

d) A partir da troca de arquivos com dados no formato JSON, aplicações escritas em linguagens diferentes podem trocar informações de forma estruturada e simples.

e) Como o JSON é uma notação recente, ela foi especificada baseada nas linguagens mais novas, sendo compatíveis somente com as linguagens originadas a partir da linguagem C.

Comentários:

(a) Errado, é possível – sim – utilizar caracteres de escape; (b) Errado, há também objetos, arrays e null; (c) Errado, não há necessidade de ser números sequenciais; (d) Correto; (e) Errado, ela é independente de linguagens, sendo compatível com dezenas delas.

Gabarito: Letra D

---

25. (CESPE / TCE-PA – 2016) Comparativamente ao XML, o parsing de informações em JSON é mais rápido devido ao fato de ser capaz de executar instruções de processamento.

Comentários:

Parsing é o processo de análise sintática de um código! O parsing do JSON é realmente mais rápido, mas o JSON não é capaz de executar instruções de processamento. Ele é mais rápido porque o formato do JSON é mais simples, compacto e leve em comparação com o XML. Por fim, o XML – sim – é capaz de executar instruções de processamento.

Gabarito: Errado

---

26. (CESPE / TCE-PA – 2016) O formato JSON (JavaScript Object Notation) é uma especialização do XML e pode ser utilizado para representar dados.

Comentários:

Realmente pode ser utilizado para representar dados, mas não se trata de uma especialização do XML. Não há nenhuma relação entre ambos os formatos!

Gabarito: Errado

---

27. (BIO-RIO / IF-RJ – 2015) No que diz respeito aos padrões XML e JSON, analise as afirmativas a seguir.

I - JSON representa informações no formato texto, da mesma forma que XML.
II - JSON não é uma linguagem de marcação, da mesma forma que XML também não é.
III - JSON não permite a execução de instruções de processamento, algo possível em XML.

Assinale a alternativa correta:

a) somente a afirmativa I está correta.
b) somente a afirmativa II está correta.
c) somente a afirmativa II está correta.
d) somente as afirmativas I e III estão corretas.
e) todas as afirmativas estão corretas.

Comentários:

(I) Correto; (II) Errado, XML é uma linguagem de marcação, dado que não possui tags de abertura e fechamento; (III) Correto, realmente não é permitido em JSON, mas é permitido em XML.

Gabarito: Letra D

---

28. (CESPE / TRE-MT – 2015) Assinale a opção que apresenta corretamente um objetivo seguido de uma representação em JSON (JavaScript Object Notation).

a) Representar o ano de 2015:
```
ano:= [2015].
```

b) Representar a cotação do dólar:
```
"dolar": 3.87.
```

c) Representar a projeção do PIB brasileiro negativo:
```
{PIB-BR}= -3[%].
```

d) Representar valor booleano:
`{recessão}:= [true].`

e) Representar array de strings:
`{"[DF],[MT],[MS],[AM]"}.`

Comentários:

(a) Errado, o correto seria `"ano":2015`; (b) Correto; (c) Errado, o correto seria `"PIB-BR":"-3%"`; (d) Errado, o correto seria `"recessão":true`; (e) Errado, o correto seria `["DF","MT","MS","AM"]`.

Gabarito: Letra B

---

29. (CESPE / STJ – 2015) JSON (JavaScript Object Notation) é um formato de arquivo de texto para troca de dados em que um objeto é um conjunto de pares nome/valor.

Comentários:

É um formato de arquivo de texto? Sim! *É utilizado para troca de dados?* Sim! *Um objeto é um conjunto de pares de nome/valor?* Sim!

Gabarito: Correto

---

30. (CCV / UFC – 2013) Um Analista de TI, ao analisar um determinado sistema WEB, observa o arquivo a seguir.

```
{
"id":1,
"nome":"José Patriolino Silveira",
"endereco":"Campus do Pici, Bloco 02 A"
}
```

Podemos afirmar que este arquivo está no formato:

a) XML.
b) JSON.
c) DOM.
d) XSLT.
e) SOAP.

Comentários:

Observem que se trata de uma sintaxe de nome/valor, logo se trata de um Arquivo JSON. XML possui tags de abertura e fechamento; SOAP, XSLT e DOM são baseados em XML.

