RESUMEN LINGUAXE DE MARCAS

XML

Diferencia XML vs Otros lenguajes de marcas como HTML:

-No existen etiquetas predefinidas. Para cada documento se definen unas que identifican su contenido para una definición concreta. Almacenan información

-Metalenguaje que permite definir otros lenguajes de marcas.

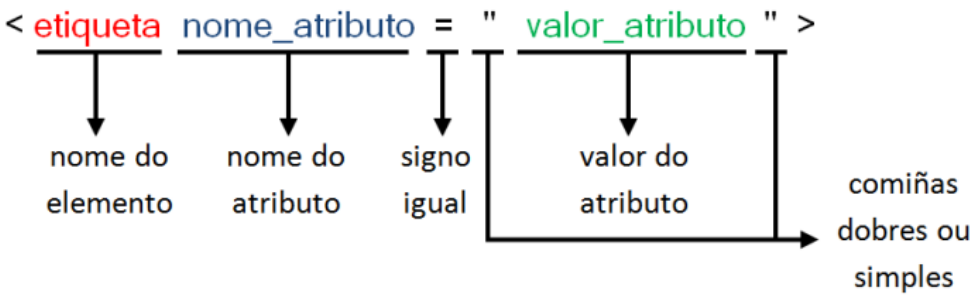
-Simplificación del SGML, para crear gramáticas de lenguajes de marcado. Se dejaron fuera las partes menos utilizadas, de forma que las especificaciones del XML ocupan 30 páginas.

Ventajas del empleo de XML:

-Creado para estructurar, almacenar y transportar información.

-Creación, interpretación y procesado informático sencillos.

-Compatibilidad con SGML.



El nombre del elemento o atributo no puede empezar por caracteres no válidos (números, +, ., -, :, palabra reservada "xml")

Caracteres prohibidos en el texto: < o &. Si queremos que aparezcan hacemos referencia a un carácter Unicode. '&' -> '&' '<' -> '<'

Empleando entidades

"&" → "&".
"<" →"<".
">" →">".
""" →"""".
"'" → "'".

Espacios en blanco

Para preservar los espacios en el texto hay que añadir el atributo xml:space="preserve" o xml:space="default" si no se quieren tener en cuenta.

Identificación del idioma del contenido

Utilizando un atributo xml:lang="gl" para gallego, o "es" para español y "en" para inglés.

Contenido CDATA

Utilizando la siguiente estructura en el texto, el analizador ignora cualquier carácter prohibido utilizado.

<![CDATA[aquí se introduce el texto con caracteres prohibidor]]>

Puede ser útil para introducir contenido en lenguaje HTML.

Elementos vs Atributos

Los atributos están pensados para guardar info o descripciones sobre los datos y sólo pueden almacenar un valor cada uno, mientras que los elementos pueden almacenar varios datos e incluso extenderse mediante estructuras anidadas de varios elementos.

Bien formado: Cumple las reglas generales del XML, como por ejemplo, la especificación XML 1.0.

XML + DTD

Los documentos XML no nos permiten saber si la estructura que siguen es válida o no para el tipo de documento que se está tratando de crear.

La validación permite comparar un documento XML con un documento que define la estructura y las reglas a seguir. La gramática bien formada del XML no es suficiente.

Standalone

Especifica si el documento XML depende de otra gramática distinta a las normas impuestas por XML. Por ejemplo, si cuenta con una DTD → standalone="no"

Analizadores

El parser es una herramienta que interpreta un documento XML, a menudo el documento textual a una página web. El doc debe estas bien formado y validado (si cuenta con DTD) para poder ser interpretado.

Construcción de una DTD

La definición de la gramática de la DTD viene dada por la etiqueta:

```
<!DOCTYPE nomeElementoRaiz [ declaracions ]>
```

si es interna y:

```
<!DOCTYPE nomeElementoRaiz SYSTEM "nomeArquivo.dtd">
```

si es externa.

Definir **elementos**

```
<!ELEMENT nombreElemento contenido>
```
si es simple

```
<!ELEMENT nombreElemento (subelementos, texto)>
```
si tiene subelementos

El tipo de CONTENIDO que pueden tener los elementos son:

-EMPTY: no puede tener ni subelementos ni datos textuales, pero sí atributos.

DTD: `<!ELEMENT nombre EMPTY>`
XML: `<nombre />`

-ANY: contiene cualquier cosa, datos textuales, otros elementos…

-PCDATA: indica que el contenido del elemento son datos de texto.

