XML Resumen

1. XML

XML (eXtensible Markup Language) se trata de un lenguaje de marcado estándar.

Tiene la siguiente estructura:

```
<?xml version="1.0" encoding="utf-8"?>
<raiz atributo="algo">
   <elem1>
…
   </elem1>
</raíz>
```

Tiene una serie de particularidades:

- La primera línea (<¿xml…) se llama **prólogo**.
- Los **elementos vacíos** se indican:
  - <elem></elem>
  - <elem />
- Siempre debe haber un elemento **raíz**.
- Los **comentarios** se ponen:
  - <!--comentario -->
- **No** se permiten **espacios en los nombres**:
  - <elem 1></elem 1> -> Está mal.
- Se permiten acentos, números y signos de puntuación (aunque no os lo recomiendo).

2. DTD (Document Type Definition)

DTD permite describir la estructura de un documento XML. En la actualidad se suele utilizar XSD en lugar de DTD.

Un XML es correcto si:

- **Está bien formado**: respeta reglas de:
  - Prólogo válido.
  - Etiquetas cerradas.
  - Anidación correcta.
  - Atributos entre comillas.
  - Nombres con caracteres válidos.
  - Tiene un elemento raíz.
- **Se puede validar**: respeta reglas que nosotros definimos en un DTD o XSD. En este caso el XML sería VALID.

2.1. Inclusión de DTD

El código puede ir en:

- **El documento XML**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE pizzas [
<!ELEMENT pizzas (pizza+)>
<!ELEMENT pizza (ingrediente+, bebida?)>
<!ATTLIST pizza
    nombre CDATA #REQUIRED
    precio CDATA #IMPLIED>
<!ELEMENT ingrediente EMPTY>
```

- **En un fichero independiente (.dtd)**

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

2.2. Elementos

Define cada uno de los elementos o nodos del XML.

Hay diferentes **tipos**:

- <!ELEMENT A **(B,C)>** -> A contiene a B y seguido a C.
- <!ELEMENT A **(#PCDATA)>** -> A contiene datos textuales
- <!ELEMENT A **EMPTY>** -> A no contiene nada.
- <!ELEMENT A **ANY>** -> A contiene cualquier cosa (texto u otros elementos).

2.2.1. Operadores

- **?**: opcional
- **+**: 1 o más veces.
- *****: 0 o más veces.
- **A|B**: A o B (el orden no es importante.

2.3. Atributos

Define cada uno de los atributos de los nodos del XML.

<!ATTLIST nodoXML nombreAtributo CDATA #REQUIRED>

Hay varios **tipos** de atributos:

- **CDATA**: contiene texto.
- **ID**: es un identificador único.
- **IDREF**: Hace referencia al valor de otro nodo.
- **IDREFS**: Listado de referencias a otros nodos.
- **ENTITY**: Hace referencia a una entidad externa al documento XML.
- **ENTITYS**: Listado de entidades XML externas.
- **NMTOKEN**: Nombre de un XML válido.
- **NMTOKENS**: Lista de nombres XML válidos.

Los atributos pueden ser:

- **#IMPLIED**: opcional
- #**REQUIRED**: obligatorio
- **#FIXED "valor":** siempre tiene ese valor.

2.4. Ejemplo

XML - listin.xml

```
<?xml version="1.0"?>
<!DOCTYPE listin SYSTEM "listin.dtd">
<listin>
      <persona edad="48" sexo="hombre" id="ricky">
            <nombre>Ricky Martin</nombre>
            <email>ricky@puerto-rico.com</email>
            <relacion amigo-de="leticia" />
      </persona>
      <persona sexo="mujer" id="leticia">
            <nombre>leticia Casta</nombre>
            <email>castal@micasa.com</email>
      </persona>
      <persona sexo="mujer" id="bea">
            <nombre>Bea la Santa</nombre>
            <email>bea@micasa.com</email>
            <telefono>945235648</telefono>
            <relacion enemigo-de="ricky"/>
      </persona>
</listin>
```

DTD - listin.dtd

```
<!ELEMENT listin (persona+)>
<!ELEMENT persona (nombre, (telefono | email)*, relacion?)>
<!ATTLIST persona id ID #REQUIRED>
<!ATTLIST persona sexo (hombre | mujer) #IMPLIED>
<!ATTLIST persona edad CDATA #IMPLIED>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT relacion EMPTY>
<!ATTLIST relacion amigo-de IDREFS #IMPLIED>
<!ATTLIST relacion enemigo-de IDREFS #IMPLIED>
```

3. XSD (XML Schema Definition)

Se trata de un mecanismo de validación similar a DTD pero que aporta ciertas ventajas:

- Permite evaluar tipos de datos para elementos y atributos.
- Permite concretar el número de veces que aparece un elemento (cardinalidad).

3.1. Sintaxis

Normalmente se define un espacio de normes en su etiqueta raíz para simplificar su uso ("xs" o "xsd", dependiendo de nuestras preferencias).

Opción 1:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
…
</xs:schema>
```

