XML
(and other data interchange formats)

2024/25 Q2

Jaime Delgado *

DAC – UPC

* Part of this material comes from other sources.

But before, let’s remember HTML

HTML
(HyperText Markup Language)

- Language to express the WWW documents
  - "Markup" language
    (as SGML, *Standard Generalized Markup Language*)
    manera para indicar cómo imprimir una página del mismo modo que HTTP está para indicar cómo se visualiza
  - World Wide Web Consortium (W3C) http://www.w3.org
    estándar de esta organización
- Characteristics:
  - Based on tags: <tag> … </tag>
  - **Logic** structure coding …
  - but also **presentation**!
  - *Logical structure vs. Layout/Physical structure*
  - Links to other objects
    (value or "inline" / URL reference)

Basic concepts

- Tags:
  - Separate text/data fragments
  - Provide separated text with “semantics”
  - In general, there is a *start* and an *end* (exceptions exist)

  - Start of area delimited by a tag: `<tag_name>`
  - End of area delimited by a tag: `</tag_name>`

  - Example: `<tag_name>` delimited text `</tag_name>`

- Attributes:
  - Complete the semantics of a tag

  - Form: `<tag attrib1=`“value” `attrib2=`“value”> text `</tag>`

HTML (v.4): tags (HTML5 later on)

<!-->
<
<A>
<ABBREV>
<ACRONYM>
<ADDRESS>
<APPLET>
<AREA>
<AU>
<AUTHOR>
<B>
<BANNER>
<BASE>
<BASEFONT>
<BGSOUND>
<BIG>
<BLINK>
<BLOCKQUOTE>
<BQ>
<BODY>

<CAPTION>
<CENTER>
<CITE>
<CODE>
<COL>
<COLGROUP>
<CREDIT>
<DEL>
<DFN>
<DIR>
<DIV>
<DL>
<DT>
<DD>
<EM>
<EMBED>
<FIG>
<FN>
<FONT>
<FORM>
<FRAME>
<FRAMESET>
<H1>
<H2>
<H3>
<H4>
<H5>
<H6>
<HEAD>
<HR>
<HTML>
<I>
<IFRAME>
<IMG>
<INPUT>
<INS>
<ISINDEX>
<KBD>
<LANG>
<LH>
<LI>
<LINK>
<LISTING>
<MAP>
<MARQUEE>
<MATH>
<MENU>
<META>
<MULTICOL>
<NOBR>
<NOFRAMES>
<NOTE>
<OL>
<OVERLAY>
<P>
<PARAM>
<PERSON>
<PLAINTEXT>
<PRE>
<Q>
<RANGE>
<SAMP>
<SCRIPT>
<SELECT>
<SMALL>
<SPACER>
<SPOT>
<STRIKE>
<STRONG>
<SUB>
<SUP>
<TAB>
<TABLE>
<TBODY>
<TD>
<TEXTAREA>
<TEXTFLOW>
<TFOOT>
<TH>
<THEAD>
<TITLE>
<TR>
<TT>
<U>
<UL>
<VAR>
<WBR>
<XMP>

HTML (v.4): tags (HTML5 later on)

<!-->
<
<A> link
<ABBREV>
<ACRONYM>
<ADDRESS>
<APPLET>
<AREA>
<AU>
<AUTHOR>
<B>
<BANNER>
<BASE>
<BASEFONT>
<BGSOUND>
<BIG>
<BLINK>
<BLOCKQUOTE>
<BQ>
<BODY>