DTD: `<!ELEMENT titulo (#PCDATA)>`
XML: `<titulo>El lenguaje XML</titulo>`

-MIXED: permite que el contenido sean datos textuales o mezcla de datos textuales y subelementos.

DTD: `<!ELEMENT objeto (#PCDATA | imagen)*>`

-ELEMENT: solamente puede contener subelementos empleando modelos de contenidos.

Modelos de **contenidos**

-Hijo único o en un orden determinado

```
<!ELEMENT ciclo (código, nombre,grado)>

<!ELEMENT tutor (nombre)>
```

-Opción de que aparezcan O unos U otros.

```
<!ELEMENT ciclo ((ciclo|nombre), grado)>
```

Frecuencia de contenidos

Se indica mediante los siguiente signos al lado del contenido, tanto en grupo como en solitario.

? ->El elemento aparece 0 o 1 vez.

+ ->El elemento aparece 1 o n veces.

'*' ->El elemento aparece 0 o n veces.

Definir **atributos**

Los atributos no contienen subatributos. Se utilizan para dar un valor por defecto, definir unos valores válidos y fijos. También para crear referencias entre distintos elementos. Los elementos pueden tener 0, 1 o varios atributos y se declaran mediante:

```
<!ATTLIST nombreElemento
      nombreAtributo1 tipo valor
      nombreAtributo2 tipo valor
      …
>
```

Por ejemplo:

```
<!ATTLIST ciclo
      código CDATA #REQUIRED
      grado CDATA #REQUIRED>
```

Valores de los atributos

#IMPLIED: valor opcional

Si se quiere indicar un valor por defecto opcional se escribe ese valor entre "" y no se indica el valor IMPLIED.

#REQUIRED: valor obligatorio

#FIXED: fijo

A continuación se #FIXED se indica el valor fijo entre "".

Tipos de atributos

CDATA

O characterdata, indican que el atributo contiene datos de texto

Enumeraciones

Para limitar los valores que puede tomar un atributo e una lista definida. Ya no es necesario indicar el CDATA.

```
<!ATTLIST curso
      nivel (bajo | media | alto) #IMPLIED>
```

ID, IDREF o IDREFS

Los atributos ID hacen referencia a un camplo clave. El identificador debe ser único para ese elemento y viene indicado como:

```
<!ATTLIST profesor
      nif ID #REQUIRED>
```

Los atributos IDREF hacen referencia a los indicadores de otro atributo de un elemento ya existente. El IDREF debe coincidir en su valor con el ID al que hace referencia.

```
<!ATTLIST alumno
      tutor IDREF #IMPLIED>
```

Los atributos IDREFS permiten hacer varias referencia dentro de un mismo atributo. Las distintas referencias van separadas por un espacio en blanco.

```
<!ATTLIST receta id ID #REQUIRED>

<!ATTLIST ingrediente ref IDREFS #IMPLIED>
```

En XML:

```
<receta id=”rec_1”>Filloas</receta>

<ingrediente ref=”rec_1 rec_2”>Huevos</ingrediente>
```

Atributos de tipo ENTITY

Los atributos de tipo ENTITY permiten definir constantes para el documento o pueden hacer referencia a elementos externos que no deben ser analizados sintácticamente según el XML.

Existen entidades predefinidas, que permiten mostrar caracteres prohibidos en XML:

Entidad: < -> <

Entidad: > -> >

Entidad: & -> &

Entidad: ' -> ‘

Entidad: " -> “

Y las entidades definidas:

```
<!ENTITY nombreEntidad “definicionEntidad”>
```

Ejemplo: `<!ENTITY ciudad “Santiado de Compostela”>`

Para aplicarla durante el documento XML utilizamos:

```
&nombreEntidad;
```

O como el ejemplo: `&ciudad;`

Para consultar más sobre entidades internas, externas y de parámetro -> ver páginas 12,13,14 de LinguaxeXML_actividadesValidacionConDTD_platega_v2.pdf

XML + SCHEMA

Ventajas del Schema frente a la DTD para la validación de XML

-Puede dar más detalle del tipo de datos.

-La DTD valida la estructura, pero no su contenido.

-El Schema está en el mismo lenguaje que el XML.

-Permiten la ocurrencia de secuencias desordenadas.

-Soporta espacios de nombre o namespace.

-Permite hace referencias a partir de varios atributos.

El Schema exige que los elementos tengan el prefijo xs: asociado al nombre de esquema XML declarado en `xmlns:xsd=http://www.w3.org/2001/XMLSchema`. También se usa el prefijo xs: para los nombres de tipos simples predefinidos como `xs:string`.

El propósito es identificar los elementos y los tipos simple pertenecientes al vocabulario del lenguaje Esquema XML.