Gabarito: Letra B

31. (CESGRANRIO / BNDES – 2011) Um programador, ao analisar determinado sistema WB, observa o arquivo a seguir.

```
{
  "menu": "m1",
  "acoes": [
       {
            "titulo": "X",
            "desc":"A"
       },
       {
            "titulo": "Y",
            "desc": "B"
       },
       {
            "titulo": "Z",
            "desc": "C"
       }
   ]
}
```

Qual o formato desse arquivo:

a) JSON.
b) REST.
c) SOAP.
d) XSLT.
e) XML.

Comentários:

Observem que se trata de uma sintaxe de nome/valor, logo se trata de um Arquivo JSON. XML possui tags de abertura e fechamento; SOAP e XSLT são baseados em XML; e REST é um estilo arquitetural, logo não se trata de um formato.

Gabarito: Letra A

LISTA DE QUESTÕES – DIVERSAS BANCAS

1. **(CESPE / Ministério de Economia – 2020)** O JavaScript Object Notation (JSON) é um formato de intercâmbio de dados baseado em texto. De acordo com a gramática JSON, especificada na RFC 8259, é correto afirmar que:

   a) um array é representado como colchetes em torno de zero ou mais elementos.
   b) uma string começa e termina com crases.
   c) nomes literais devem ser utilizados em caixa alta.
   d) um número é representado na base 16 usando símbolos alfanuméricos.
   e) um objeto é representado como um par de parênteses em torno de zero ou mais pares nome/valor.

2. **(CESPE / TRT8 - 2022)** Assinale a opção que apresenta a notação que representa corretamente em JSON a propriedade de `Nome` para as Pessoas `João` e `Maria`.

   a) `{"Pessoas": {"Nome":"João" }, {"Nome":"Maria"}}`
   b) `{"Pessoas": [{"Nome":"João" } {"Nome":"Maria"}]}`
   c) `{"Pessoas": ["Nome":"João", "Nome":"Maria"]}`
   d) `{"Pessoas" [{"Nome":"João"}, {"Nome":"Maria"}]}`
   e) `{"Pessoas": [{"Nome":"João"}, {"Nome":"Maria"}]}`

3. **(CESGRANRIO / CEF – 2021)** A seguir, é apresentada parte de um arquivo em JSON, em que foram inseridos números seguidos de ponto (.) apenas para indicar as linhas.

```
1. [
2.  {
3.      "Fabrica": "FabricaB",
4.      "Pais": "PaisB",
5.      "Carros": [
6.         "CarroA",
7.         "CarroB"
8.      ]
9.  },
10.  {
11.        "Fabrica": "FabricaC",
12.        "Pais": "PaisC",
13.        "Carros": [
14.           "CarroC",
15.           "CarroD"
16.     ]
17.  }
18.]
```

   Tendo como referência as informações precedentes, julgue os itens subsecutivos, a respeito de JSON.

   A sintaxe das linhas 1, 5, 8, 13, 16 e 18 está errada, pois, em JSON, os vários valores em um array não devem ser listados entre colchetes, como ocorre nas linhas mencionadas, mas entre parênteses.

4. **(CESPE / BANESE – 2021)** Em JSON cada objeto é representado por uma lista de nomes e valores apresentados entre chaves e agrupados por colchetes.

5. **(CESGRANRIO / Banco da Amazônia – 2021)** Para transportar os dados de um sistema para outro, um programador recebeu a tarefa de transformar um arquivo CSV, gerado no primeiro sistema, para o formato JSON, suportado pelo segundo sistema. Nesse contexto, considere o seguinte arquivo CSV:

```
nome;saldo Ana Zurique;3000 Bernardo Washington;4500 Carlos York;12345
```

