Welcome

Summer School
Digital Tools for Humanists

Pisa – June 3-12 2024

What is XML?

- The acronym means **eXtensible Markup Language**
- It is used to describe “data” in a way which is simple, structured and (usually) readable also by humans
- Developed at the end of the nineties by W3C (World Wide Web Consortium) as a simplification of the language SGML (Standard Generalized Markup Language), from which also HTML was derived
  - SGML originated from IBM’s GML and is still today an ISO standard (ISO 8879:1986)
- Initial objective was to define a standard language for exchanging information on the Web

Representing documents

- Plain text
  - No information about structure
  - Different representation for line breaks
    - Windows represent a new line with the sequence “carriage return” followed by “line feed”
    - Unix and Apple/Mac represent a new line with “line feed” only
- Page description languages
  - PostScript
  - PDF – Portable Document Format
- Word processors (text editors)
  - ODF – Open Document Format
  - RTF – Rich Text Format
  - Microsoft Word
  - LaTeX

Knowledge of internal representation needed to extract text

Text editors

Editing of the contents

Editing of the format

- Mark-up languages
- WYSIWYG (What You See Is What You Get)

Simple example

- Define the structure of a (simple) telephone book:
  - *Name*; *Surname; Telephon*.
- Example of a text file (phonebook.txt):

  Mario; Rossi; 031221222
  Francesco; Neri; 123876453
  Michele; Bianchi; 022121222
- Within the computer it is just one long string of characters
  Mario;Rossi;031221222◘Francesco;Neri;123876453◘Michele;Bianchi;022121222◘
- Use of “delimiters” to separate the elements of the data (e.g. “;” and “◘”)
- In this example the data have three elements (name, surname, telephone number)

Simple example in XML

```
<phonebook>
        <entry>
                <firstname>Mario</firstname>
                <lastname>Rossi</lastname>
                <phone>031221222</phone>
        </entry>
        <entry>
                <firstname>Francesco</firstname>
                <lastname>Neri</lastname>
                <phone>123876453</phone>
        </entry>
        <entry>
                <firstname>Michele</firstname>
                <lastname>Bianchi</lastname>
                <phone>022121222</phone>
         </entry>
</phonebook >
```

- The delimiters have a name (elements or tags)
- Within the computer it is still a long string of characters, but now a hierarchical structure is becoming apparent, by means of “opening tags” and “closing tags” nested one within the other

XML is not HTML

- The main purpose of XML is to describe how the data is structured and what is the contents of the components of the structure
- XML tags can be freely defined by the user, but....
  - they do not give any “formal” indication about the meaning (the semantics) of the data delimited by the tags
  - they do not give any indication on how the data can be represented, processed, utilized
- XML is only a way to create “languages” (i.e. the names of the tags) with a standard syntax to describe the structure of data
- HTML is just one of those “languages” whose main purpose is to describe how to visualize on a screen some data
- HTML tags (the language) are defined by W3C

Structure of a XML document

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

Prolog

```
<personnel>
        <employee>
                <First>Fred</First>
                <Last>Landis</Last>
                <Title>Project Manager</Title>
                <Phone>123-456-7890</Phone>
                <Email>f.landis@nanonull.com</Email>
        </employee>
        <--This is a comment-->
</personnel>
```

Root

The XML elements

- There must be only one root element
- Tag elements must be closed
- Tag names can be any string of letters and numbers, not containing spaces (blank characters) and not starting with a number or special character
- XML distinguishes small letters and capital letters, e.g. <title> is different from <Title>
- Elements can contain “text” or other elements (can also be void)
- Contained elements must be closed before closing the “container” element

Special characters

Some special characters cannot appear in the value (the “text”) of an element

name	meaning
&	**&**
<	**<**
>	**>**
'	’
"	”

Attributes of an element

- Attributes can be defined in the opening tag, to provide additional information about the element, e.g.
  <actor role="primary">Brad Pitt</actor>
- Whether the "additional information" is provided as an attribute or in a nested element is decided by the designer of the XML definitions

```
<person>
   <name>Andrea</name>
   <surname>Rossi</surname>
   <sex>male</sex>
</person>
```

```
<person sex="male">
   <name>Andrea</name>
   <surname>Rossi</surname>
</person>
```

~~<person surname="Rossi">~~
~~<name>Andrea</name>~~
~~<sex>male</sex>~~
~~</person>~~