Vinculación del XML con el Schema

Ver página 5 y 6 de XML_XSD_actividades_verAlumno.pdf

Sin tener su propio namespace

- Esquema .xsd

```
<?xml version=”1.0” … ?>
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema
      elementFormDefault=”qualified”>
…definición esquema…
</xs:schema>
```

- Documento XML

```
<?xml version=”1.0” …?>
<elementoRaiz xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
      xsi:noNamespaceSchemaLocation=”esquema.xsd”>
…
</elementoRaiz>
```

Construcción de esquemas

Tipos simples

Aquellos elementos que contienen únicamente texto. Estructura básica del elemento:

```
<xs:element name=”xxx” type=”xs:tipoSimple” default/fixed=””/>
```

Son opcionales:
Default -> valor por defecto
Fixed -> valor fijo

En cuanto a los tipos predefinidos, encontramos:

Categoría	Tipo	Exemplo	Observacións
Texto	xs:string	Calquera cadea de caracteres alfanumérica	
	xs:normalizedString	Calquera cadea de carácteres alfanumérica	Os caracteres de nova liña, tabulador e retorno de carro son convertidos a espazos en branco antes do procesamento do esquema.
	Xs:token	Calquera cadea de caracteres alfanumérica	Os espazos en branco adxacentes son comprimidos a un único espazo en branco, e os espazos en branco de diante e detrás son eliminados
	xs:anyURI	http://www.example.com	
Números	xs:byte	125, -2	
	xs:unsignedByte	0, 45	
	xs:short	-1, 12534	
	xs:unsignedShort	0, 12534	
	xs:integer	-1, 126789675	
	xs:unsignedInt	0, 126789675	
	xs:positiveInteger	1, 12345	Maior que 0 (>0)
	xs:nonPositiveInteger	0, -1, -12345	Menor igual que 0 (<=0)
	xs:negativeInteger	-1, -126789	Menor que 0 (<0)
	xs:nonNegariveInteger	0, 1, 126789	Maior igual que 0 (>=0)
	xs:long	-1, 12678967541111	
	xs:unsignedLong	0, 12678967541111	
	xs:decimal	-1.23, 0, 123.4	Separador decimal é o punto
	xs:float	-1E4, 12.78E-2	equivalente a punto flotante de precisión simple de

Categoría	Tipo	Exemplo	Observacións
			32-bit,
	xs:double	-1E4, 12.78E-2	equivalente a punto flotante de precisión dobre de 64-bit,
Data/Hora	xs:dateTime	2010-12-31T13:20:00	formato especificado no estándar ISO/8601: Primeiro a data e despois a hora separados por unha T .12 de Decembro de 2010 ás 1.20pm
	xs:date	2012-12-31	formato "AAAA-MM-DD"
	xs:time	13:20:00	formato "hh:mm:ss"
	xs:gDay	- - -31	O día 31
	xs:gMonth	- -05- -	O mes de maio
	xs:gYear	2013	
	xs:gMonthDay	- -04-18	Cada 18 de abril
	xs:gYearMonth	2013-02	O mes de Febreiro de 2013, sen importar o número de días
Lóxicos	xs:boolean	true, false 1, 0	
Varios	xs:language	ES, EN, FR	especifica un idioma, no formato de dous caracteres establecido pola norma ISO639.
	xs:NMTOKEN	diaSemana	é unha cadea de caracteres composta por letras, números, puntos, dous puntos, guións e suliñados. Non pode ter espazos.
	xs:NMTOKENS	Nome completo	é unha cadea de caracteres composta por letras, números, puntos, dous puntos, guións e suliñados.. Pode conter espazos.

Estructura básica del atributo simple:

```
<xs:attribute name=”xxx” type=”xs:tipoSimple” default/fixed=””/>
```

Default y fixed son opcionales. Si queremos expresar la obligatoriedad o opcionalidad del atributo:

```
<xs:attribute name=”xxx” type=”xs:tipoSimple” use=”required”/”optional”/>
```

Required -> obligatorio
Optional -> opcional

Tipos complejos

Permiten definir elementos CON atributos y/o subelementos introduciendo `complexType`.

