XML (Extensible Markup Language)

- XML é uma Recomendação W3C
- XML apenas descreve os dados e o que eles significam
  - O HTML que foi desenvolvido para mostrar os dados e a "aparência" deles
  - XML é um complemento ao HTML e não um substituto deste
- XML foi desenvolvido para estruturar, armazenar e enviar os dados
- *"XML is a cross-platform, software and hardware independent tool for transmitting information" – W3Schools*
- Com XML os dados são mantidos separados do seu código HTML
- Com XML a troca de dados entre sistemas incompatíveis é possível
- As *tags* do XML não são pré-definidas, você deve criar as suas próprias *tags*, obedecendo a um pequeno conjunto de regras de sintaxe.
- **Regras de Sintaxe do XML**
  - **Todos os documentos XML devem conter a "*declaração XML*"**
    - Define a versão do XML e a codificação de caracteres usada no documento
    - *<?xml version="1.0" encoding="ISSO-8858-1"?>*
    - A declaração XML não é parte do documento XML, não é um elemento do documento XML, e por isso não precisa de uma *tag* de fechamento
  - **Todos os documentos XML devem conter um, e apenas um, elemento raiz**
    - *<raiz> ... </raiz>*
    - Entre as *tags* do elemento raiz que estarão todas as outras *tags* do seu documento XML
    - Elemento que "diz" o que é o documento
  - **Todos os elementos devem ter uma *tag* de fechamento**
    - Existem duas construções válidas
    - *<teste>Teste 1</teste>* o elemento teste tem conteúdo
    - *<teste />* o elemento teste é vazio
    - Diferente do HTML que pode ter tags que não são fechadas
  - **XML é *case sensitive***
    - *<teste>* é diferente de *<Teste>>*
  - **Os elementos XML devem estar corretamente aninhados**
    - Correto: *texto em negrito e em itálico*****
    - Incorreto: *texto em negrito e em itálico*****
  - **Os valores dos atributos devem estar entre aspas**
    - Tanto faz se são aspas duplas ou simples
    - Correto: *<data="22/12/1981"></data>*
    - Incorreto:*<data=22/12/1981></data>*
  - **Os nomes dos elementos devem seguir as regras**
    - Nomes podem possuir letras, números e outros caracteres
    - Nomes não podem começar com número ou caractere de pontuação
    - Nomes não podem começar com as letras XML e suas variações
    - Nomes não podem conter espaços
  - **Sintaxe dos comentários:**
    - *<!--Comentário-->*
- **Um documento XML que siga essas regras de sintaxe é um documento XML Bem Formado**

- **Elementos XML**

```
<?xml version="1.0" encoding="ISSO-8859-1"?>
<familiares>
        <filho1 atributo="1">Conteúdo do elemento</filho1>
        <filho2 atributo="332">Conteúdo do elemento</filho2>
</familiares>
```

  - O elemento raiz é *familiares*
  - Os elementos *filho1* e *filho2* são os elementos filhos de *familiares*
  - *familiares* é o elemento pai de *filho1* e *filho2*
  - *filho1* e *filho2* são elementos irmãos por possuírem o mesmo pai

- **Atributos**
  - Todos os elementos XML podem conter atributos
  - Os atributos geralmente carregam informações que não são parte dos dados
  - Como já foi mencionado, todos os valores dos atributos devem estar entre aspas, duplas ou simples
  - Não existem regras que digam quando usar elementos filhos ou atributos
  - Os exemplos a seguir carregam a mesma informação:

```
<pessoa sexo="feminino">
        <primeiro_nome>Lya</primeiro_nome>
        <ultimo_nome>Castro</ultimo_nome>
</pessoa>
```

```
<pessoa>
        <sexo>feminino</sexo>
        <primeiro_nome>Lya</primeiro_nome>
        <ultimo_nome>Castro</ultimo_nome>
</pessoa>
```

- **Documento XML válido**
  - Um documento XML é dito válido se, além de ele ser bem formado, ele estiver de acordo com um *DTD* ou *XML Schema.*
- **Utilizando CSS com XML**
  - É possível formatar um documento XML utilizando folhas de estilo CSS
  - Não é o padrão W3C
    - Coloca-se a descrição abaixo no documento XML para ligar um CSS ao documento XML
    - *<?xml-stylesheet type="text/css"href="arquivo.css"?>*
- **Utilizando XSL com XML**
  - É o padrão W3C para formatar documentos XML:
    - *<?xml-stylesheet type="text/xsl" href="simple.xsl"?>*
  - Este assunto será abordado mais adiante.