<CAPTION>
<CENTER>
<CITE>
<CODE>
<COL>
<COLGROUP>
<CREDIT>
<DEL>
<DFN>
<DIR>
<DIV>
<DL>
<DT>
<DD>
<EM>
<EMBED>
<FIG>
<FN>
<FONT>
<FORM>
<FRAME />
<FRAMESET>
<H1> header 1
<H2>
<H3>
<H4>
<H5>
<H6>
<HEAD>
<HR>
<HTML>
<I>
<IFRAME>
<IMG> image
<INPUT />
<INS>
<ISINDEX>
<KBD>
<LANG>
<LH>
<LI> element of a list
<LINK>
<LISTING>
<MAP>
<MARQUEE>
<MATH>
<MENU>
<META />
<MULTICOL>
<NOBR>
<NOFRAMES>
<NOTE>
<OL> ordered list
<OVERLAY>
<P>
<PARAM>
<PERSON>
<PLAINTEXT>
<PRE>
<Q>
<RANGE>
<SAMP>
<SCRIPT>
<SELECT>
<SMALL>
<SPACER>
<SPOT>
<STRIKE>
<STRONG>
<SUB>
<SUP>
<TAB>
<TABLE>
<TBODY>
<TD>
<TEXTAREA>
<TEXTFLOW>
<TFOOT>
<TH>
<THEAD>
<TITLE>
<TR>
<TT>
<U>
<UL> non-ordered list
<VAR>
<WBR>
<XMP>

XML

Define un método para que tú mismo te definas tus propias etiquetas, por eso se llama extensible.

- XML: eXtensible Markup Language
- XML 1.0 (5th Ed., 2008), https://www.w3.org/TR/xml/
- XML 1.1 (2nd Ed., 2006) (for better Unicode handling)

- Designed to interchange and store data (HTML to *display* data)

XML es como una base de datos, describe cómo es la estructura de datos.

- XML
  - To describe information structures → Process them automatically with applications
  - "Users" must define their own **tags**
    - "**Users**": "Private" users and SDO ("Standards Development Organizations")

XML structure & syntax

- XML:
  - Tree structure
  - Elements, attributes & text
  - Example:

```
<book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    …
  </book>
```

XML structure & syntax

- XML:
  - Tree structure
  - Elements, attributes & text
  - Example:

```
<book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    …
</book>
```

XML structure & syntax

- First line (example):
  `<?xml version="1.0" encoding="ISO-8859-1"?>`

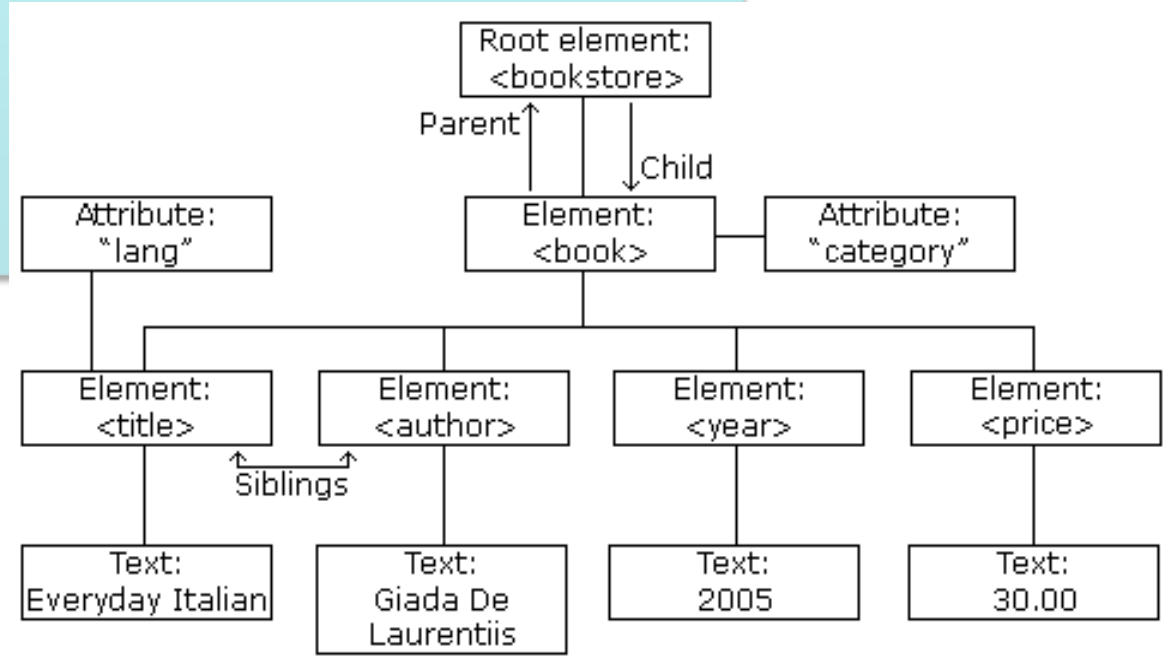
- XML simple syntax:
  - Closing tag mandatory
  - Tags case sensitive
  - Elements could be nested:
    `<a> <b>…</b> <c>…</c> </a>`
    (`a` parent, `b`, `c` childs , `b`, `c` siblings)
  - Root element needed
  - Attribute values must be quoted
  - Comments: `<!-- … -->`