Conflicts in tag names

In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications.

```
<table>
  <tr>
    <td>Room1</td>
    <td>Room2</td>
  </tr>
</table>
```

HTML table

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

piece of furniture

Use of prefixes

```
<h:table>
  <h:tr>
    <h:td>Room1</h:td>
    <h:td>Room2</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>
```

This comes from the "namespace" **h**

This comes from the "namespace" **f**

Declaration of namespaces

```
<root
xmlns:h="http://www.w3.org/TR/html4/"
xmlns:f="http://www.w3schools.com/furniture">

<h:table>
  <h:tr>
    <h:td>Room1</h:td>
    <h:td>Room2</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>

</root>
```

Declaration of "namespace" **h**

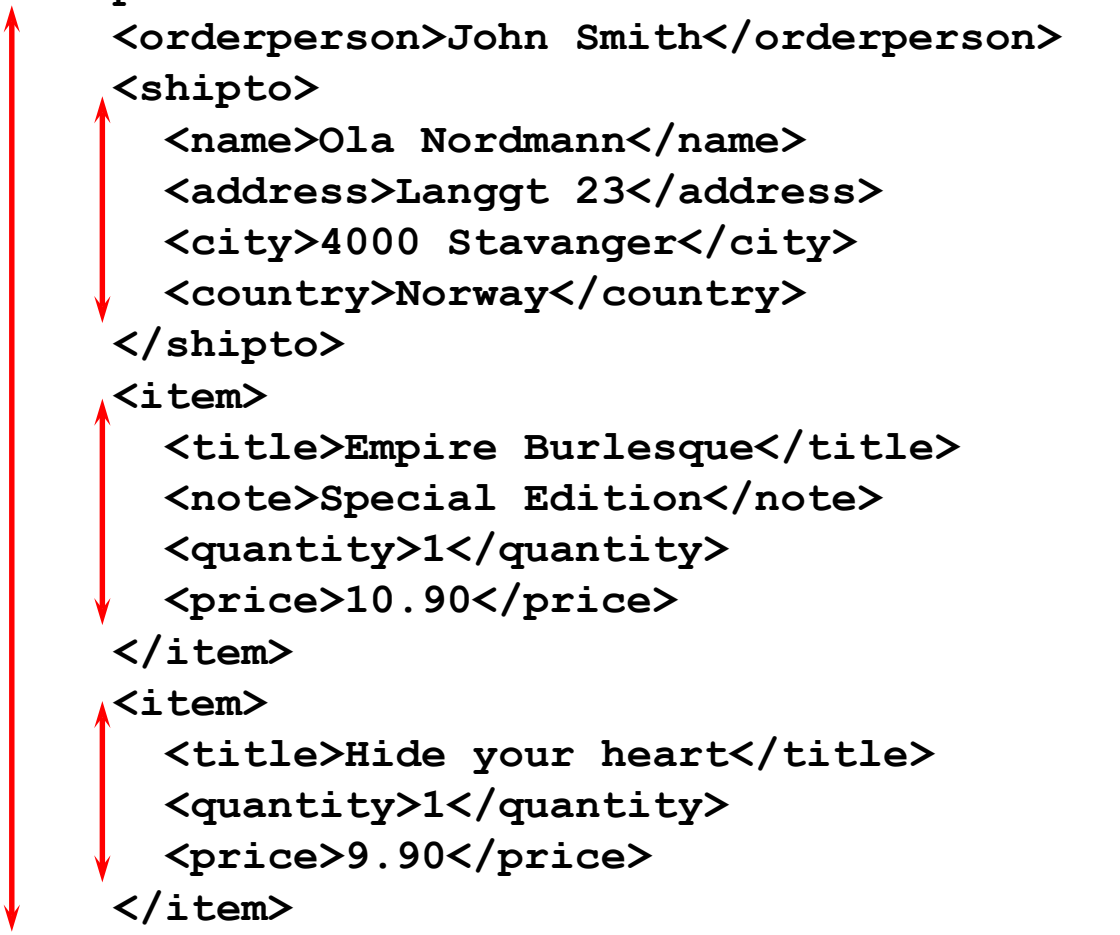
Declaration of "namespace" **f**

Namespaces are "resources" and are identified by their URI and (usually) are specified as attributes of the root

Definition of document types

- Need to define exactly the “abstract” structure of a given XML document, regardless of its actual content
  - Element (tag) names
  - Content of each element (simple or complex)
- DTD: Data Type Definition (deprecated)
  - Uses a syntax different from XML
- XML Schema
  - Uses the same syntax as any other XML document