O fragmento de arquivo JSON válido que possui a mesma informação que o arquivo CSV apresentado acima é:

a)
```
[
 {
   nome: "Ana Zurique",
   saldo: 3000
 },
 {
   nome: "Bernardo Washington",
   saldo: 4500
 },
 {
   nome: "Carlos York",
   saldo: 12345
 }
]
```

b)
```
[
 {
   "nome": "Ana Zurique",
   "saldo": 3000
 },
 {
   "nome": "Bernardo Washington",
   "saldo": 4500
 },
 {
   "nome": "Carlos York",
   "saldo": 12345
 }
]
```

c)
```
{
   "Ana Zurique": {
      saldo: 3000
   },
   "Bernardo Washington": {
      saldo: 4500
   },
   "Carlos York": {
      saldo: 12345
   }
}
```

d)
```
{
   nome:[Ana Zurique,Bernardo Washington,Carlos York],
   saldo:[3000,4500,12345]
}
```

e)
```
[
  [Ana Zurique,3000  ],
  [Bernardo Washington,4500  ],
  [Carlos York,12345  ]
]
```

6. **(IDIB / Prefeitura de Xinguara-PA – 2020)** JSON e XML são exemplos de dois padrões para estruturação e representação de dados. Ambos são amplamente utilizados na Internet em atividades que envolvem a integração entre sistemas. A respeito destes dois importantes padrões, analise as afirmativas abaixo.

I. Comparado ao XML, JSON possui a vantagem de apresentar a informação de forma mais compacta.
II. Enquanto JSON apresenta as informações em formato texto, em XML as informações são apresentadas em formato binário.
III. Ambos são capazes de representar relações de hierarquia. Já em relação a atributos multivalorados, apenas o padrão XML é capaz de suportar este tipo de representação.

É correto o que se afirma:

a) apenas em I.
b) apenas em II.
c) apenas em II e III.
d) em I, II e III.

7. **(VUNESP / EBSERH - 2020)** Deseja-se representar, em formato JSON (JavaScript Object Notation), um objeto contendo notas de dois alunos. Uma representação correta de um possível objeto para este fim é:

a)
```
[{"nome": "José",
"notas": [5.5, 8, 7]},
{"nome": "Maria",
"notas": [8, 9, 7.5]}]
```

b)
```
{"Nomes": ["José", "Maria"]}, {
"Notas": [[5.5, 8, 7], [8, 9, 7.5]]}
```

c)
```
{"Alunos": [ {"nome": "José", "notas": [5.5, 8, 7]}, {"nome": "Maria", "notas": [8, 9, 7.5]}]}
```

d)
```
{"Alunos": ["José" = [5.5, 8, 7], "Maria" = [8, 9, 7.5]]}
```

e) {"Alunos" = [{"nome" = "José", "notas" = [5.5, 8, 7]}, {"nome" = "Maria", "notas" = [8, 9, 7.5]}]}

8. **(CESPE / TJ-AM – 2019)** Um objeto representado em JSON é constituído por um nome e por um valor, de modo que cada ocorrência dessa dupla (nome/valor) está restrita a dados em formato de texto.

9. **(FCC / Prefeitura de Manaus-AM – 2019)** Na sintaxe do padrão de troca de dados JSON, o tipo de dados que um campo pode conter é, dentre outros:

a) float.
b) vector.
c) date.
d) boolean.
e) function.

10. **(CCV / UFC – 2019)** Uma das formas de realizar a troca de informações entre dispositivos é através de mensagens com conteúdo JSON (Javascript Object Notation). Sobre o JSON, é correto afirmar:

a) Uma semelhança entre o XML e o JSON é a categorização das duas como sendo linguagens de marcação.

b) É uma notação textual baseada na especificação de um conjunto de elementos organizados na forma chave-valor.

c) Assim como o XML, o JSON não permite que seja especificado o tipo dos elementos que estão sendo especificados devido o seu formato textual.

d) É uma especificação onde os dados do arquivo são compilados e processados gerando o arquivo binário que será utilizado na troca de informações.

e) Uma da desvantagem no uso do JSON é a sua dependência da plataforma sobre a qual está executando, devido à necessidade do arquivo ser compilado para cada plataforma diferente.

11. **(CCV / UFC – 2019)** Utilizando JSON, na notação abaixo, Equipe é de qual tipo JSON?

{ "Equipe": ["Carlos", "Pedro", "Francisco"] }

a) Um array.
b) Um object.
c) Um struct.
d) Uma classe.
e) Uma string JSON inválida.

12. (CESPE / CGE-CE – 2019) As estruturas básicas que constituem um JSON são:

a) uma lista de valores e uma coleção de strings/inteiros.
b) uma lista de inteiros e uma coleção de pares de strings.
c) uma lista de valores ordenados e uma coleção de pares nome/valor.
d) uma lista de strings e uma coleção de pares de booleanos.
e) uma lista de booleanos e uma coleção de listas de strings.