Elementos vacíos

CON atributos pero SIN contenido

```
<xs:element name=”distancia”>
      <xs:complexType>
            <xs:attribute name=”unidad” type=”xs:string”/>
            <xs:attribute name=”valor” type=”xs:decimal”/>
      </xs:complexType>
</xs:element>
```

Elementos con subelementos

CON subelementos pero SIN atributos

```
<xs:complexType>
      <xs:sequence>
            <xs:element name=”nombre” type=”xs:string”/>
            <xs:element name=”apellido1” type=”xs:string”/>
            <xs:element name=”apellido2” type=”xs:string” minOccurs=”0”/>
      </xs:sequence>
</xs:complexType>
```

Elemento que constan de texto

CON texto, CON atributo, SIN subelementos, añadimos simpleContent

```
<xs:element name=”distancia”>
      <xs:complexType>
            <xs:simpleContent>
                  <xs:extensión base=”xs:string”>
                        <xs:attribute name=”unidade” type=”xs:string”/>
                  </xs:extensión>
            </xs:simpleContent>
      </xs:complexType>
</xs:element>
```

Elementos que constan de texto y otros elementos

CON texto, CON subelementos entremezclados en el XML

```
<xs:element name="carta">
      <xs:complexType mixed="true">
            <xs:sequence>
                  <xs:element name="nome" type="xs:string"/>
                  <xs:element name="fillo" type="xs:string"/>
                  <xs:element name="numero" type="xs:unsignedShort"/>
                  <xs:element name="data" type="xs:date"/>
            </xs:sequence>
      </xs:complexType>
</xs:element>
```

Indicadores de orden de los elementos

<xs:sequence>

Indica el orden de aparición de los elementos

<xs:choice>

Indica que puede aparecer un/unos elemento/s u otro/s

```
<xs:complexType name="persoa">
```

```
      <xs:choice>
            <xs:element name="alcume" type="xs:string"/>
      <xs:sequence>
            <xs:element name="nome" type="xs:string"/>
            <xs:element name="apelido1" type="xs:string"/>
            <xs:element name="apelido2" type="xs:string" minOccurs=”0”/>
      </xs:sequence>
      </choice>
</xs:complexType>
```

En este ejemplo, los subelementos que pueden aparecer son O alcume O nombre +apelido1+apelido2

<xs:all>

Similar a sequence, sólo que no impone un orden, pueden aparecer en uno cualquiera.

Modos de declarar elementos

Ver página 12 de XML_XSD_actividades_verAlumnos.pdf

Se indica como se puede modulalizar el documento haciendo referencias a la estructura de los elementos en otra parte del documento a través de tipos y referencias.

Multiplicidad de elementos

Por defecto los elementos declarados son obligatorios una vez. Si queremos indicar otro tipo de frecuencia en su ocurrencia:

`minOccurs`=”0” si es opcional (0 o 1 vez)

`maxOccurs`=”unbounded” si no hay límite de ocurrencias

Creación de tipos propios

Se puede especificar el tipo de dato del contenido de elementos y atributos mediante restricciones. Las restricciones se aplican sobre un tipo existente “base”.

RESTRICCIÓN	DESCRIPCIÓN
length	Número exacto de caracteres da cadea ou número mínimo de elementos da lista permitidos. Debe ser igual ou maior que cero
minLength	Número mínimo de caracteres da cadea ou número mínimo de elementos da lista permitidos. Debe ser igual ou maior que cero
maxLength	Número máximo de caracteres da cadea ou número máximo de elementos da lista permitidos. Debe ser
	igual ou maior que cero
pattern	O contido debe encaixar cunha expresión regular
enumeration	Define unha lista de valores aceptables
whiteSpace	Controla a normalización dos espazos en branco (saltos de liña, tabuladores, espazos e retornos de carro)
minInclusive	Indica o límite inferior do valor numérico (debe ser maior ou igual ao valor indicado)
minExclusive	Indica o límite inferior do valor numérico (debe ser maior ao valor indicado)
maxInclusive	Indica o límite superior do valor numérico (debe ser menor ou igual ao valor indicado)
maxExclusive	Indica o límite superior do valor numérico (debe ser menor ao valor indicado)
totalDigits	Indica o máximo número de díxitos nos tipos numéricos. Debe ser maior que 0
fractionDigits	Indica o máximo número de decimais nos tipos numéricos. Debe ser maior ou igual que 0

Dependiendo del tipo de "base" que asignemos podemos imponer una restricción de distinto tipo.

Por ejemplo, para xs:string, podemos aplicar restricciones: length, minLength, maxLength, pattern, enumeration, whitespace. Y para xd:integer podemos aplicar restricciones: totalDigits, pattern, whitespace, enumeration, maxInclusive, maxExclusice, minInclusive, minExclusive.

