Data for the Web

INSA de Lyon

Computer Science and Information Technology Department

3rd Year

Part 2: XML Core

Előd EGYED-ZSIGMOND

INSTITUT NATIONAL
DES SCIENCES
APPLIQUÉES
LYON

Plan

- Introduction
- XML Core
- XML Galaxy
- NOSQL
- Conclusion

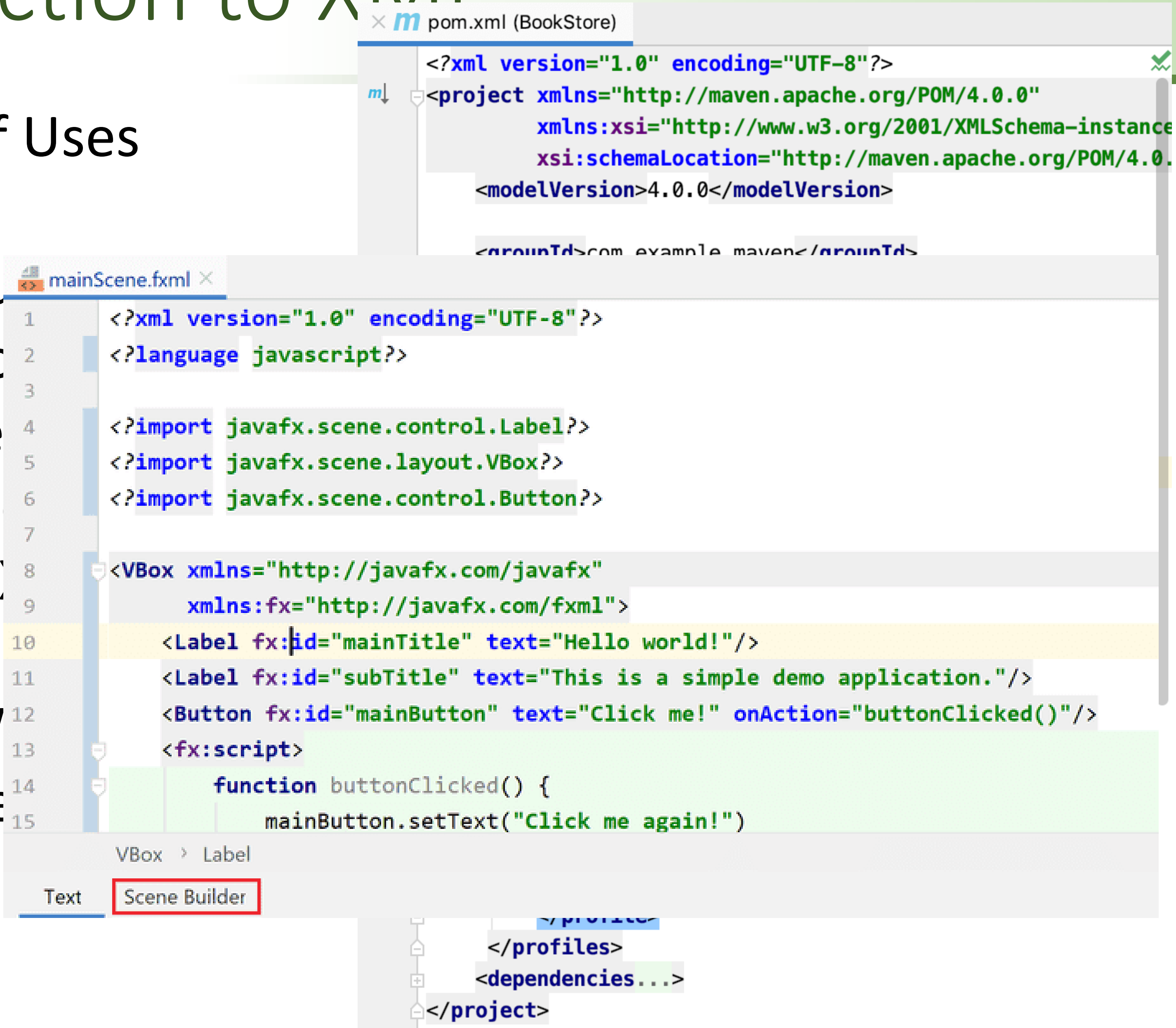
Plan

- Introduction
- XML Core
  - Introduction to XML
  - DTD
  - XML Element
  - XML attribute
  - Reusable Objects: XML Entities
  - XML instance and example
  - Namespaces
  - XML schemas
  - Bibliography
- XML Galaxy
- NOSQL
- Conclusion

Introduction to XMl

Examples of Uses

- Ajax commu
- Configuratic
    - Mave
- GUI (Graphi
    - JavaF
- RSS feed
- Semantic W
- Web Service
- Web Graphi



```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example.maven</groupId>
```

```xml
        </profile>
    </profiles>
    <dependencies...>
</project>
```

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

<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.control.Button?>

<VBox xmlns="http://javafx.com/javafx"
      xmlns:fx="http://javafx.com/fxml">
    <Label fx:id="mainTitle" text="Hello world!"/>
    <Label fx:id="subTitle" text="This is a simple demo application."/>
    <Button fx:id="mainButton" text="Click me!" onAction="buttonClicked()"/>
    <fx:script>
        function buttonClicked() {
            mainButton.setText("Click me again!")
```

Introduction to XML

Where is XML?

- Web (web pages are often XML instances)
- Behind many CMS (content management systems)
- In industrial materials (DocBook)
- In the digital edition (TEI, ePub, docx)
- Programing environments (Gradle, Maven,…)
- …

- It is web browser compatible

What is XML?

- XML: a "skeleton" for creating markup languages
- You already know it!
  - syntax is identical to XHTML's:

```
<element attribute="value">content</element>
```

- Languages written in XML specify:
  - Tag names in XHTML: `h1, div, img`, etc.
  - Attribute names in XHTML: `id/class, src, href`, etc.
  - Rules about how they go together in XHTML: inline vs. block-level elements

Things that can appear in an XML document

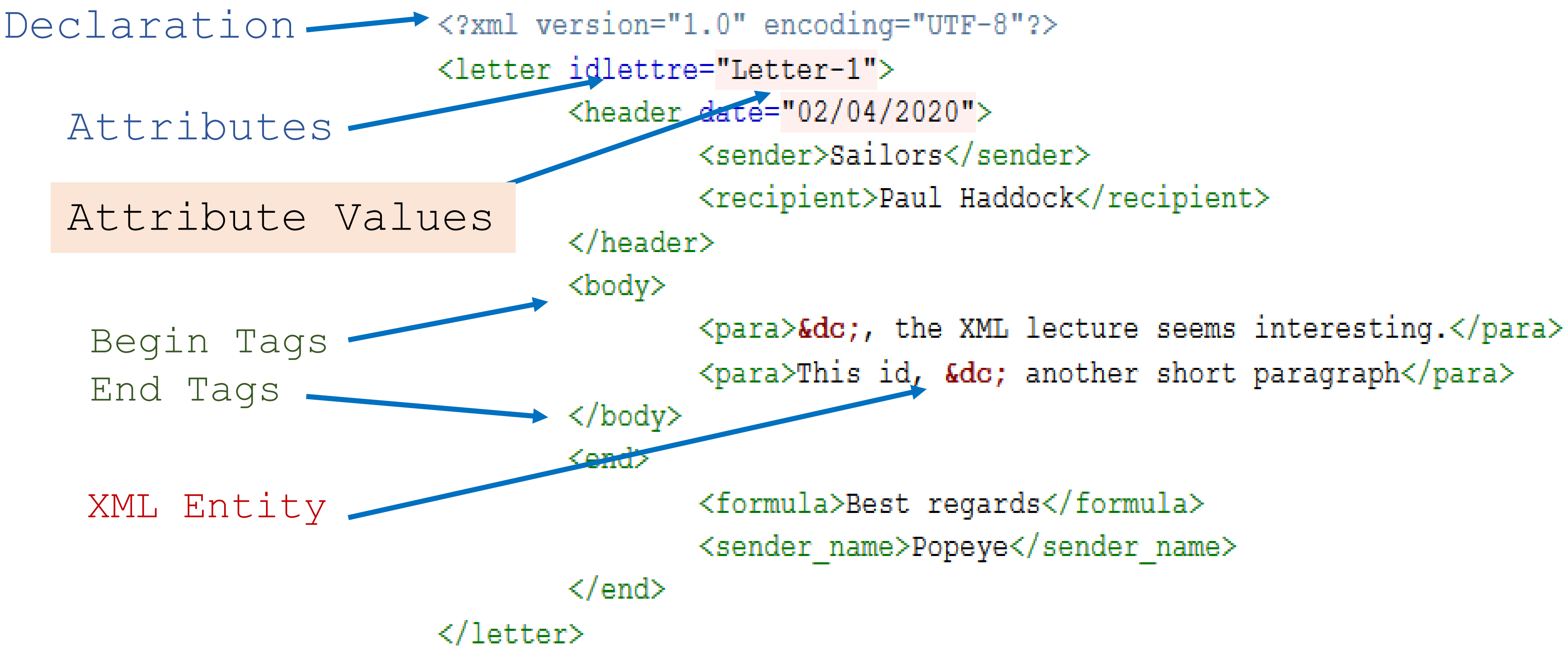
- ELEMENTS: *simple*, *complex*, *empty*, or *mixed* content; *attributes.*

- The XML declaration

- Processing Instructions(PIs) `<? …?>`
    - Most common is `<?xml-stylesheet …?>`
    - `<?xml-stylesheet type=“text/css” href=“mys.css”?>`

- Comments `<!--` *comment text* `-->`

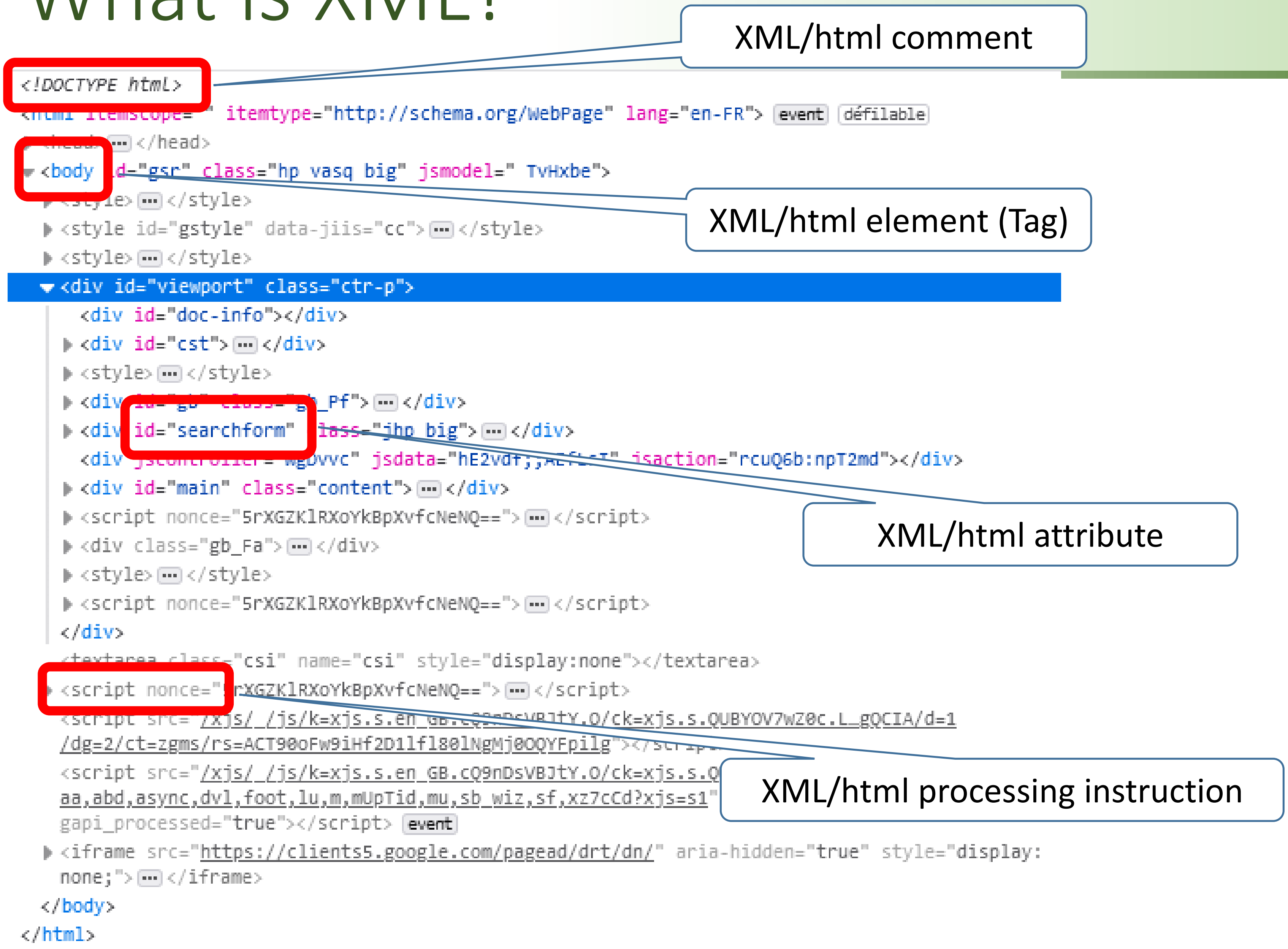
Parts of an XML document