13. (UECE / Prefeitura de Sobral-CE – 2019) Assinale a opção que traz um exemplo correto de dados escritos no formato JSON (JavaScript Object Notation):

a)

```
{ "pessoa" :
    [ { "nome" : "Claudia Rosa",
      "salario" : "40",
      "aluno" : false },
     { "nome" : "Silva Neto",
      "salario" : "33",
      "aluno" : true } ]
     }
```

b)

```
pessoas
    pessoa
      nome = "Claudia Rosa"
      salario = "40"
      aluno = FALSE
    pessoa
      nome = "Silva Neto"
      salario = 33
      aluno = TRUE
```

c)

```
<JSON id="Pessoa">
    <TH><TD>Nome</TD>
      <TD>salario</TD>
      <TD>aluno</TD></TH>
    <TR><TD>Claudia Rosa</TD>
      <TD>40</TD>
        <TD>false</TD></TR>
    <TR><TD>Silva Neto</TD>
        <TD>3340</TD>
        <TD>true</TD></TR>
     </JSON>
```

d)

```
<Pessoas>
    <Pessoa>
      <nome>Claudia Rosa</nome>
      <salario>40</salario>
      <aluno>false</aluno></Pessoa>
    <Pessoa>
      <nome>Silva Neto</nome>
      <salario>3340</salario>
      <aluno>true</aluno></Pessoa>
     </Pessoas>
```

14. (FUNRIO / Câmara de São João de Mariti-RJ – 2018) Avalie se as seguintes afirmativas, relativas ao JSON, são falsas (F) ou verdadeiras (V):

( ) JSON é uma linguagem de marcação, possuindo tag de abertura e de fechamento.
( ) JSON é um modelo para armazenamento e transmissão de informações no formato texto.
( ) JSON é independente de linguagem, sendo possível o acesso aos dados por qualquer linguagem de programação, por meio de API's específicas.

As afirmativas são respectivamente:

a) V, F e F.
b) V, V e F.
c) F, V e V.
d) F, F e V.
e) F, V e F.

15. **(QUADRIX / CRM - PR - 2018)** JSON (JavaScript Object Notation) é um formato de arquivo de padrão aberto para uso exclusivo em programação JavaScript.

16. **(CESGRANRIO / LIQUIGÁS – 2018)** Interoperabilidade refere-se ao processo de comunicação entre sistemas sem que seja criada uma dependência tecnológica entre eles. Uma maneira de implementar a interoperabilidade é pela transferência de dados pelo uso de uma linguagem de intercâmbio de dados como:

a) JSON.
b) GTFS.
c) SPARQL.
d) XML SCHEMA.
e) XSLT.

17. **(UECE / FUNCEME – 2018)** Atente ao que se diz a seguir sobre o JSON (JavaScript Object Notation) e assinale a afirmação verdadeira.

a) É uma notação em formato texto criada para programas web desenvolvidos em JavaScript.
b) Apesar de ser fácil de ler e escrever para humanos, é difícil de gerar e interpretar para máquinas.
c) Diferentemente de XML, não pode ser utilizado como formato de representação de dados em web services.
d) Sua estrutura é constituída por um conjunto de pares chave/valor, o que o torna um formato propício para troca de dados.

18. **(IF-RS / IF-RS – 2018)** JSON (JavaScript Object Notation) é um formato para intercâmbio de dados baseado em texto e independente de linguagem. A respeito das características que apresenta este formato de acordo com o padrão ECMA-404, classifique cada uma das afirmativas abaixo como verdadeira (V) ou falsa (F), e assinale a alternativa que apresenta a sequência CORRETA, de cima para baixo:

( ) Suporta os seguintes tipos de dados: object, array, number, string, true, false e null.
( ) É indicado para aplicações que requerem transferência de dados binários.
( ) Não pode ser considerado uma especificação de intercâmbio completo de dados.
( ) Sua sintaxe é formada por chaves, colchetes, dois pontos, vírgulas e cifrões.

a) V – F – F – V.
b) F – V – F – F.
c) V – F – V – V.
d) V – V – V – V.
e) V – F – V – F

19. **(SUGEP / UFRPE – 2018)** Sobre JSON, analise as afirmativas abaixo:

1) É um formato de arquivo de texto para troca de dados em que um objeto é um conjunto de pares nome/valor.
2) A tecnologia JSON é uma especialização do formato XML para representação de dados.
3) Para lidar com um conjunto de objetos em JSON, é necessário utilizar um array que permita realizar, em uma única operação, a carga de todos os objetos.