Restricción sobre LONGITUD max y min

```
<xs:simpleType name="nomeCorto">
      <xs:restriction base="xs:string">
            <xs:maxLength value="15"/>
      </xs:restriction>
 </xs:simpleType>
```

Restricción sobre el intervalo de VALORES max y min

```
<xs:element name="idade">
      <xs:simpleType>
             <xs:restriction base="xs:byte">
                  <xs:minInclusive value="0"/>
                  <xs:maxInclusive value="3"/>
            </xs:restriction>
      </xs:simpleType>
</xs:element>
```

Restricción sobre el conjunto de valores que puede tomar

```
<xs:simpleType name="tipoGrao">
      <xs:restriction base="xs:string">
            <xs:enumeration value="medio"/>
            <xs:enumeration value="superior"/>
      </xs:restriction>
</xs:simpleType>
```

Restricción de seguir un PATRÓN específico

```
<xs:simpleType name="tipoTelefono">
      <xs:restriction base="xs:string">
            <xs:pattern value="([0-9]{3})\s[0-9][0-9]-[0-9][0-9]-[0-9][0-9]"/>
      </xs:restriction>
</xs:simpleType>
```

Para saber como crear un patrón hay que seguir la tabla:

Patrón	Descrición	Exemplo		Non validaría
^	Coincidencia ao principio da cadea	^Isto	Isto é unha cadea	É isto de aquí
$	Coincidencia ao final da cadea	final$	Chegamos ao final	
*	O caracter que o precede pode aparecer 0, 1 ou máis veces	ma*	maaaaa ou ma ou m	
+	O carácter que o precede debe aparecer polo menos una vez (*1 ou máis veces*)	ma+	maaaaa ou ma	m
?	O carácter que o precede pode aparecer como moito unha vez (0 ou 1 vez)	ma?	ma ou m	maaaaa
[...]	Calquera caracter dentro dos paréntesis	a[px]e	ape ou axe	ale
[-]	Indica un rango de caracteres	[c-k]a	ca ou ea	pa
[^...]	Calquera caracter menos os que están entre paréntesis	a[^px]e	ale ou a1e	ape ou Axe

[^ -]	Calquera caracter menos os que estean nese rango	[^c-k]a	pa	ea
{n}	O carácter que o precede debe aparecer exactamente n veces	bua{2}	buaa	bua
{n,}	O carácter que o precede debe aparecer n veces ou máis veces	bu{2,}	buuuuu ou buu	bu
{n,m}	O carácter que o precede debe aparecer como mínimo n e como máximo m veces	[0-9]{2,4}	32 ou 4444	8 ou 78978
(...)	Podemos agrupar con paréntese	(co){2}liso	cocoliso	Coliso
|	Unha barra vertical separa alternativas	M[0-9] | MP	MP ou M9	MA 9M
\d	Calquera díxito (é equivalente a [0-9])	\d{2}	23	S7
\D	Calquera non díxito	\D{2}M	L_M ou ABM	4CM
\w	Calquera letra ou díxito	ala\w	ala4 ou alag	ala_
\W	Calquera caracter diferente de letra ou díxito	M\Wa	M-a	M1a
.	Calquera caracter salvo salto de liña			
\n	Salto de liña			
\s	Caracter en branco, tabulador ou salto de liña			

Restricciones de espacios en blanco

Ver página 16 -> <xs:whiteSpace value=”preserve, replace y collapse”>

Restricción de UNION de tipos

Ver página 18 -> <xs:union>

Formación de grupos

Ver página 20 -> <xs:group> o <xs:attributeGroup>

Identificadores claves

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
      <xs:element name="empresa">
            <xs:complexType>
                  <xs:sequence>
                        <xs:element name="empregado" type="tipoEmpregado"
maxOccurs="200"/>
                        <xs:element name="departamento" type="tipoDepartamento"
maxOccurs="8"/>
                  </xs:sequence>
            </xs:complexType>

            <xs:key name="depUnico">
                  <xs:selector xpath="departamento"/>
                  <xs:field xpath="@codigo"/>
            </xs:key>

            <xs:keyref name="departamento" refer="depUnico">
                  <xs:selector xpath="empregado"/>
                  <xs:field xpath="departamento"/>
            </xs:keyref>
      </xs:element>

 <xs:complexType name="tipoEmpregado">
 <xs:sequence>
 <xs:element name="nome" type="xs:string"/>
 <xs:element name="departamento" type="xs:string"/>
 </xs:sequence>
 </xs:complexType>

 <xs:complexType name="tipoDepartamento">
 <xs:sequence>
 <xs:element name="nome" type="xs:string"/>
 </xs:sequence>
 <xs:attribute name="codigo" type="xs:string" use="required"/>
 </xs:complexType>
</xs:schema>
```