Opción 2:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
…
</xsd:schema>
```

NOTA: "elementFormDefault" indica si todos los elementos tienen o no ("qualified" o "unqualified") que estar certificados por un prefijo del espacio de nombres.

Para incluir un XSD en un XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<concesionario marca="Renault" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="concesionario.xsd">
…
</concesionario>
```

3.1.1. Componentes

Hay 3 tipos de componentes principales:

- **xs:schema** -> raíz

- **xs:element** -> cada elemento del XML
- **xs:attribute** -> atributos de los elementos.

3.1.2. xs:element

Dispone de varios atributos:

- **name**: nombre del elemento.
- **type**: tipo del elemento.
- **minOccurs** y **maxOccurs**: Indica el número mínimo o máximo ocurrencias de un elemento en todo el xml. “unbounded” es ilimitado.
- **ref**: referencia a otros elementos.
- **fixed**: valor fijo.
- **default**: valor por defecto.

Ejemplos:

```
<xs:element name="concesionario">
<xs:element ref="coche" maxOccurs="unbounded"/>
<xs:element name="direccion" type="xs:string" minOccurs="0"/>
```

3.1.3. xs:attribute

Dispone de varios atributos:

- **name**
- **type**
- **fixed**
- **default**
- **use**: Hay distintas opciones: “optional”, “required”, “prohibited”.

Ejemplos:

```
<xs:attribute name="nombre" type="xs:string" use="required"/>
<xs:attribute name="coche" type="xs:IDREF" use="required"/>
```

