



---

# **Introduction to XML**

**Danilo Montesi**

[danilo.montesi@unibo.it](mailto:danilo.montesi@unibo.it)

<http://www.unibo.it/docenti/danilo.montesi>



# The origins of XML

---

- XML (eXtensible Markup Language) derives from SGML (Standard Generalized Markup Language).
- Both with XML and SGML it is possible to define markup languages specific to several domains, such as finance or math.
- For instance, HTML is one of the languages derived from SGML.
- With respect to SGML, XML is easier to use, and it's designed to specify markup languages to be used on the Internet.
- Initially XML was born as a format for data exchange
- We will look at XML for its data storage and data lookup capabilities.
- Consequently, we will not go through all the in-depth details but only the aspects relevant to data management.



# Example of XML document

```
<?xml version="1.0"?>
<?tex doctype[report] ?>
<doc isbn="2-266-04744-2">
    <!-- editor is missing! -->
    <author>T. Harris</author>
    <title xml:lang="en">The silence of the lambs</title>
    <title xml:lang="fr">Le silence des agneaux</title>
    <comment>
        A book full of <i>suspance</i>.
    </comment>
    <price currency="euro">7</price>
</doc>
```



# Markup and *character data*

```
<?xml version="1.0"?>
<?tex doctype[report] ?>
<doc isbn="2-266-04744-2">
    <!-- editor is missing! -->
    <author>T. Harris</author>
    <title xml:lang="en">The silence of the lambs</title>
    <title xml:lang="fr">Le silence des agneaux</title>
    <comment>
        A book full of <i>suspance</i>.
    </comment>
    <price currency="euro">7</price>
</doc>
```



# Prolog (1/2)

```
<?xml version="1.0"?>
<?tex doctype[report] ?>                                PROLOG
<doc isbn="2-266-04744-2">
    <!-- editor is missing! -->
    <author>T. Harris</author>
    <title xml:lang="en">The silence of the lambs</title>
    <title xml:lang="fr">Le silence des agneaux</title>
    <comment>
        A book full of <i>suspance</i>.
    </comment>
    <price currency="euro">7</price>
</doc>
```



## Prolog (2/2)

---

- The prolog contains information useful for the interpretation of the document.
- In particular, it can contain:
  - A declaration that the document is in XML format ([optional](#)).
  - A *grammar* that allows to validate the content of the document ([optional](#)).
  - Comments and information for software applications that will use the document (Processing Instructions, or PI) ([zero or more](#)).



# Body of the document (1/2)

```
<?xml version="1.0"?>
<?tex doctype[report] ?>
<doc isbn="2-266-04744-2">
    <!-- editor is missing! -->
    <author>T. Harris</author>
    <title xml:lang="en">The silence of the lambs</title>
    <title xml:lang="fr">Le silence des agneaux</title>
    <comment>
        A book full of <i>suspance</i>.
    </comment>
    <price currency="euro">7</price>
</doc>
```

DOCUMENT  
BODY



## Body of the document (2/2)

---

- The body of the document is made of one **element**, which itself can contain other nested elements in its content, and also comments.
- We will see, through some examples, how to write *well-formed* elements.



# Elements

---

- An element can be of two forms:

`<name attributes_list>`      (*extended form*)  
*content...*

`</name>`

`<name attributes_list />`      (*short form*)



# Well-formed elements and attributes (1/5)

- Each element must be contained between an opening **tag** and a closing **/tag** or with a short form.

```
<?xml version="1.0"?>
<?tex doctype[report] ?>
<doc isbn="2-266-04744-2">
    <title xml:lang="en">
        The silence of the lambs
    </title>
    <price euro="7"/>
</doc>
```

<BR>

ERROR:  
this tag (BR) is opened but  
never closed!

# Well-formed elements and attributes (2/5)

- The names of elements and attributes are *case-sensitive*.

ERROR

```
<?xml version="1.0"?>
<?tex doctype[report] ?>
<doc isbn="2-266-04744-2">
    <title xml:lang="en">
        The silence of the lambs
    </title>
    <price euro="7"/>
</DOC>
```



# Well-formed elements and attributes (3/5)

- Elements must be nested correctly, and there is only one element that contains all the other elements.

```
<?xml version="1.0"?>
<?tex doctype[report] ?>
<doc isbn="2-266-04744-2">
    <b><i>Missing book</i></b></i>
</doc>
```

ERROR



# Well-formed elements and attributes (4/5)

- Values of the attributes must be contained between quotes or double quotes.

```
<?xml version="1.0"?>
<?tex doctype[report] ?>
<doc isbn="2-266-04744-2">
    <title xml:lang=en>
        The silence of the lambs
    </title>
    <price euro="7"/>
</doc>
```

ERROR



# Well-formed elements and attributes (5/5)

---

- An element cannot have more than one attribute with the same name.