```xml
<?xml version="1.0" encoding="UTF-8"?>
<letter idlettre="Letter-1">
    <header date="02/04/2020">
        <sender>Sailors</sender>
        <recipient>Paul Haddock</recipient>
    </header>
    <body>
        <para>&dc;, the XML lecture seems interesting.</para>
        <para>This id, &dc; another short paragraph</para>
    </body>
    <end>
        <formula>Best regards</formula>
        <sender_name>Popeye</sender_name>
    </end>
</letter>
```

An **XML element** is everything from (including) the element's start tag to (including) the element's end tag.

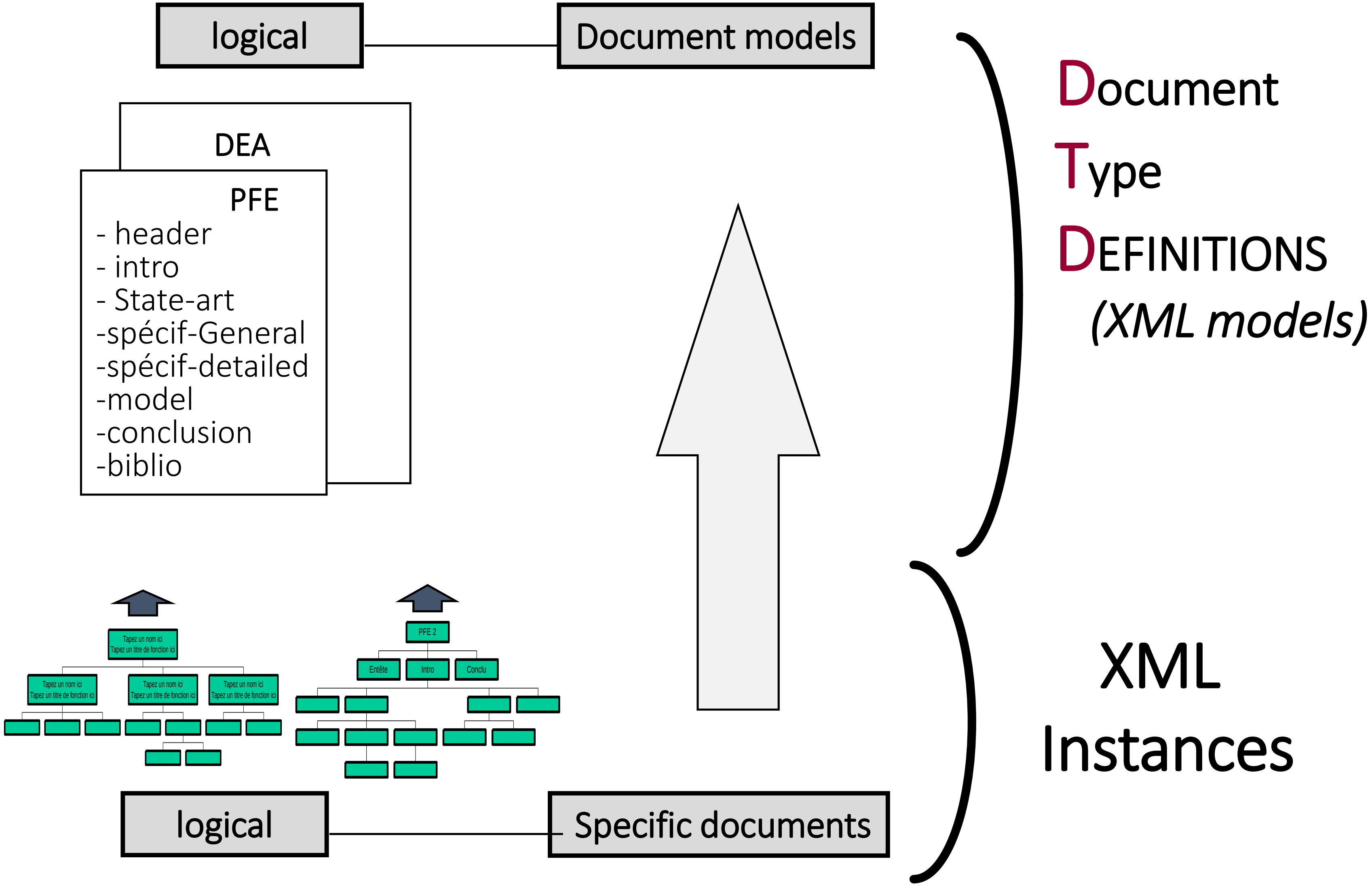
What is XML?