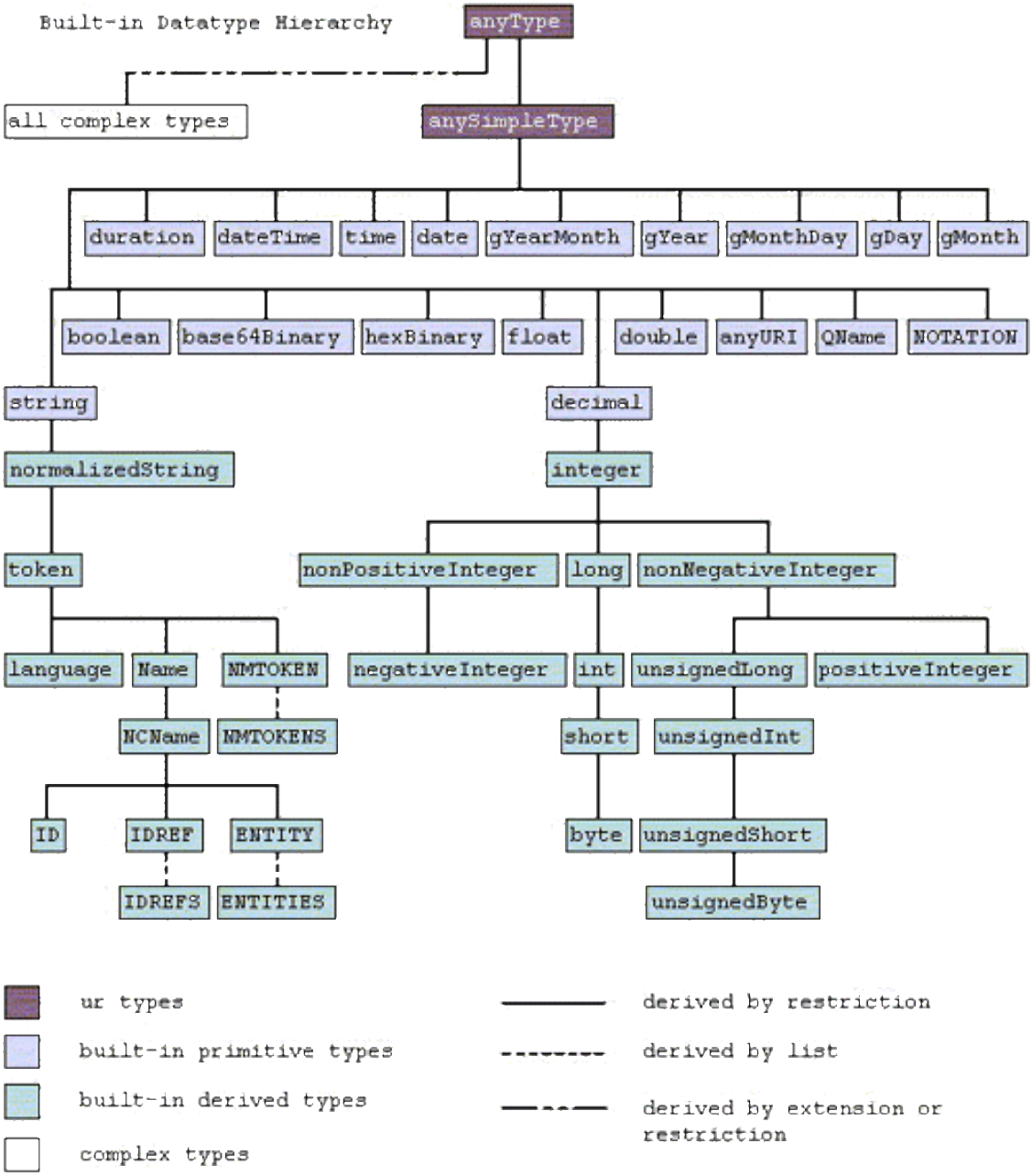
3.2. Tipos de datos

Nos podemos encontrar dos tipos de datos:

- **Predefinidos**
- **Construidos por el usuario**

3.3. Datos predefinidos

Hay 44 tipos de datos, con una jerarquía especial:



Los más comunes son:

- **xs:string**
- **xs:decimal, xs:float, xs:double, xs:integer.**
- **xs:boolean**
- **xs:date, xs:time**

3.4. Construidos por el usuario

En este caso también hay dos tipos:

- **Simples**
- **Complejos**: Compuestos por datos simples y complejos.

3.4.1. Datos simples

Se definen como **xs:simpleType** y se asocian a un tipo predefinido mediante la siguiente sintaxis:

```
<xs:restriction base="xs:string">
```

Dentro de la restricción se limita el tipo.

Ejemplos:

```
<xs:simpleType name="tipourl">
  <xs:restriction base="xs:string">
    <xs:pattern value="https?://.+\.[a-z]{2,3}"></xs:pattern>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="tipoemail">
  <xs:restriction base="xs:string">
     <xs:pattern value="[a-z]+@[^@]+\.(es|com)"></xs:pattern>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="tipoisbn">
   <xs:restriction base="xs:string">
     <xs:pattern value="\d{3}-\d{2}-\d{5}-\d{2}-\d"></xs:pattern>
   </xs:restriction>
</xs:simpleType>
```

Existen muchas restricciones o facetas:

Facetas de XSD	
Faceta	***Descripción***
xs:length	Especifica una longitud fija.
xs:minLength	Especifica una longitud mínima.
xs:maxLength	Especifica una longitud máxima.
xs:pattern	Especifica un patrón de caracteres admitidos.
xs:enumeration	Especifica una lista de valores admitidos.
xs:whiteSpace	Especifica cómo se debe tratar a los posibles espacios en blanco, las tabulaciones, los saltos de línea y los retornos de carro que puedan aparecer.
xs:maxInclusive	Especifica que el valor debe ser menor o igual que el indicado.
xs:maxExclusive	Especifica que el valor debe ser menor que el indicado.
xs:minExclusive	Especifica que el valor debe ser mayor que el indicado.
xs:minInclusive	Especifica que el valor debe ser mayor o igual que el indicado.
xs:totalDigits	Especifica el número máximo de dígitos que puede tener un número.
xs:fractionDigits	Especifica el número máximo de decimales que puede tener un número.

Ejemplo del tipo enumeración:

xs:enumeration

EJEMPLO En el siguiente ejemplo se define un elemento llamado "color" con la restricción de que los únicos valores admitidos son: **"verde"**, **"amarillo"** y **"rojo"**.

```
<xs:element name="color">
   <xs:simpleType>
      <xs:restriction base="xs:string">
         <xs:enumeration value="verde"/>
         <xs:enumeration value="amarillo"/>
         <xs:enumeration value="rojo"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

3.4.2. Datos complejos

Se definen como **xs:complexType** y hay 3 opciones:

- **xs:sequence**: define los hijos de un elemento, en ese orden.
- **xs:choice**: elige solo una de las opciones.
- **xs:all**: aparecen todos en cualquier orden.

Ejemplos:

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

```
    <xs:complexType>
      <xs:sequence>
        <xs:element name="nombre" type="xs:string" />
        <xs:element name="direccion" type="xs:string" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="dni" type="xs:ID" use="required" />
    </xs:complexType>
</xs:element>