WRONG

```
<book author="Doe" author="Blake"/>
```

CORRECT

```
<book>
  <author>Doe</author>
  <author>Verdi</author>
</book>
```



# Document Type Declaration

---

- In the beginning of an XML document there can be a **document type declaration**.
- This declaration contains a grammar, named **Document Type Definition**, or **DTD**, with the double purpose of *constrain* and *complete* the documents.
- The DTD is made of **markup declarations**, which define what can be and cannot be written in the related XML document.
- The DTDs determine which elements can be included in the document, how they can be used, what are the default values of the attributes of the elements, and other constraints.



# Valid (and well-formed) documents

---

- An XML document is valid if:
  - It contains a DTD
  - It complies to it

Valid documents

Well-formed document

The following is a well-formed document but it's not valid,  
because it doesn't have a DTD :

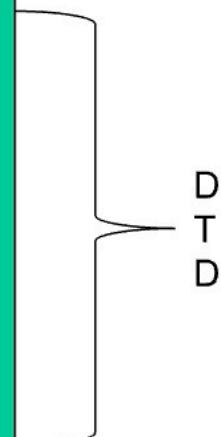
```
<greetings>Hello, world!</greetings>
```



# Examples of Document Type Declaration 1/2

- We define a type of document named *greetings* that contains an element <greetings>, which do not contain any other element. Then, the attributes of <greetings> are declared, in this example only the *id* attribute.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE greetings [
    <!ELEMENT greetings (#PCDATA)>
    <!ATTLIST greetings
        id          ID      #REQUIRED>
]>
<greetings id="0001">Hello,
world!</greetings>
```





## Examples of Document Type Declaration 2/2

---

- The same constraints can be specified inside an external file (hello.dtd) and declared in the following way.

```
<?xml version="1.0"?>
<!DOCTYPE greetings SYSTEM "hello.dtd">
<greetings>Hello, world!</greetings>
```



# DTD and XML Schema

---

- A DTD constrain an XML document.
- It is however possible to specify constraints more complex than the one allowed by DTD.
- For instance: imported keys, uniqueness constraints, or the domains of elements and attributes (as in SQL).
- The XML Schema allows to specify this kind of constraints, and it is therefore an alternative to the DTD.
- Moreover, the constraints of XML Schema are expressed in XML.



# Proper and improper usage of XML

---

- XML allows a great degree of freedom to the designers of XML documents.
- The designer can decide the tags and attributes to use, and where to put the data.
- When XML is used to store data, particular attention must be paid to the correct usage of the elements of the data model: elements, attributes, contents, hierarchies.
- We will now provide some guidelines.



# 1 – Data in elements content

---

- Data must be stored in the content of the elements.
- For instance,

```
<book>  
    <title>The Great Gatsby</title>  
</book>
```

Must be preferred to

```
<book title="The Great Gatsby" />
```

AVOID THIS



## 2 – Metadata in attributes or names of elements

---

```
<book>
  <property>
    <name>title</name>
    <value>The Great Gatsby</value>
  </property>
</book>
```

TO AVOID



- In this example, title is a metadata, therefore it must not appear as content.



## 3 – (In)Correct usage of hierarchies

---

```
<db>
  <title>The Great Gatsby</title>      TO AVOID
  <author>F Scott Fitzgerald</author> ↗
  <title>For Whom the Bell Tolls</title>
  <author>Ernest Hemingway</author>
</db>
```

- The two titles and the two authors are only separated by the order of elements, while they compose well distinct objects (they refer to different books).
- `<title>` and `<author>` must be child of one element `<book>`, and not child of the element `<db>`.



## 4 – Correct use of hierarchies

---

```
<db>
  <book>
    <title>The Great Gatsby</title>
    <author>F Scott Fitzgerald</author>
  </book>
  <book>
    <title>For Whom the Bell Tolls</title>
    <author>Ernest Hemingway</author>
  </book>
</db>
```