```
<!DOCTYPE html>
<html itemscope="" itemtype="http://schema.org/WebPage" lang="en-FR"> event défilable
  <head> ··· </head>
  ▼ <body id="gsr" class="hp vasq big" jsmodel=" TvHxbe">
    ▶ <style> ··· </style>
    ▶ <style id="gstyle" data-jiis="cc"> ··· </style>
    ▶ <style> ··· </style>
    ▼ <div id="viewport" class="ctr-p">
        <div id="doc-info"></div>
      ▶ <div id="cst"> ··· </div>
      ▶ <style> ··· </style>
      ▶ <div id="gb" class="gb_Pf"> ··· </div>
      ▶ <div id="searchform" class="jhp big"> ··· </div>
        <div jscontroller="wgDvvc" jsdata="hE2vdf;;AEfLcI" jsaction="rcuQ6b:npT2md"></div>
      ▶ <div id="main" class="content"> ··· </div>
      ▶ <script nonce="5rXGZKlRXoYkBpXvfcNeNQ=="> ··· </script>
      ▶ <div class="gb_Fa"> ··· </div>
      ▶ <style> ··· </style>
      ▶ <script nonce="5rXGZKlRXoYkBpXvfcNeNQ=="> ··· </script>
    </div>
    <textarea class="csi" name="csi" style="display:none"></textarea>
    ▶ <script nonce="5rXGZKlRXoYkBpXvfcNeNQ=="> ··· </script>
    <script src="/xjs/_/js/k=xjs.s.en_GB.cQ9nDsVBJtY.O/ck=xjs.s.QUBYOV7wZ0c.L_gQCIA/d=1
    /dg=2/ct=zgms/rs=ACT90oFw9iHf2D1lf180lNgMj0OQYFpilg"></script>
    <script src="/xjs/_/js/k=xjs.s.en_GB.cQ9nDsVBJtY.O/ck=xjs.s.Q
    aa,abd,async,dvl,foot,lu,m,mUpTid,mu,sb_wiz,sf,xz7cCd?xjs=s1"
    gapi_processed="true"></script> event
    ▶ <iframe src="https://clients5.google.com/pagead/drt/dn/" aria-hidden="true" style="display:
    none;"> ··· </iframe>
  </body>
</html>
```

Introduction to XML

SGML model inherited by XML



logical — Document models

DEA

PFE
- header
- intro
- State-art
-spécif-General
-spécif-detailed
-model
-conclusion