<xs:element name="libro">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="titulo" type="xs:string" />
        <xs:element ref="autor" maxOccurs="unbounded"/>
        <xs:element ref="ejemplar" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="isbn" type="tipoisbn" use="required"/>
    </xs:complexType>
</xs:element>
```

Tipos complejos con valor **mixed**:

- **False**: Es el valor por defecto y no permite contenido (ni otros elementos ni contenido): elemento empty.
- **True**: Permite contenido.

Ejemplos:

```
<?xml version="1.0" encoding="UTF-8"?>
<poema fecha="1915-12-01" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="poema.xsd">
   <titulo abr="ea">el alba</titulo>
   <verso>Mi corazon oprimido</verso>
   <verso>late junto a la alborada</verso>
   <verso>el dolor de sus amores...</verso>
</poema>

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

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="poema">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="titulo"/>
        <xs:element name="verso" type="xs:string" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="fecha" type="xs:date" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="titulo">
    <xs:complexType mixed="true">
      <xs:attribute name="abr" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

NOTA: Si en este caso no se pusiera el *mixed* a *true*, fallaría puesto que tiene contenido (“el alba”).

Ejemplo **mixed:false:**

```
<pizza nombre="4quesos">
 <ingrediente nombre="parmesano"/>
 <ingrediente nombre="azul"/>
 <bebida>
  <agua />
 </bebida>
</pizza>
```

Se puede poner así:

```
<xs:element name="agua" >
 <xs:complexType mixed="false"/>
</xs:element>
```

O así: (*mixed:false* es el valor por defecto).

```
<xs:element name="agua" >
        <xs:complexType/>
```

```
</xs:element>
```

Como dentro del último *complexType* no contiene un *xs:attribute*, tampoco se permiten atributos.

Problema: con *mixed* a *true* se permiten contenido de todo tipo, no podemos especificar el tipo de dato que va dentro del nodo. Para solucionarlo, se dispone de **simpleContent**.

En realidad, con los tipos complejos hay 3 casos ***muy especiales***:

- Elementos con atributos, contenido y sin hijos (**simpleContent**):

```
<movil empresa="Azarquiel">666555444</movil>
<movil>666555444</movil>
```

Nota: el *optional* del atributo se indica porque puede no aparecer (segundo móvil)

```
<!--Elemento con atributos, contenido y sin hijos-->
<xs:complexType name="tipoMovil">
    <xs:simpleContent>
        <xs:extension base="tMovil">
            <xs:attribute name="empresa" type="xs:string" use="optional"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
```

- Elementos vacíos y sin atributos (**mixed: false**):

```
<sexo>
    <hombre />
</sexo>
```

```
<!--Elementos vacios y sin atributos-->
<xs:element name="hombre">
    <xs:complexType/>
</xs:element>
```

- Elementos vacíos y con atributos (“**mixed:false”** se puede omitir):

```
<pareja codigo="c1" />
```

```
<!--Elemento vacío y con atributos-->
<xs:element name="pareja">
    <xs:complexType mixed="false">
        <xs:attribute name="codigo" type="xs:IDREF" use="required"/>
    </xs:complexType>
</xs:element>
```

4. XPath

XPath es un lenguaje que permite seleccionar nodos en un documento XML. En otras palabras, permite definir “rutas” en un XML.

Para los ejemplos disponemos de estos XML:

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

<bookstore>

<book>
  <title lang="en">Harry Potter</title>
  <price>29.99</price>
</book>

<book>
  <title lang="en">Learning XML</title>
  <price>39.95</price>
</book>

</bookstore>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<biblioteca>
  <libro>
    <titulo>La vida está en otra parte</titulo>
    <autor>Milan Kundera</autor>
    <fechaPublicacion año="1973"/>
  </libro>
  <libro>
    <titulo>Pantaleón y las visitadoras</titulo>
    <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
    <fechaPublicacion año="1973"/>
  </libro>
  <libro>
    <titulo>Conversación en la catedral</titulo>
    <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
    <fechaPublicacion año="1969"/>
  </libro>
</biblioteca>
```

4.1. Selección de nodos

El eje nos permite seleccionar un subconjunto de nodos del documento y corresponde a recorridos en el árbol del documento, para ello se disponen de las siguientes expresiones:

- **NombreNodo**: Selecciona gtodos los nodos con ese nombre.
- **/** : Selecciona el nodo raíz. Permite seleccionar rutas absolutas.
- **//elemento**: Selecciona todos los nodos “elemento” por debajo de la raíz. Rutas relativas.
- Punto (**.**): Nodo actual.
- Dos puntos(**..**): Nodo raíz.
- *: Cualquier nodo.
- @**atributo**: El atributo con nombre “atributo”.
- **|** : Permite indicar varios recorridos.
- **/node()**: selecciona todos los hijos (elementos o texto) del nodo.
- **//node()**: selecciona todos los descendientes, no solo los directos (elementos o texto) del nodo.
- **/text()**: selecciona únicamente el texto contenido en el nodo.
- **//text()**: selecciona únicamente el texto contenido en el nodo y todos sus descendientes.
- **/***: selecciona todos los hijos (sólo elementos) del nodo.
- **//***: selecciona todos los descendientes (sólo elementos) del nodo.
- **/@***: selecciona todos los atributos del nodo.
- **//@***: selecciona todos los atributos de los descendientes del nodo.

Ejemplos:

Path Expression	Result
bookstore	Selects all nodes with the name "bookstore"
/bookstore	Selects the root element bookstore **Note:** If the path starts with a slash (/) it always represents an absolute path to an element!
bookstore/book	Selects all book elements that are children of bookstore
//book	Selects all book elements no matter where they are in the document
bookstore//book	Selects all book elements that are descendant of the bookstore element, no matter where they are under the bookstore element
//@lang	Selects all attributes that are named lang

```
//autor|//titulo
```

```
<titulo>La vida está en otra parte</titulo>
<autor>Milan Kundera</autor>
<titulo>Pantaleón y las visitadoras</titulo>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
<titulo>Conversación en la catedral</titulo>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
```

```
//autor|//titulo|//@año
```

```
<titulo>La vida está en otra parte</titulo>
<autor>Milan Kundera</autor>
 año="1973"
<titulo>Pantaleón y las visitadoras</titulo>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
 año="1973"
<titulo>Conversación en la catedral</titulo>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
 año="1969"
```

```
//libro/node()
```

```
<titulo>La vida está en otra parte</titulo>
<autor>Milan Kundera</autor>
<fechaPublicacion año="1973"/>
<titulo>Pantaleón y las visitadoras</titulo>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
<fechaPublicacion año="1973"/>
<titulo>Conversación en la catedral</titulo>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
<fechaPublicacion año="1969"/>
```

```
//autor/node()
```

```
Milan Kundera
Mario Vargas Llosa
Mario Vargas Llosa
```

```
//libro//node()
```

```
<titulo>La vida está en otra parte</titulo>
La vida está en otra parte
<autor>Milan Kundera</autor>
Milan Kundera
<fechaPublicacion año="1973"/>
<titulo>Pantaleón y las visitadoras</titulo>
Pantaleón y las visitadoras
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
Mario Vargas Llosa
<fechaPublicacion año="1973"/>
<titulo>Conversación en la catedral</titulo>
Conversación en la catedral
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
Mario Vargas Llosa
<fechaPublicacion año="1969"/>
```

```
//@año[.>1970]
```

```
año="1973"
año="1973"
```

```
//autor[.="Mario Vargas Llosa"]
```

```
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
```

```
//libro[autor="Mario Vargas Llosa" and fechaPublicacion/@año="1973"]
```

```
<libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
</libro>
```

```
//libro[autor="Mario Vargas Llosa" or fechaPublicacion/@año="1973"]
```

```
<libro>
  <titulo>La vida está en otra parte</titulo>
  <autor>Milan Kundera</autor>
  <fechaPublicacion año="1973"/>
</libro>
<libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
</libro>
<libro>
  <titulo>Conversación en la catedral</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1969"/>
</libro>
```

```
//autor/text()
```

```
Milan Kundera
Mario Vargas Llosa
Mario Vargas Llosa
```

```
//libro/text()
```

No devuelve nada porque <libro> no contiene texto.

```
//libro//text()
```

```
La vida está en otra parte
Milan Kundera
Pantaleón y las visitadoras
Mario Vargas Llosa
Conversación en la catedral
Mario Vargas Llosa
```

```
/biblioteca/*
```

```
<libro>
  <titulo>La vida está en otra parte</titulo>
  <autor>Milan Kundera</autor>
  <fechaPublicacion año="1973"/>
</libro>
<libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
</libro>
<libro>
  <titulo>Conversación en la catedral</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1969"/>
</libro>
```

```
//autor/*
```

No devuelve nada porque <autor> sólo contiene texto.