- **Prefixos e XML Namespaces**
  - São utilizados para resolver conflitos de nomes entre documentos XML
  - No primeiro exemplo a tag *<table>* se refere a uma tabela enquanto que no segundo exemplo a tag *<table>* se refere a uma mesa, existindo claramente um conflito de nomes:

```
<table>
   <tr>
   <td>Apples</td>
   <td>Bananas</td>
   </tr>
</table>
```

```
<table>
   <name>Coffee Table</name>
   <width>80</width>
   <length>120</length>
</table>
```

  - Uma maneira de resolver conflitos é utilizando prefixos:

```
<h:table>
   <h:tr>
   <h:td>Apples</h:td>
   <h:td>Bananas</h:td>
   </h:tr>
</h:table>
```

```
<f:table>
   <f:name>Coffee Table</f:name>
   <f:width>80</f:width>
   <f:length>120</f:length>
</f:table>
```

  - Outra maneira de resolver os conflitos é utilizando namespaces
    - Utilizamos o atributo *xmlns* para definir um namespace em uma *tag*
    - *xmlns:namespace-prefix="namespaceURI"*
    - Todos os filhos do elemento no qual o namespace foi definido terão associados a eles o mesmo namespace do elemento pai
  - Os exemplos abaixo são os mesmos anteriores só que com a utilização dos namespaces:

```
<h:table xmlns:h="http://www.w3.org/TR/html4/">
   <h:tr>
   <h:td>Apples</h:td>
   <h:td>Bananas</h:td>
   </h:tr>
</h:table>
```

```
<f:table xmlns:f="http://www.w3schools.com/furniture">
   <f:name>Coffee Table</f:name>
   <f:width>80</f:width>
   <f:length>120</f:length>
</f:table>
```

  - Podemos definir um *namespace* padrão para o documento para que não seja necessário o uso dos prefixos em todos os elementos do documento XML
    - *xmlns="namespaceURI"*

```
<table xmlns="http://www.w3.org/TR/html4/">
   <tr>
   <td>Apples</td>
   <td>Bananas</td>
   </tr>
</table>
```

```
<table xmlns="http://www.w3schools.com/furniture">
   <name>Coffee Table</name>
   <width>80</width>
   <length>120</length>
</table>
```

- **Caracteres Ilegais**
  - Alguns caracteres são ilegais no documento XML e causarão erro se utilizados. Para evitar isso utilizamos as entidades de referência da tabela abaixo:

&lt;	<	Menor que
&gt;	>	Maior que
&amp;	&	E comercial
&em;	‘	Apóstrofo
&quot;	“	Aspas

    - Com erro: *<message>4M salary < 1000 then</message>*
    - Sem Erro: *<message>4M salary < 1000 then</message>*
    - **Observação:** somente os caracteres “<” e “&” são ilegais no XML, os outros são legais, mas é uma boa prática também substituí-los

- **XML CDATA**
  - Todo o texto em um documento XML será analisado pelo *parser*
  - Para que algum texto não seja analisado pelo *parser* devemos colocá-lo dentro de uma seção CDATA:
    - **<![CDATA[** ... **]]>**
    - Todo o conteúdo de uma seção CDATA é ignorado pelo *parser*
    - O XML não permite que existam seções CDATA aninhadas

```
<script>
<![CDATA[
function matchwo(a,b)
{
if (a < b && a < 0) then
   {
   return 1
   }
else
   {
   return 0
   }
}
]]>
</script>
```

DTD (Document Type Definition)

- Um documento XML é válido se ele, além de ser bem formado, segue as regras de um DTD ou de um XML Schema
- O propósito do DTD é definir os blocos válidos de um documento XML. Ele define a estrutura do documento como uma lista de elementos e atributos válidos.
- O DTD pode ser declarado dentro do documento XML ou pode ser criado num outro documento e ser referenciado no XML
- *"DTD define a construção de blocos válidos para um documento XML, bem como a estrutura desse documento, usando uma lista de elementos válidos" FCC*
- *"Permite descrever cada marca (tag) e fornecer regras para interpretar cada informação usada em um arquivo XML" FCC*
- **Declaração Interna**
  - *<!DOCTYPE elemento-raiz [declaração-dos-elementos]>*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note [
        <!ELEMENT to      (#PCDATA)>
        <!ELEMENT from    (#PCDATA)>
        <!ELEMENT heading (#PCDATA)>
        <!ELEMENT body    (#PCDATA)>
]>
<note>
   <to>João</to>
   <from>Isa</from>
   <heading>Lembrete</heading>
   <body>É hoje</body>
</note>
```

Define o elemento raiz

Define o tipo de dados dos elementos como PCDATA

Fecha o DTD

- **Declaração Externa**
  - É necessário associar o XML ao DTD
    - Adiciona-se a linha abaixo no documento XML
    - *<!DOCTYPE elemento-raiz SYSTEM "nome-do-arquivo">*
    - Cria-se um documento com as informações do DTD

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
   <to>João</to>
   <from>Isa</from>
   <heading>Lembrete</heading>
   <body>É hoje</body>
</note>
```