Está(ão) correta(s), apenas:

a) 1.
b) 2.
c) 1 e 2.
d) 1 e 3.
e) 3.

20. **(CESGRANRIO / BANCO DO BRASIL - 2018)** O seguinte item apresenta uma estrutura em formato JSON válido:

```
{
    nomeCompleto: {
            nome: Fulano,
            sobrenome: de Tal},
    idade: 26,
    email: {
            fulano@aqui.com.br,
            fulano@ali.com.br,
            fulano@la.com.br
    },
    endereco: {},
    preferencias: null
}
```

a)

```
{
    nomeCompleto: {
            nome: Fulano,
            sobrenome: de Tal},
    idade: 26,
    email: [
            fulano@aqui.com.br,
            fulano@ali.com.br,
            fulano@la.com.br
    ],
    endereco: {},
    preferencias: null
}
```

b)

```
{
    "nomeCompleto": {
            "nome": "Fulano",
            "sobrenome": "de Tal"},
    "idade": 26,
    "email": [
            "fulano@aqui.com.br",
            "fulano@ali.com.br",
            "fulano@la.com.br"
    ],
    "endereco": {},
    "preferencias": null
}
```

c)

```
{
    "nomeCompleto": {
            "nome": "Fulano",
            "sobrenome": "de Tal"},
    "idade": 26,
    "email": {
            "fulano@aqui.com.br",
            "fulano@ali.com.br",
            "fulano@la.com.br"
    },
    "endereco": {},
    "preferencias": null
}
```

d)

```
{
    "nomeCompleto": {
            "nome": Fulano,
            "sobrenome": de Tal},
    "idade": 26,
    "email": {
            fulano@aqui.com.br,
            fulano@ali.com.br,
            fulano@la.com.br
    },
    "endereco": {},
    "preferencias": null
}
```

e)

21. **(CESPE / STM – 2018)** O protocolo JSON é derivado da linguagem de programação Java e sua utilização é restrita a sistemas desenvolvidos em Java ou JavaScript.

22. **(COMPERVE / UFRN – 2016)** A formatação JSON (JavaScript Object Notation) é:

a) uma linguagem de marcação, pois possui tags de abertura e de fechamento.
b) um modelo independente de linguagem, pois usa convenções que são familiares às linguagens C, C++, C#, Java, JavaScript, Perl, Python e muitas outras.
c) um modelo de banco de dados relacional robusto para armazenamento e transmissão de informações.
d) uma linguagem que permite a execução de instruções de processamento.

23. **(VUNESP / Prefeitura de Presidente Prudente-SP – 2016)** Segundo a especificação ECMA-404 que descreve o formato JSON, é válido o documento:

a)
```
{
  'id': '1',
  'nome': 'João',
  'irmão': 'Mário',
  'irmã': 'Joana'
}
```

b)
```
{
  "id": 1,
  "nome": "João",
  "irmãos": [
    "Mário": 2,
    "Pedro": 3
  ],
  "irmãs": [
    "Joana": 4,
    "Rose": 5
  ]
}
```

c)
```
{
  'id': 1,
  'nome': 'João',
  'irmãos': [
    'Mário',
    'Pedro'
  ],
  'irmãs': [
    'Joana',
    'Rose'
  ]
}
```

```
{
  "id": 1,
  "nome": "João",
  "irmãos": [
    "Mário",
    "Pedro"
  ],
  "irmãs": [
    "Joana",
    "Rose"
  ]
}
```

d)

```
{
  "id": 1,
  "nome": "João",
  "irmãos": {
    "Mário",
    "Pedro"
  },
  "irmãs": {
    "Joana",
    "Rose"
  }
}
```

e)

24. (CCV / UFC - 2016) Considerando a estrutura JSON (JavaScript Object Notation), marque o item correto.