```
/biblioteca//*
```

```
<libro>
  <titulo>La vida está en otra parte</titulo>
  <autor>Milan Kundera</autor>
  <fechaPublicacion año="1973"/>
</libro>
<titulo>La vida está en otra parte</titulo>
<autor>Milan Kundera</autor>
<fechaPublicacion año="1973"/>
<libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
</libro>
<titulo>Pantaleón y las visitadoras</titulo>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
<fechaPublicacion año="1973"/>
<libro>
  <titulo>Conversación en la catedral</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1969"/>
</libro>
<titulo>Conversación en la catedral</titulo>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
<fechaPublicacion año="1969"/>
```

```
//@*
```

```
año="1973"
fechaNacimiento="28/03/1936"
año="1973"
fechaNacimiento="28/03/1936"
año="1969"
```

```
//libro/@*
```

No devuelve nada porque <libro> no tiene atributos.

```
//autor/@*
```

```
fechaNacimiento="28/03/1936"
fechaNacimiento="28/03/1936"
```

4.2. Predicados

El predicado se escribe entre corchetes, a continuación del eje. Si el eje ha seleccionado unos nodos, el predicado permite restringir esa selección a los que cumplan determinadas condiciones.

- **[@atributo]:** selecciona los elementos que tienen el atributo.
- **[número]:** si hay varios resultados selecciona uno de ellos por número de orden; **last()** selecciona el último de ellos
- **[condicion]:** selecciona los nodos que cumplen la condición. Hay diferentes operadores:
  - **operadores lógicos**: and, or, not()
  - **operadores aritméticos**: +, -, *, div, mod
  - **operadores de comparación**: =, !=, <, >, <=, >=
- Para hacer referencia al **propio valor** del elemento seleccionado se utiliza el **punto** (**.**).
- Un predicado puede contener condiciones compuestas: **AND y OR**.

Path Expression	Result
/bookstore/book[1]	Selects the first book element that is the child of the bookstore element. **Note:** In IE 5,6,7,8,9 first node is[0], but according to W3C, it is [1]. To solve this problem in IE, set the SelectionLanguage to XPath: *In JavaScript: xml*.setProperty("SelectionLanguage","XPath");
/bookstore/book[last()]	Selects the last book element that is the child of the bookstore element
/bookstore/book[last()-1]	Selects the last but one book element that is the child of the bookstore element
/bookstore/book[position()<3]	Selects the first two book elements that are children of the bookstore element
//title[@lang]	Selects all the title elements that have an attribute named lang
//title[@lang='en']	Selects all the title elements that have a "lang" attribute with a value of "en"
/bookstore/book[price>35.00]	Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00
/bookstore/book[price>35.00]/title	Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00

5. XQuery

Permite realizar consultas dentro de un XML. Sería similar a SQL. Ambos están soportados por la mayoría de SGBD y XML.

Utiliza expresiones XPath.

Para hacer pruebas:

http://www.xpathtester.com/xquery

XML de ejemplo:

```
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">XQuery Kick Start</title>
    <author>James McGovern</author>
    <author>Per Bothner</author>
    <author>Kurt Cagle</author>
    <author>James Linn</author>
    <author>Vaidyanathan Nagarajan</author>
    <year>2003</year>
    <price>49.99</price>
  </book>
  <book category="web" cover="paperback">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

5.1. Selección de nodos

Para seleccionar nodos primero se debe seleccionar el documento XML y luego ir navegando por los nodos con XPath:

```
doc("ruta fichero XML") /nodos[@expresión]
```

Ejemplo:

```
doc("books.xml") /bookstore/book[price<30]
```

5.2. Funciones, operadores y tipos de datos

XQuery soporta las mismas **funciones y operadores que XPaths**.

XQuery soporta los mismos **tipos de datos que XSD**.

5.3. Expresiones condicionales

IF-THEN-ELSE

```
for $x in doc("books.xml")/bookstore/book
return if ($x/@category="children")
then <child>{data($x/title)}</child>
else <adult>{data($x/title)}</adult>
```

Devuelve el título del hijo, si la categoría son hijos, en caso contrario, devuelve el título del padre:

```
<adult>Everyday Italian</adult>
<child>Harry Potter</child>
<adult>XQuery Kick Start</adult>
<adult>Learning XML</adult>
```

5.4. FLWOR

Permite realizar consultas e interactuar con ellas. El nombre proviene de las operaciones que puede llevar a cabo: "For, Let, Where, Order by, Return".

```
for $x in doc("books.xml")/bookstore/book
where $x/price>30
return $x/title
```

Busca todos los libros cuyo precio sea mayor que 30 y devuelve el título:

```
<title lang="en">XQuery Kick Start</title>
<title lang="en">Learning XML</title>
```

También podríamos ordenar los resultados:

```
for $x in doc("books.xml")/bookstore/book
where $x/price>30
order by $x/title
return $x/title
```

```
<title lang="en">Learning XML</title>
<title lang="en">XQuery Kick Start</title>
```

En el order se podría indicar "**ascending**" (valor por defecto) o "**descending**".