XML simple example

```
<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>
...
</bookstore>
```

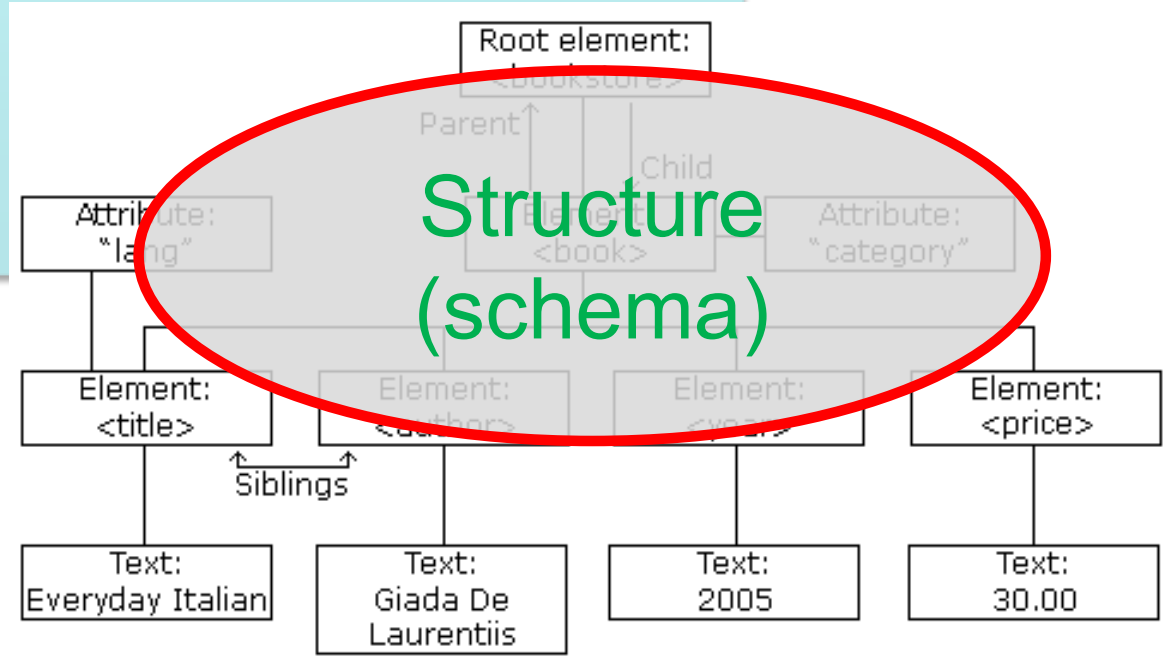
XML simple example

```
<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>
...
</bookstore>
```