-biblio

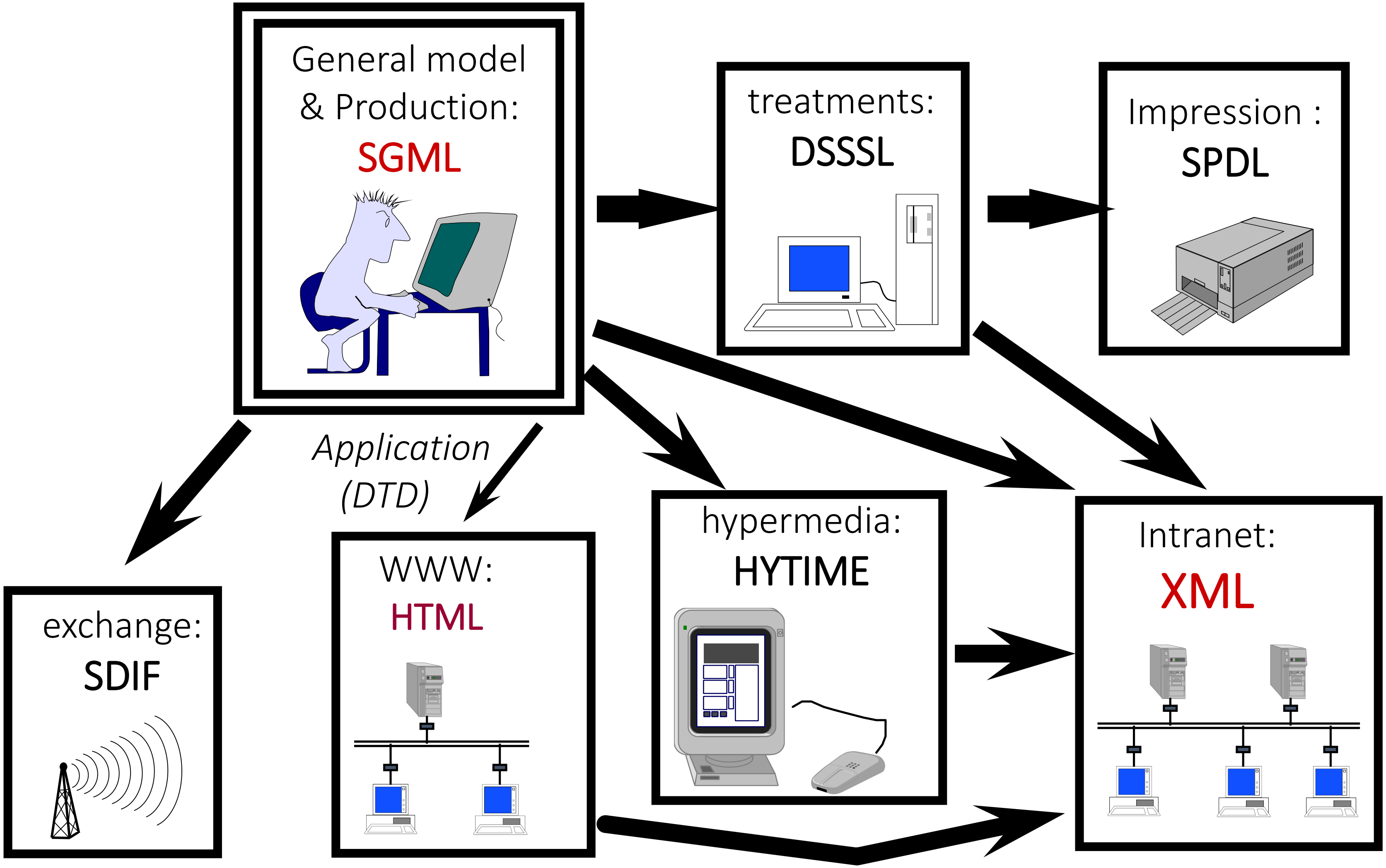
Document
Type
DEFINITIONS
(XML models)

XML
Instances

logical — Specific documents

Introduction to XML

The SGML family



General model & Production: SGML

treatments: DSSSL

Impression : SPDL

Application (DTD)

exchange: SDIF

WWW: HTML

hypermedia: HYTIME

Intranet: XML

Introduction to XML

Why XML?

- ➢ **HTML and SGML** offer imperfect solutions for document and structured information exchange on the Internet or Intranet

- ➢ SGML is maladjusted for Hyperdocument management
  - ✓ Very complex and heavy useless options
  - ✓ Does not support hyperlinking mechanisms
    - → *Poorly suited to WEB*
  - ✓ No Browser / SGML Editor in public domain
    - → *Solutions owners "onerous"*

- ➢ HTML is limited :
  - ✓ presentation oriented hypertext
  - ✓ this is a fixed SGML application (DTD + software) :

    inability to define new tags
    difficult adaptation to customer specific applications

Introduction to XML

- XML is a simplified version of SGML
  - e**X**tensible **M**arkup **L**anguage

  - Created for exchanging data on the web
  - Strict separation between content and presentation
  - Simplicity, universality and extensibility
  - Text format with support for special characters
  - Strong structuring
  - Document Templates (DTD and XML schemas)
  - Free format

Introduction to XML

strong structuring of the document: Examples

HTML :

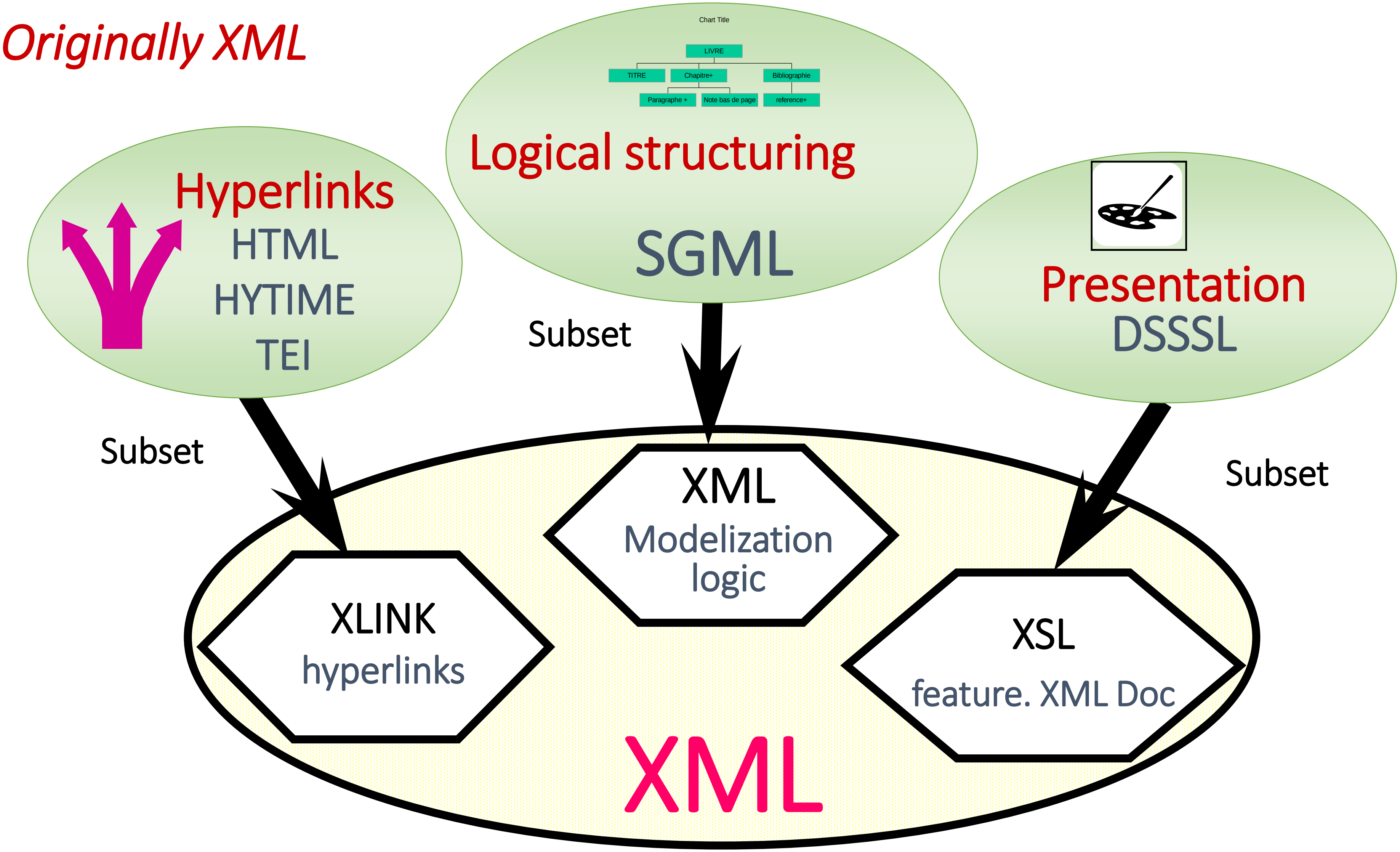
```
---------------------
<html>
<head>
<title>
 extraterrestrial dialogue
</title>
</head>
<body bgcolor="White"
text="dark blue">
<P> Hi Earth! </P>
<P> Login and land! </P>
</body>
</html>
```

XML :