Let permite asignar valores a una variable:

```
for $x in doc("books.xml")/bookstore/book
let $precio := $x/price
return $precio
```

6. XSL

Permite agrupar lenguajes basados en XML:

- **XPath**
- **XSL-FO**: Da formato a los datos.
- **XSLT**: Tranforma un XML en otros documentos.

6.1. Hoja de estilos XSL

Para realizar una transformación se utiliza una hoja de estilos .xsl, que junto con el XML y un procesador XSLT da como resultado un documento nuevo.

Ejemplo de plantilla xsl

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE xsl:stylesheet  [
   <!ENTITY nbsp   " ">
   <!ENTITY copy   "&#169;">
   <!ENTITY reg    "&#174;">
   <!ENTITY trade  "&#8482;">
   <!ENTITY mdash  "&#8212;">
   <!ENTITY ldquo  "&#8220;">
   <!ENTITY rdquo  "&#8221;">
   <!ENTITY pound  "&#163;">
   <!ENTITY yen    "&#165;">
   <!ENTITY euro   "&#8364;">
]>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="utf-8" doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"/>
<xsl:template match="/">

<html>

<head>
   <meta charset="UTF-8" />
   <title>Título página</title>
   <link rel="stylesheet" type="text/css" href="mycss.css" />
</head>

<body>
```

….

```
</xsl:template>
</xsl:stylesheet>
```

Y en el fichero XML habría que asociar esta plantilla:

```
<?xml version="1.0"  encoding="UTF-8" ?>
<?xml-stylesheet type="text/xsl" href="estilo.xsl"?>
<library>
```

….

NOTA: La siguiente condición indica la ruta de partida del XML, dentro de la hoja de estilos xls (xPath):

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

…

</xsl:template>
```

En este caso se indica que va a coger el XML desde el principio (desde library).

6.2. Sentencias más comuness

6.2.1. value-of select

Para obtener un dato del XML. Ejemplo:

```
<p><xsl:value-of select="escrutinio/escanios" /></p>
```

6.2.2. if test

Permite definir condiciones. Ejemplo:

```
<xsl:if test="comment/@rating!="">
  <img src="img/stars-{comment/@rating}.gif" alt="star" />
</xsl:if>
```

Como se puede comprobar, utiliza XPath para obtener el atributo rating de los comentarios.

6.2.3. foreach select

Define un bucle que se ejecutará el número de veces que aparezca el elemento indicado en la select. Ejemplo:

```
<xsl:for-each select="edition">
 <div class="edicion">
  <div class="tedicion">
   <xsl:value-of select="title" />
  </div>
  <div class="dedicion">
   <xsl:value-of select="publisher" />,
   <xsl:value-of select="edition" />,
   <xsl:value-of select="year" />
  </div>
  <p class="derecha">ISBN: <xsl:value-of select="isbn" /></p>
 </div>
 <hr />
</xsl:for-each>
```

6.2.4. choose

Funciona como un selector multiple (switch). Ejemplo:

```
<xsl:choose>
        <xsl:when test="goleslocal &lt; golesvisitante">
                <td class="byn">1</td>
        </xsl:when>
        <xsl:when test="goleslocal=golesvisitante">
       <td class="byn">X</td>
        </xsl:when>
        <xsl:otherwise>
             <td class="byn">2</td>
        </xsl:otherwise>
</xsl:choose>
```

6.2.5. Sort

Permiten ordenar los datos mostrados por el bucle for.

```
<xsl:sort select="XPath dato a ordenar" data-type="number/text" order="descending/ascending" />
```

La línea anterior debe estar colocada inmediatamente detrás de la etiqueta xsl:for-each

```
<xsl:for-each select="catalog/cd">
<xsl:sort select="artist"/>
<tr>
 <td><xsl:value-of select="title"/></td>
```

```
 <td><xsl:value-of select="artist"/></td>
</tr>
</xsl:for-each>
```

6.2.6. Dato XML en un atributo

Se deben utilizar llaves para incrustar un dato del XML dentro de un atributo HTML.