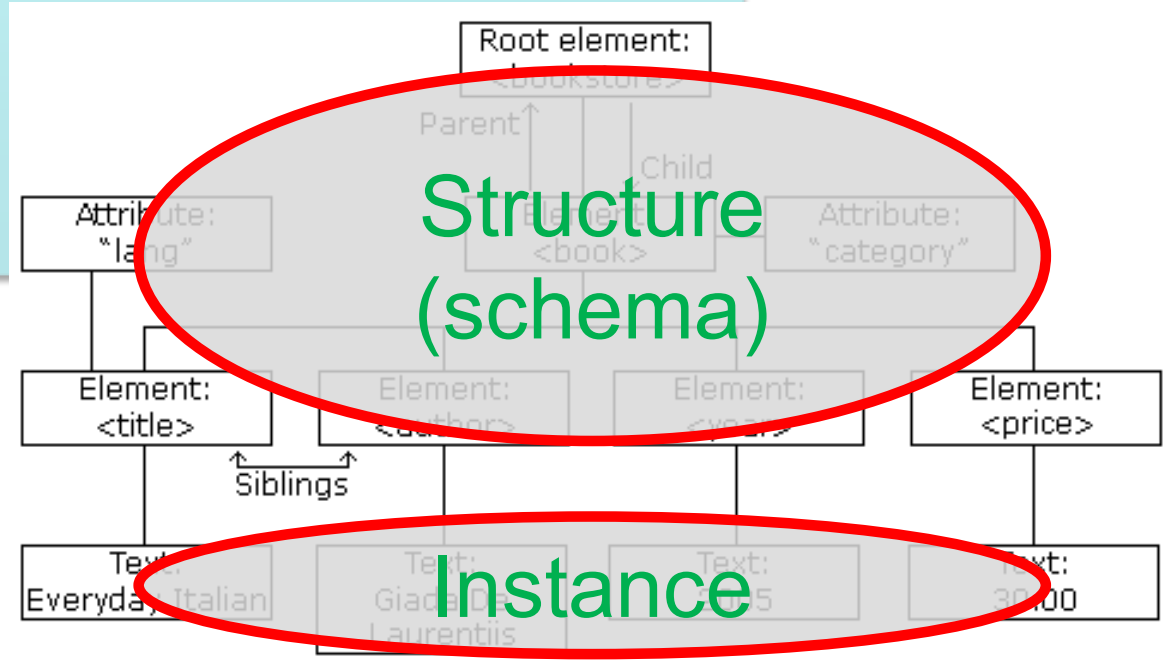
XML simple example

```
<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>
...
</bookstore>
```



XML simple example

```
<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>
...
</bookstore>
```



Related tools: XPath example

```
<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>
...
</bookstore>
```

Title of the first book of the bookstore:

/bookstore/book[1]/title

XML main issues

- Attributes vs. Elements: *Design decision*
- Name conflicts:
  - *Namespaces*
    - To differentiate element names defined by different developers/standards
  - xmlns attribute (in the start tag of an element):

    `xmlns:prefix="URI"` → identificador único.

    - URLs often used to define "unique" namespaces
- How to define tags and "structure": Schemas
  (specify the valid structure (grammar) of a set of XML documents (a XML application))
  - Examples …

XML: Idea of Schema

- XML Schema Definition, XSD
  - Content of the file "note.xsd":

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="note">
    <xs:complexType>
      <xs:sequence>
            <xs:element name="to" type="xs:string"/>
            <xs:element name="from" type="xs:string"/>
            <xs:element name="heading" type="xs:string"/>
            <xs:element name="body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

http://www.w3schools.com/xml/

Root element of the schema + namespace where the schema is defined, the namespace should be prefixed xs.

namespace de www.w3.org. cualquiera nombre se puede usar

root element for the instances

complexType: contains other elements

sequence: child elements must appear in the same order

- Reference to the XSD defined in "note.xsd":

estándar

```
<?xml version="1.0"?>
<note xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="note.xsd">
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Para ligar una instancia con su esquema tenemos que usar schemalocation

XSD schema defined in location "note.xsd"

Another example of Schema

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.films.org"
        xmlns="http://www.films.org">
  <xsd:element name="films">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="film" type="filmType" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="filmType">
        <xsd:sequence>
                <xsd:element name="title" type="xsd:string"/>
                <xsd:element name="genre" type="xsd:string"/>
                <xsd:element name="year" type="xsd:string"/>
        </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Another example of Schema

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.films.org"
        xmlns="http://www.films.org">
   <xsd:element name="films">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="film" type="filmType" maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   <xsd:complexType name="filmType">
            <xsd:sequence>
                        <xsd:element name="title" type="xsd:string"/>
                        <xsd:element name="genre" type="xsd:string"/>
                        <xsd:element name="year" type="xsd:string"/>
            </xsd:sequence>
   </xsd:complexType>
</xsd:schema>
```