An XML document



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<shiporder orderid="889923">
    <orderperson>John Smith</orderperson>
    <shipto>
      <name>Ola Nordmann</name>
      <address>Langgt 23</address>
      <city>4000 Stavanger</city>
      <country>Norway</country>
    </shipto>
    <item>
      <title>Empire Burlesque</title>
      <note>Special Edition</note>
      <quantity>1</quantity>
      <price>10.90</price>
    </item>
    <item>
      <title>Hide your heart</title>
      <quantity>1</quantity>
      <price>9.90</price>
    </item>
</shiporder>
```

An XML schema (shiporderschema.xsd)

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="shiporder">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="orderperson" type="xs:string"></element>
      <xs:element name="shipto">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="address" type="xs:string"/>
            <xs:element name="city" type="xs:string"/>
            <xs:element name="country" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="item" ... > .......... </element>
    </xs:sequence>
      ................
</xs:element>
</xs:schema>
```

An XML schema

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="shiporder">
  <xs:complexType>
    <xs:sequence>
      ..................
      <xs:element name="item" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="title" type="xs:string"/>
            <xs:element name="note" type="xs:string" minOccurs="0"/>
            <xs:element name="quantity" type="xs:positiveInteger"/>
            <xs:element name="price" type="xs:decimal"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="orderid" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
</xs:schema>
```

The XML schema

- defines elements that can appear in a document
- defines attributes that can appear in a document
- defines which elements are child elements
- defines the order of child elements
- defines the number of child elements
- defines whether an element is empty or can include text
- defines data types for elements and attributes
- defines default and fixed values for elements and attributes

Namespaces

- A schema is a collection (vocabulary) of element declarations and type definitions
- Each different schema can be assigned a unique name (i.e. a URI) which indicates the “namespace” defined in the schema
- The namespace(s) are useful to “validate” an “instance document”
- The namespaces used in an instance document are (usually) declared as attributes of the root

Target name space

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xs:targetNamespace="http://www.example.com/shiporderschema">
<xs:element name="shiporder">
  <xs:complexType>
    <xs:sequence>
      ................

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

Referencing a schema

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<shiporder orderid="889923"
   xmlns="http://www.example.com/shiporderschema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.example.com/shiporderschema
                       http://www.example.com/shiporderschema.xsd">

    <orderperson>John Smith</orderperson>
    <shipto>
      .............
    </shipto>
    <item>
      .............
    </item>
    <item>
      .............
    </item>
</shiporder>
```

JSON

- JSON stands for **J**ava**S**cript **O**bject **N**otation
- JSON is a **text format** for storing and transporting data
- JSON is "self-describing" and easy to understand

- {"name":"John", "age":30, "city":”New York”}
- This is an “object” with three “properties” (name, age, city)
- Each property has a value
- Syntax is derived from JavaScript object notation syntax

Source: w3schools

JSON Syntax

- Unordered sets of name/value pairs
  - Begins with **{** (left brace)
  - Ends with **}** (right brace)
  - Each name is followed by **:** (colon)
  - Name/value pairs are separated by **,** (comma)

- Arrays in JSON
  - An ordered collection of values
  - Begins with **[** (left bracket)
  - Ends with **]** (right bracket)
  - Name/value pairs are separated by **,** (comma)

JSON and XML

```
<employees>
   <employee>
      <firstName>John</firstName> <lastName>Doe</lastName>
   </employee>
   <employee>
      <firstName>Anna</firstName> <lastName>Smith</lastName>
   </employee>
   <employee>
      <firstName>Peter</firstName> <lastName>Jones</lastName>
   </employee>
</employees>
```

```
{"employees":[
  { "firstName":"John", "lastName":"Doe" },
  { "firstName":"Anna", "lastName":"Smith" },
  { "firstName":"Peter", "lastName":"Jones" }
]}
```

Source: w3schools

JSON and XML

- JSON is Like XML Because
  - Both JSON and XML are "self describing" (human readable)
  - Both JSON and XML are hierarchical (values within values)
  - Both JSON and XML can be parsed and used by lots of programming languages

- JSON is Unlike XML Because
  - JSON doesn't use end tag
  - JSON is shorter
  - JSON is quicker to read and write
  - JSON can use arrays
  - JSON is (almost) a JavaScript object

Source: w3schools