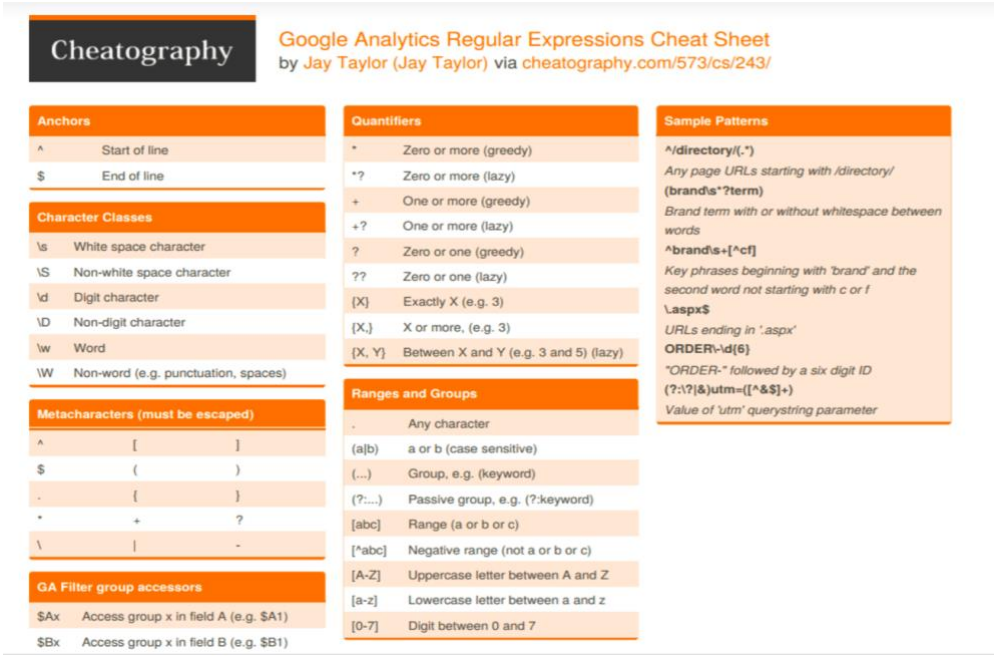
```
<img src="img/{partidos/partido/local}/.gif"/>
```

Si se quiere obtener el propio elemento, operador punto (predicado XPath):

```
<xsl:for-each select="types/type">
        <img src="img/{.}_type.png" />
</xsl:for-each>
```

7. ANEXO: Uso de patrones

Símbolos	*Significado*
`.`	Cualquier carácter.
`\d`	Cualquier dígito del **0** al **9**.
`\D`	Cualquier carácter que no sea un dígito del **0** al **9**.
`x*`	**x** puede aparecer cero o más veces.
`x+`	**x** debe aparecer al menos una vez.
`x?`	**x** puede aparecer una vez o no aparecer.
`[abc]` o `[a|b|c]`	Cualquier carácter indicado entre los corchetes: **a**, **b** o **c**.
`[a-z]`	Cualquier carácter de la **a** a la **z**.
`x{n}`	**x** debe aparecer **n** veces.
`x{n,m}`	**x** debe aparecer entre **n** y **m** veces.
`x{n,}`	**x** debe aparecer al menos **n** veces.



Cheatography

Google Analytics Regular Expressions Cheat Sheet

by Jay Taylor (Jay Taylor) via cheatography.com/573/cs/243/

Anchors

^	Start of line
$	End of line

Character Classes

\s	White space character
\S	Non-white space character
\d	Digit character
\D	Non-digit character
\w	Word
\W	Non-word (e.g. punctuation, spaces)

Metacharacters (must be escaped)

^	[	]
$	(	)
.	{	}
*	+	?
\	|	-

GA Filter group accessors

$Ax	Access group x in field A (e.g. $A1)
$Bx	Access group x in field B (e.g. $B1)

Quantifiers

*	Zero or more (greedy)
*?	Zero or more (lazy)
+	One or more (greedy)
+?	One or more (lazy)
?	Zero or one (greedy)
??	Zero or one (lazy)
{X}	Exactly X (e.g. 3)
{X,}	X or more, (e.g. 3)
{X, Y}	Between X and Y (e.g. 3 and 5) (lazy)

Ranges and Groups

.	Any character
(a|b)	a or b (case sensitive)
(...)	Group, e.g. (keyword)
(?:...)	Passive group, e.g. (?:keyword)
[abc]	Range (a or b or c)
[^abc]	Negative range (not a or b or c)
[A-Z]	Uppercase letter between A and Z
[a-z]	Lowercase letter between a and z
[0-7]	Digit between 0 and 7

Sample Patterns

^/directory/(.*)
Any page URLs starting with /directory/
(brand\s*?term)
Brand term with or without whitespace between words
^brand\s+[^cf]
Key phrases beginning with 'brand' and the second word not starting with c or f
\.aspx$
URLs ending in '.aspx'
ORDER\-\d{6}
"ORDER-" followed by a six digit ID
(?:\?|&)utm=([^&$]+)
Value of 'utm' querystring parameter