NameSpace defined in this XML document

Default NameSpace

Example of instance

```
<?xml version="1.0"?>
<films xmlns="http://www.films.org"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation= "http://www.films.org films.xsd">
      <film>
            <title>The Sting</Title>
            <genre>Crime</genre>
            <year>1973</year>
     </film>
</films>
```

XML Schema & namespaces

```
<films xmlns="http://www.films.org“                                  (1)
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance“   (2)
       xsi:schemaLocation="http://www.films.org films.xsd">        (3)
```

1) Names without prefix belong to the *default* namespace “http://www.films.org”

2) Names with prefix “xsi” (only the schemaLocation) belong to the namespace “http://www.w3.org/2001/XMLSchema-instance“

3) The schema corresponding to the namespace “http://www.films.org” is in file “films.xsd”

XML Schema: Types

no entra en el examen

Simple Types (Datatypes) – Primitive

string	*any Unicode string*
boolean	true, false, 1, 0
decimal	3.1415
float	6.02214199E23
double	42E970
dateTime	2004-09-26T16:29:00-05:00
time	16:29:00-05:00
date	2004-09-26
hexBinary	48656c6c6f0a
base64Binary	SGVsbG8K
anyURI	http://www.brics.dk/ixwt/
QName	rcp:recipe, recipe
...	

XML Schema: Types

Derivation of Simple Types – Restriction

Constraining facets:

- `length`
- `minLength`
- `maxLength`
- `pattern`
- `enumeration`
- `whiteSpace`
- `maxInclusive`
- `maxExclusive`
- `minInclusive`
- `minExclusive`
- `totalDigits`
- `fractionDigits`

XML Schema: Types

Examples

```
<simpleType name="score_from_0_to_100">
  <restriction base="integer">
    <minInclusive value="0"/>
    <maxInclusive value="100"/>
  </restriction>
</simpleType>

<simpleType name="percentage">
  <restriction base="string">
    <pattern value="([0-9]|[1-9][0-9]|100)%"/>
  </restriction>
</simpleType>
```

regular expression

XML Schema: Types

Built-In Derived Simple Types

- normalizedString
- token
- language
- Name
- NCName
- ID
- IDREF
- integer
- nonNegativeInteger
- unsignedLong
- long
- int
- short
- byte
- ...

XML Schema: Types

Complex Types with Complex Contents

- Content models as **regular expressions**:
  - Element reference `<element ref="name"/>`
  - Concatenation `<sequence> ... </sequence>`
  - Union `<choice> ... </choice>`
  - All `<all> ... </all>`
  - Element wildcard: `<any namespace="..." processContents="..."/>`
- Attribute reference: `<attribute ref="..."/>`
- Attribute wildcard: `<anyAttribute namespace="..." processContents="..."/>`

Cardinalities: `minOccurs, maxOccurs, use`

Mixed content: `mixed="true"`

XML Schema: Types

Example

```
<element name="order" type="n:order_type"/>

<complexType name="order_type" mixed="true">
  <choice>
    <element ref="n:address"/>
    <sequence>
      <element ref="n:email"
               minOccurs="0" maxOccurs="unbounded"/>
      <element ref="n:phone"/>
    </sequence>
  </choice>
  <attribute ref="n:id" use="required"/>
</complexType>
```

XML Schema: Types

Complex Types with Simple Content

```
<complexType name="category">
  <simpleContent>
    <extension base="integer">
      <attribute ref="r:class"/>
    </extension>
  </simpleContent>
</complexType>
```

```
<complexType name="extended_category">
  <simpleContent>
    <extension base="n:category">
      <attribute ref="r:kind"/>
    </extension>
  </simpleContent>
</complexType>
```

```
<complexType name="restricted_category">
  <simpleContent>
    <restriction base="n:category">
      <totalDigits value="3"/>
      <attribute ref="r:class" use="required"/>
    </restriction>
  </simpleContent>
</complexType>
```