```
<?xml --------  ?>
 ---------------------
<play>
<title>extraterrestrial
Ddialogue
</title>
<conversation>
<Greeting>Hi Earth!
</Greeting>
<Answer>Log in and
land!</Answer>
</conversation>
</play>
```

Introduction to XML

Originally XML



Hyperlinks
HTML
HYTIME
TEI

Logical structuring
SGML

Presentation
DSSSL

Subset

Subset

Subset

XML
Modelization logic

XLINK
hyperlinks

XSL
feature. XML Doc

XML

XML galaxy

Related Languages:

xPath, XQuery, XML Schemas, Relax-NG, XSLT, ...

dialects:

RSS, SVG, XUL, MathML, WSDL, SOAP, OpenStreetMap, SAML, OpenDoument, TEI, DocBook, epub...

XML file structure

Document prolog

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE letter SYSTEM "letter.dtd" [
<!ENTITY dc "Dear Captain">
]>
```

Comment

```
<!-- This is a comment -->
```

Document Elements

```
<letter idlettre="Letter-1">
      <header date="02/04/2020">
            <sender name="Sailors"/>
            <recipient>Paul Haddock</recipient>
      </header>
      <body>
            <para>The XML lecture seems interesting.</para>
            <para>This is another short paragraph.</para>
      </body>
      <end>
            <formula>Best regards</formula>
            <sender_name>Popeye</sender_name>
      </end>
</letter>
```

Plan

- Introduction
- XML Core
  - Introduction to XML
  - DTD
  - XML Element
  - XML attribute
  - Reusable Objects: XML Entities
  - XML instance and example
  - Namespaces
  - XML schemas
  - Bibliography
- XML Galaxy
- NOSQL
- Conclusion

DTD Introduction

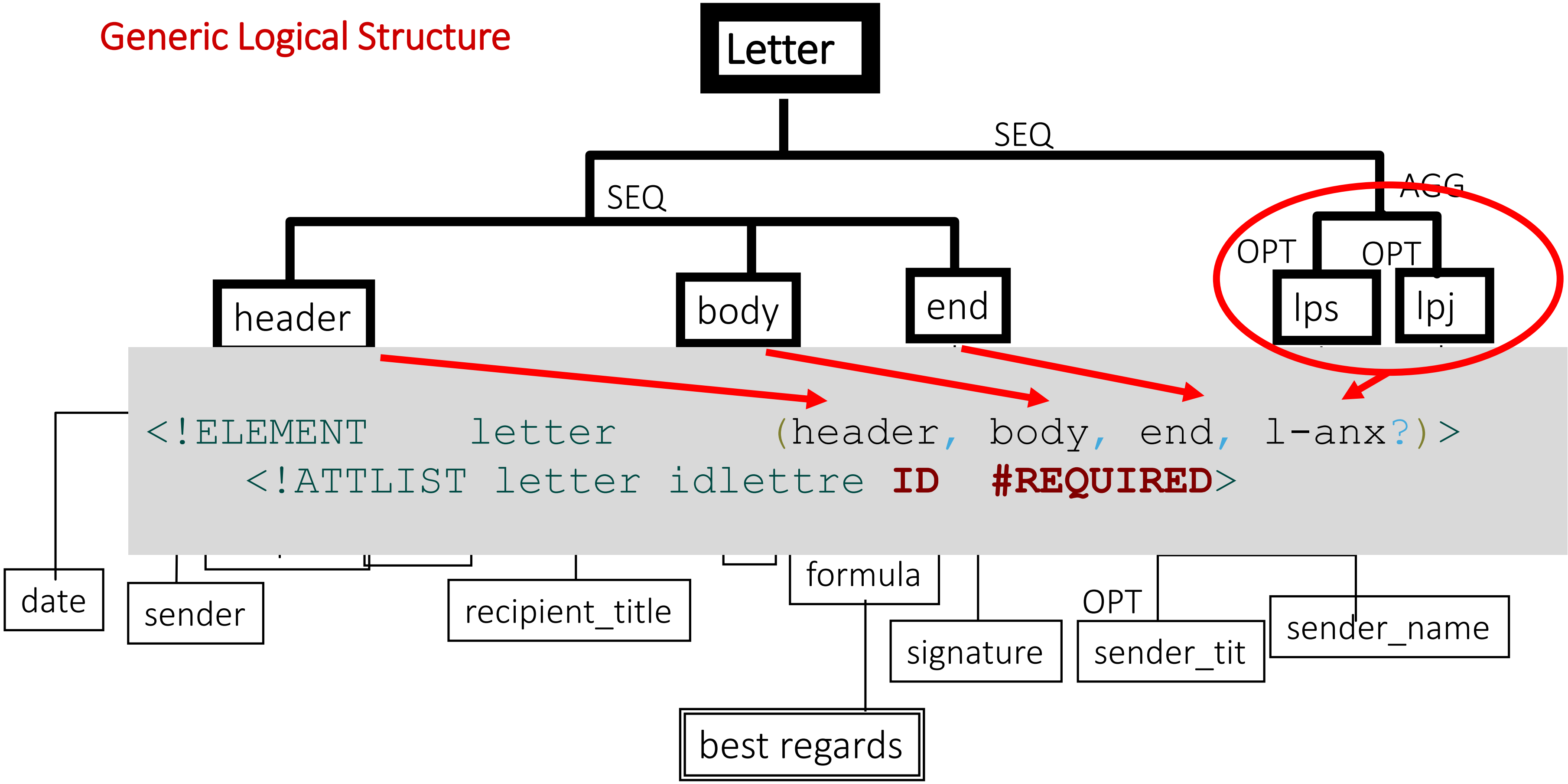
- An XML document must be well formed (syntactically correct)
- Can be valid with respect to a model
- Specifying a grammar for a language
  - automatically test its compliance with a given document
  - facilitate the exchange and sharing of documents produced by different authors;
  - help developers who create automated tools to process documents following the same DTD.