Arquivo XML

```
<!ELEMENT note    (to,from,heading,body)>
<!ELEMENT to      (#PCDATA)>
<!ELEMENT from    (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body    (#PCDATA)>
```

Arquivo note.dtd

Blocos de Construção

Um documento DTD pode conter:

Elementos Atributos Entidades PCDATA CDATA

1 . Declaração dos elementos

<!ELEMENT nome-do-elemento categoria>
<!ELEMENT nome-do-elemento (conteúdo-do-elemento)>

- **Elementos vazios**

<!ELEMENT nome-do-elemento EMPTY>
Exemplo DTD:
<!ELEMENT br EMPTY>
Exemplo XML:

- **Elementos com PCDATA, CDATA ou qualquer conteúdo**

<!ELEMENT nome-do-elemento (#PCDATA)>
Exemplo DTD:
<!ELEMENT from (#PCDATA)>

<!ELEMENT nome-do-elemento ANY>
Exemplo DTD:
<!ELEMENT note ANY)>

- **Elementos com filhos**
  - Os filhos devem aparecer no documento XML na ordem exata na qual foram declarados no DTD

<!ELEMENT nome-do-elemento (filho1)>
<!ELEMENT nome-do-elemento (filho1,filho2,...)>
Exemplo DTD:
<!ELEMENT note (to,from,heading,body)>

- **Declaração de elementos com apenas uma ocorrência**

<!ELEMENT nome-do-elemento (nome-do-filho)>
Exemplo DTD:
<!ELEMENT note (mensagem)> → somente uma ocorrência de mensagem

- **Declaração de elementos com no mínimo uma ocorrência**

<!ELEMENT nome-do-elemento (nome-do-filho+)>
Exemplo DTD:
<!ELEMENT note (mensagem+)> → no mínimo uma ocorrência de mensagem

- **Declaração de elementos com zero ou mais ocorrências**

<!ELEMENT nome-do-elemento (nome-do-filho)>*
Exemplo DTD:
<!ELEMENT note (mensagem*)>

- **Declaração de elementos com zero ou uma ocorrência**

<!ELEMENT nome-do-elemento (nome-do-filho?)>
Exemplo DTD:
<!ELEMENT note (mensagem?)> → zero ou exatamente uma ocorrência de mensagem

- **Declaração de conteúdo do tipo "um ou outro"**

Exemplo DTD:
<!ELEMENT note (to,from,header,(message|body))>
O exemplo indica que o elemento *note* deve conter **um** elemento *to*, **um** *to* e **um** *header* e ainda **um** *message* **ou um** *body*

- **Declaração de conteúdo misto**
  - Podemos misturar o conteúdo dos elementos da maneira que acharmos conveniente

Exemplo DTD:
<!ELEMENT note (#PCDATA|to|from|header|message)*>
O exemplo indica que o elemento *note* pode conter zero ou mais ocorrências de *PCDATA*, *to*, *from*, *header* ou *message*

2 . Declaração dos atributos

<!ATTLIST element-name attribute-name attribute-type default-value>

Attribute-type	Descrição
CDATA	O valor é CDATA
(*em1*|*em2*|..)	O valor deve ser um dos contidos na enumeração
ID	O valor é um identificador único
IDREF	O valor é um ID de outro elemento
IDREFS	O valor é uma lista de Ids
NMTOKEN	O valor é um nome XML válido
NMTOKENS	O valor é uma lista de nomes XML válidos
ENTITY	O valor é uma entidade
ENTITIES	O valor é uma lista de entidades
NOTATION	O valor é um nome de uma notação
XML:	O valor é um valor XML predefinido

Default-value	Descrição
value	O valor padrão do atributo se nada for declarado
#REQUIRED	O atributo é obrigatório
#IMPLIED	O atributo não é obrigatório
#FIXED *value*	O atributo tem um valor fixo que não pode ser mudado

DTD:
<!ELEMENT quadrado EMPTY>
<!ATTLIST quadrado largura CDATA "0">
XML válido:
<quadrado largura="100" />