XML Schema: Types

Derivation with Complex Content

```
<complexType name="basic_card_type">
 <sequence>
  <element ref="b:name"/>
 </sequence>
</complexType>
```

```
<complexType name="extended_type">
 <complexContent>
  <extension base=
             "b:basic_card_type">
   <sequence>
    <element ref="b:title"/>
    <element ref="b:email"
             minOccurs="0"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

```
<complexType name="further_derived">
 <complexContent>
  <restriction base=
               "b:extended_type">
   <sequence>
    <element ref="b:name"/>
    <element ref="b:title"/>
    <element ref="b:email"/>
   </sequence>
  </restriction>
 </complexContent>
</complexType>
```

Note:* `restriction` *is* *not* *the opposite of* `extension`*!

XML Schema: Types

Global vs. Local Descriptions

Global (toplevel) style:

```
<element name="card"
         type="b:card_type"/>
<element name="name"
         type="string"/>

<complexType name="card_type">
  <sequence>
    <element ref="b:name"/>
    ...
  </sequence>
</complexType>
```

Local (inlined) style:

```
<element name="card">
  <complexType>
    <sequence>
      <element name="name"
               type="string"/>
      ...
    </sequence>
  </complexType>
</element>
```

inlined

Vuelve a entrar en el examen

eXtensible Stylesheet Language, XSL

- Main component: XSL Transformations, XSLT
- XSLT: Programming language for specifying transformations XML <--> other target language (e.g. HTML, another XML schema)
- All major browsers support XML/XSLT
- XSL style sheet: one or more rules (“templates”)
- Templates are applied when a specified node is matched

XML: Idea XSL - XSLT

- XSL Transformations, XLST

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
  </cd>
-
</catalog>
```

?

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore

Source: http://www.w3schools.com/xml/

XML: Idea XSL - XSLT

- XSL Transformations, XLST

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
  </cd>
-
</catalog>
```

reference an XLST "cdcatalog.xsl" in an XML document

- Content of the file "cdcatalog.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>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Title</th>
      <th>Artist</th>
    </tr>
    <xsl:for-each select="catalog/cd">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
    </tr>
    </xsl:for-each>
  </table>
  </body></html>
</xsl:template>
</xsl:stylesheet>
```

se pasa a HTML para que se pueda ver en browser

defines a template. Attibute match specifies the nodes using XPath

select every element of a node-set

extract the value of an XML element

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore

Source: http://www.w3schools.com/xml/

Other data interchange formats:

- JSON

JSON

mismo estándar publicado en 3 sitios distintos

- JSON: "**JSON** (JavaScript Object Notation)"
- RFC 8259 (2017): *JSON Data Interchange Format* (1st RFC in 2006)
- ECMA-404 (identical to RFC 8259)
  → ISO/IEC 21778:2017 ("Fast-track procedure")
- lightweight, text-based, language-independent data interchange format
- Elements:
  - 4 primitive types (strings, numbers, booleans, null)
  - 2 structured types (objects and arrays)

Conclusión:
- Estructura simple es mejor JSON, XML sólo crea overhead.
- Estructura compleja, es mejor XML.

JSON

- Example:

```
{
  "Image": {
    "Width": 800,
    "Height": 600,
    "Title": "View from 15th Floor",
    "Thumbnail": {
      "Url": http://www.example.com/image/481989943",
      "Height": 125,
      "Width": 100
    },
    "Animated" : false,
    "IDs": [116, 943, 234, 38793]
  }
}
```

JSON

- JSON: "**JSON** (JavaScript Object Notation)"
- lightweight, text-based, language-independent data interchange format
- Elements:
  - 4 primitive types (strings, numbers, booleans, null)
  - 2 structured types (objects and arrays)
- JSON Schema:

En JSON tener el esquema no es mandatorio, pero es recomendado para garantizar que la automatización funcione correctamente.

  - IETF: - Internet Drafts going on
    - JSON Type Definition (RCF 8927, Nov 2020)
  - But: https://json-schema.org/ (Dec 2020) (implementations available)