DTD Introduction

- DTD: mechanism by which structures are specified.
- The DTD can be directly in the document
- Necessary to verify the validity of the document
- A DTD is applicable to multiple XML documents.
- Enables defining new languages

DTD introduction

Generic Logical Structure



```
<!ELEMENT    letter         (header, body, end, l-anx?)>
    <!ATTLIST letter idlettre ID  #REQUIRED>
```

DTD: XML Model (1)

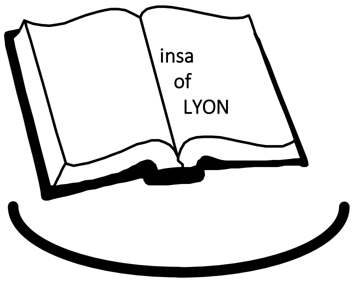
XML "Document" = (*(Representation [Or program] XML a document)*) →

```
<? XML … ..?>
---------
<xxx>
---------
text and tags
---------
</xxx >
```

2 Components (Level 1):

- The model *(Prologue DTD or Scheme)*
- The instance of a specific document

not to be confused with



Physical Document (presentation)

DTD syntax

❖ *Writing conventions*

connectors

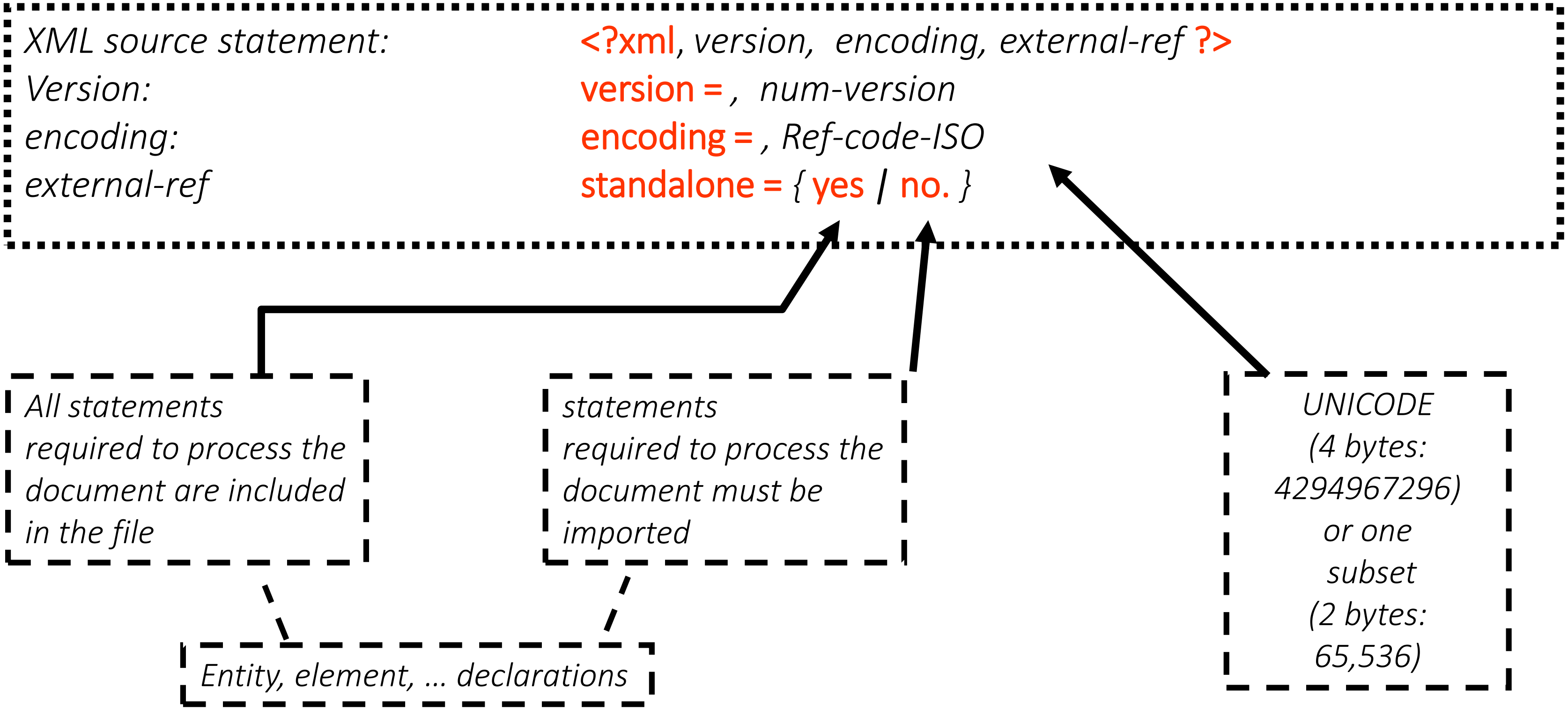
,	AND, ordered *(sequence)*
|	XOR *(choice)*
&	AND, unordered *(aggregate)*

Occurrence indicators

+	One or more times
?	1 or 0
*	0, 1 or more

DTD: XML Model (2)

❖ XML Source Declaration



XML source statement:	<?xml, *version*, *encoding*, *external-ref* ?>
Version:	version = , *num-version*
encoding:	encoding = , *Ref-code-ISO*
external-ref	standalone = { yes / no. }