DTD:
<!ATTLIST person number CDATA #REQUIRED>
XML válido:
<person number="5677" />
XML inválido:
<person />

DTD:
<!ATTLIST telefone fax CDATA #IMPLIED>
XML válido:
<telefone fax="555-667788" />
Esse também é válido:
<telefone/>

DTD:
<!ATTLIST sender company CDATA #FIXED "Microsoft">
XML válido:
<sender company="Microsoft" />
XML inválido:
<sender company="Apple" />

DTD:
<!ATTLIST pagamento tipo (check|cash) "cash">
Exemplo de XML:
<payment type="check" />
ou
<payment type="cash" />

3. Entidades

- As entidades são variáveis usadas como “atalhos” para algum texto padrão ou caracteres especiais
- A declaração das entidades pode ser interna ou externa:

<!ENTITY nome-da-entidade "valor-da-entidade">
<!ENTITY nome-da-entidade SYSTEM "URI/URL">

Exemplo de declaração DTD interna:
<!ENTITY escritor "João Castro.">
<!ENTITY copyright "Copyright Resumos&Apostilas.">
Exemplo XML
<autor>&escritor;&copyright;</autor>

Exemplo de declaração DTD externa:
<!ENTITY escritor SYSTEM "http://resumosapostilas.blogspot.com/entidades.dtd">
<!ENTITY copyright SYSTEM" http://resumosapostilas.blogspot.com/entidades.dtd ">
Exemplo XML
<autor>&escritor;&copyright;</autor>

XSL = eXtensible Stylesheet Language

- XSL são as folhas de estilo do XML
  - XSL descreve como o documento XML deve ser apresentado
  - Recomedação W3C
- XSL consiste em três partes
  - XSLT = linguagem para transformar documentos XML
  - Xpath = linguagem para navegar nos documentos XML
  - XSL-FO = linguagem para formatar documentos XML

XSLT = e**X**tensible **S**tylesheet **L**anguage **T**ransformer

- É uma recomendação W3C
- XSLT transforma um documento XML em outro documento que pode ser XML, XHTML, HTML ou qualquer linguagem reconhecida por um browser
- Um arquivo XSL uma folha de estilos para o XML.
- Os arquivos XSL são na verdade arquivos XML que utilizam o namespace específico do XSL, com o prefixo **"xsl"**
- O elemento raiz do XSLT pode ser o *<xsl:stylesheet>* ou, opcionalmente, o *<xsl:transform>*

```
<?xml version="1.0" encoding="ISSO-8858-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3c.org/1999/XSL/Transform">
...
</xsl:stylesheet>
Ou
<?xml version="1.0" encoding="ISSO-8858-1"?>
<xsl: transform version="1.0" xmlns:xsl="http://www.w3c.org/1999/XSL/Transform">
...
</xsl:transform>
```

Exemplos de arquivos XSL

- É necessário incluir uma referência ao arquivo XSL no seu documento XML

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

- Vamos utilizar o XML simplificado abaixo e o XSL logo a seguir

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="combustível.xsl"?>
<combustível>
   <abastecimento id="1">
        <dia>22</dia>
        <mês>05</mês>
        <ano>2007</ano>
        <km_total>103045 km</km_total>
        <km_rodados>452,5 km</km_rodados>
        <gas>31,132 litros</gas>
        <preço_litro>R$2,68</preço_litro>
        <valor>R$83,43</valor>
   </abastecimento>
...
</combustível>
```