a) Segundo a especificação do JSON, não é possível a utilização de caracteres de escape, sendo uma limitação na troca de dados.

b) Por sua notação simplista, somente é possível utilizar tipos de dados primitivos: caractere, números e valores lógicos (Verdadeiro ou Falso).

c) Na especificação dos dados no formato JSON, os valores são especificados em pares (chave:valor), onde a chave é identificada por números sequenciais.

d) A partir da troca de arquivos com dados no formato JSON, aplicações escritas em linguagens diferentes podem trocar informações de forma estruturada e simples.

e) Como o JSON é uma notação recente, ela foi especificada baseada nas linguagens mais novas, sendo compatíveis somente com as linguagens originadas a partir da linguagem C.

25. (CESPE / TCE-PA – 2016) Comparativamente ao XML, o parsing de informações em JSON é mais rápido devido ao fato de ser capaz de executar instruções de processamento.

26. (CESPE / TCE-PA – 2016) O formato JSON (JavaScript Object Notation) é uma especialização do XML e pode ser utilizado para representar dados.

27. (BIO-RIO / IF-RJ – 2015) No que diz respeito aos padrões XML e JSON, analise as afirmativas a seguir.

I - JSON representa informações no formato texto, da mesma forma que XML.
II - JSON não é uma linguagem de marcação, da mesma forma que XML também não é.
III - JSON não permite a execução de instruções de processamento, algo possível em XML.

Assinale a alternativa correta:

a) somente a afirmativa I está correta.
b) somente a afirmativa II está correta.
c) somente a afirmativa II está correta.
d) somente as afirmativas I e III estão corretas.
e) todas as afirmativas estão corretas.

28. (CESPE / TRE-MT – 2015) Assinale a opção que apresenta corretamente um objetivo seguido de uma representação em JSON (JavaScript Object Notation).

a) Representar o ano de 2015:
`ano:= [2015].`

b) Representar a cotação do dólar:
`"dolar": 3.87.`

c) Representar a projeção do PIB brasileiro negativo:
`{PIB-BR}= -3[%].`

d) Representar valor booleano:
`{recessão}:= [true].`

e) Representar array de strings:
`{“[DF],[MT],[MS],[AM]”}.`

29. (CESPE / STJ – 2015) JSON (JavaScript Object Notation) é um formato de arquivo de texto para troca de dados em que um objeto é um conjunto de pares nome/valor.

30. (CCV / UFC – 2013) Um Analista de TI, ao analisar um determinado sistema WEB, observa o arquivo a seguir.

```
{
"id":1,
"nome":"José Patriolino Silveira",
"endereco":"Campus do Pici, Bloco 02 A"
}
```

Podemos afirmar que este arquivo está no formato:

a) XML.
b) JSON.

c) DOM.
d) XSLT.
e) SOAP.

31. **(CESGRANRIO / BNDES – 2011)** Um programador, ao analisar determinado sistema WB, observa o arquivo a seguir.

```
{
  "menu": "m1",
  "acoes": [
      {
          "titulo": "X",
          "desc":"A"
      },
      {
          "titulo": "Y",
          "desc": "B"
      },
      {
          "titulo": "Z",
          "desc": "C"
      }
   ]
}
```

Qual o formato desse arquivo:

a) JSON.
b) REST.
c) SOAP.
d) XSLT.
e) XML.

GABARITO – DIVERSAS BANCAS

1. LETRA A
2. LETRA E
3. ERRADO
4. CORRETO
5. LETRA B
6. LETRA A
7. LETRA C
8. CORRETO
9. LETRA D
10. LETRA B
11. LETRA A
12. LETRA C
13. LETRA A
14. LETRA C
15. ERRADO
16. LETRA A
17. LETRA D
18. LETRA E
19. LETRA A
20. LETRA C
21. ERRADO
22. LETRA B
23. LETRA D
24. LETRA D
25. ERRADO
26. ERRADO
27. LETRA D
28. LETRA B
29. CORRETO
30. LETRA B
31. LETRA A

ESSA LEI TODO MUNDO CONHECE:
PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.

1. 

Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.

2. 

Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).

3. 

Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).

4. 

Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).

5. 

Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).

6. 

Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).

7. 

Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.

8. 

O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.

Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.