```
<?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>
        <h1>Abastecimentos</h1>
        <table border="1">
        <tr bgcolor="#9acd32">
                 <th align="left">Dia</th>
                 <th align="left">Mês</th>
                <th align="left">Ano</th>
                <th aling="left">KM Total</th>
                 <th align="left">Km Rodados</th>
                <th align="left">Gasolina</th>
                <th align="left">Preço/litro</th>
                <th align="left">Valor</th>
        </tr>
  <xsl:for-each select="combustível/abastecimento">
  <tr>
        <td><xsl:value-of select="dia"/></td>
        <td><xsl:value-of select="mês"/></td>
        <td><xsl:value-of select="ano"/></td>
         <td><xsl:value-of select="km_total"/></td>
        <td><xsl:value-of select="km_rodados"/></td>
        <td><xsl:value-of select="gas"/></td>
        <td><xsl:value-of select="preço_litro"/></td>
         <td><xsl:value-of select="valor"/></td>
  </tr>
  </xsl:for-each>
  </table>
 </body>
 </html>
</xsl:template>
```

- Para testar grave o primeiro arquivo como *combustível.xml* e o segundo como *combustível.xsl* e abra o XML no browser:

Abastecimentos

Dia	Mês	Ano	KM Total	Km Rodados	Gasolina	Preço/litro	Valor
22	05	2007	103045 km	452,5 km	31,132 litros	R$2,68	R$83,43
29	05	2007	103480 km	434,8 km	30,38 litros	R$2,69	R$81,72

- *Observação!*
  - A FCC gosta de tentar confundir os candidatos com o formato do arquivo XSL
  - XSL = extensible stylesheet language
  - XLS = arquivo do Excel

1. **Elemento <xsl:template match="XPath"> (tem tag de fechamento)**
   - É um elemento que define um "molde", ou "padrão" em conjunto com o atributo ***match***
     - O valor do atributo match é uma expressão XPath válida
     - Uma expressão XPath funciona como navegar em um *filesystem*, no qual uma barra / seleciona os subdiretórios
   - No exemplo,*<xsl:template match="/">* associa o molde ao elemento raiz do documento XML
   - O conteúdo dentro da tag *<xsl:template>* é código em XHTML que define como será exibido o conteúdo do arquivo XML no browser
2. **Elemento <xsl:value-of select="XPath"> (o fechamento é feito na própria tag)**
   - É utilizado para extrair o valor de um nó específico dentro do arquivo XML
   - O valor do atributo *select* deve ser uma expressão XPath válida
   - No nosso exemplo o elemento *<xsl:value-of select="dia"/>* extrai o valor do nó *dia*
3. **Elemento <xsl:for-each select="XPath"> (tem tag de fechamento)**
   - Utilizado para criar loops
   - Permite extrair todos os elementos quando utilizado em conjunto com o *xsl:value-of*
   - O valor do atributo *select* deve ser uma expressão XPath válida
     - No nosso exemplo definimos que deve ser extraído o valor de cada um dos filhos do elemento *abastecimento*
   - O bloco de repetição no nosso exemplo é:

```
<xsl:for-each select="combustível/abastecimento">
<tr>
        <td><xsl:value-of select="dia"/></td>
        <td><xsl:value-of select="mês"/></td>
        <td><xsl:value-of select="ano"/></td>
         <td><xsl:value-of select="km_total"/></td>
        <td><xsl:value-of select="km_rodados"/></td>
        <td><xsl:value-of select="gas"/></td>
        <td><xsl:value-of select="preço_litro"/></td>
         <td><xsl:value-of select="valor"/></td>
</tr>
</xsl:for-each>
```

- O elemento *<xsl:for-each>* pode ser utilizado para filtrar o resultado com o uso dos operadores lógicos:

=	Igual
!=	Não igual
<	Menor que
>	Maior que

```
Para extrair só os abastecimentos feitos no dia 22
<xsl:for-each select="combustível/abastecimento[dia='22']">
Para extrair abastecimentos com valores maiores que R$100
<xsl:for-each select="combustível/abastecimento[valor&gt;'100']">
```

4. Elemento <xsl:sort select="XPath"> (não tem tag de fechamento)

- Serve para ordenar os resultados
- É colocado dentro de um elemento *<xsl:for-each>*

```
<xsl:for-each select="combustível/abastecimento">
<xsl:sort select="valor">
<tr>
        <td><xsl:value-of select="dia"/></td>
        ...
        <td><xsl:value-of select="valor"/></td>
</tr>
</xsl:for-each>
```

5. Elemento <xsl:IF test="expressão de teste"> (tem tag de fechamento)

- Faz testes condicionais
- Deve ser colocado dentro de um elemento *<xsl:for-each>*

```
<xsl:for-each select="combustível/abastecimento">
<xsl:if test="valor &gt; 100">
<tr>
        <td><xsl:value-of select="dia"/></td>
        ...
        <td><xsl:value-of select="valor"/></td>
</tr>
</xsl:if>
</xsl:for-each>
```

→ Mostra somente os abastecimentos que com valores maiores que 100

6. Elementos <xsl:choose>, <xsl:when>, <xsl:otherwise> (tem tag de fechamento)

- Utilizados para fazer múltiplos testes condicionas

```
<xsl:choose>
 <xsl:when test="expression">
  ... algum código ...
 </xsl:when>
 <xsl:otherwise>
  ... algum código ....
 </xsl:otherwise>
</xsl:choose>
```

Bibliografia

W3Schools http://www.w3schools.com/

W3C http://www.w3c.org

Mini Curso Virtual Link

XML By Example

XSLT, O'Rilley