All statements required to process the document are included in the file

statements required to process the document must be imported

Entity, element, … declarations

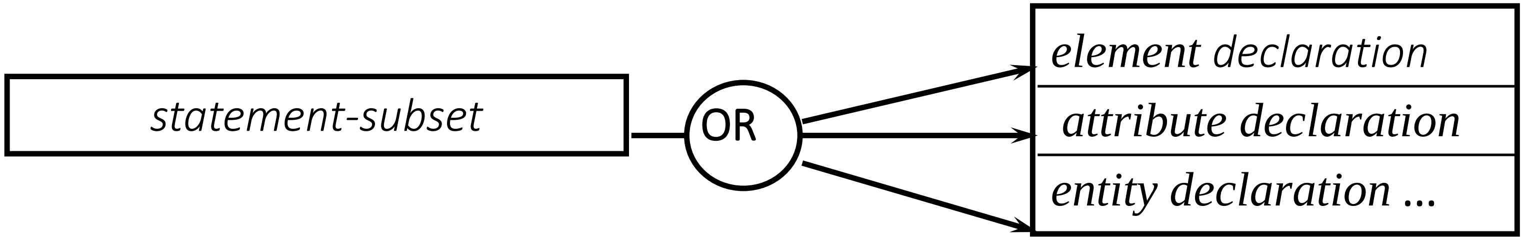
UNICODE (4 bytes: 4294967296) or one subset (2 bytes: 65,536)

example:

<?xml version = '1.0' encoding ="UTF-8" standalone = "yes" ?>

DTD: XML Model (3)

<! DOCTYPE, *dtd-name*, *external_identifier*?,
{ [{ *statement-subset* } + ,]}? ,>



statement-subset

OR

element declaration

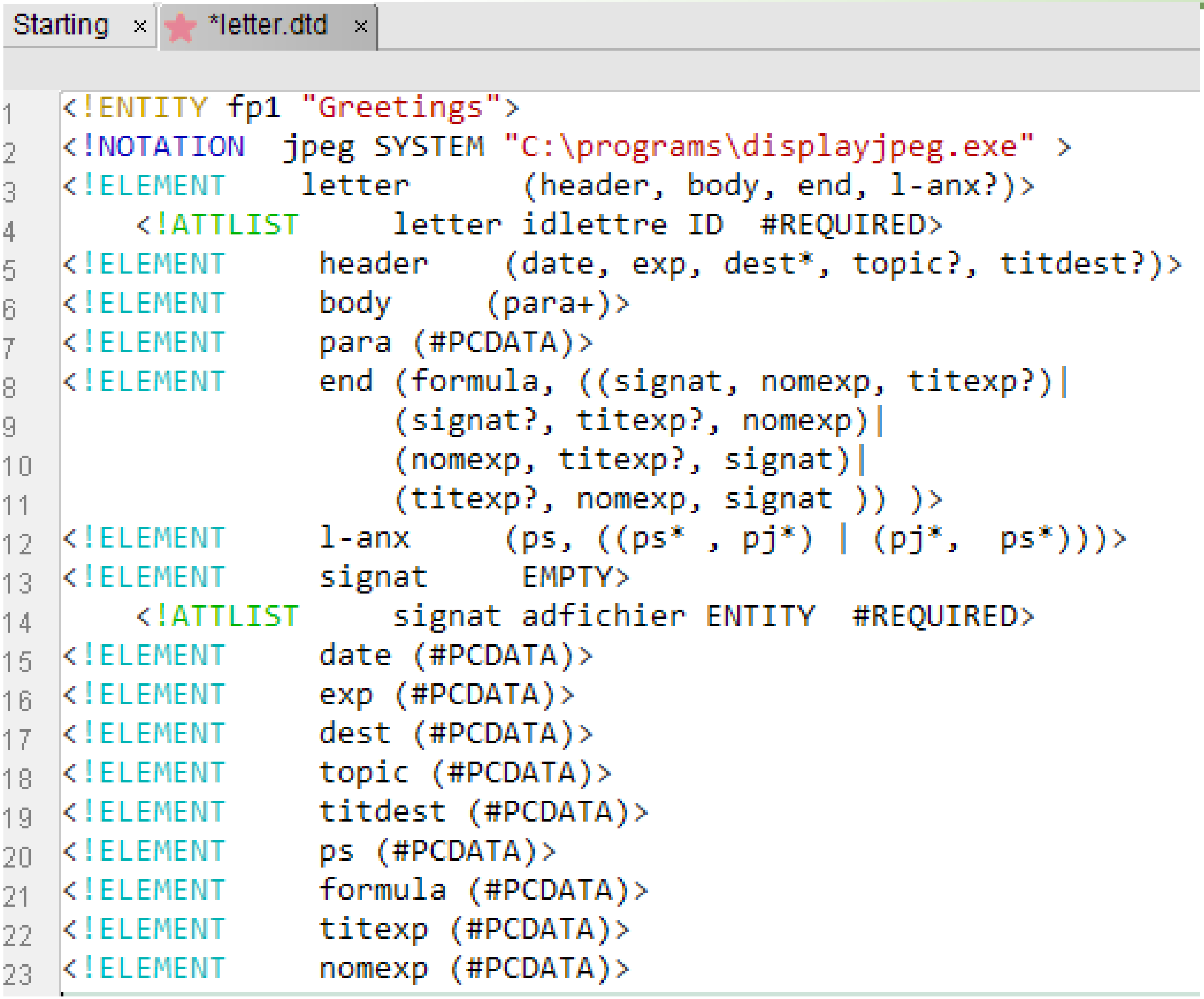
attribute declaration

entity declaration …

Example 1:
<! DOCTYPE *letter* [<-description of the components of a letter ->]>

Example 2: *(Reference to an external DTD)*
<! DOCTYPE *letter* SYSTEM "Letter.dtd" >

DTD: XML Model (3)

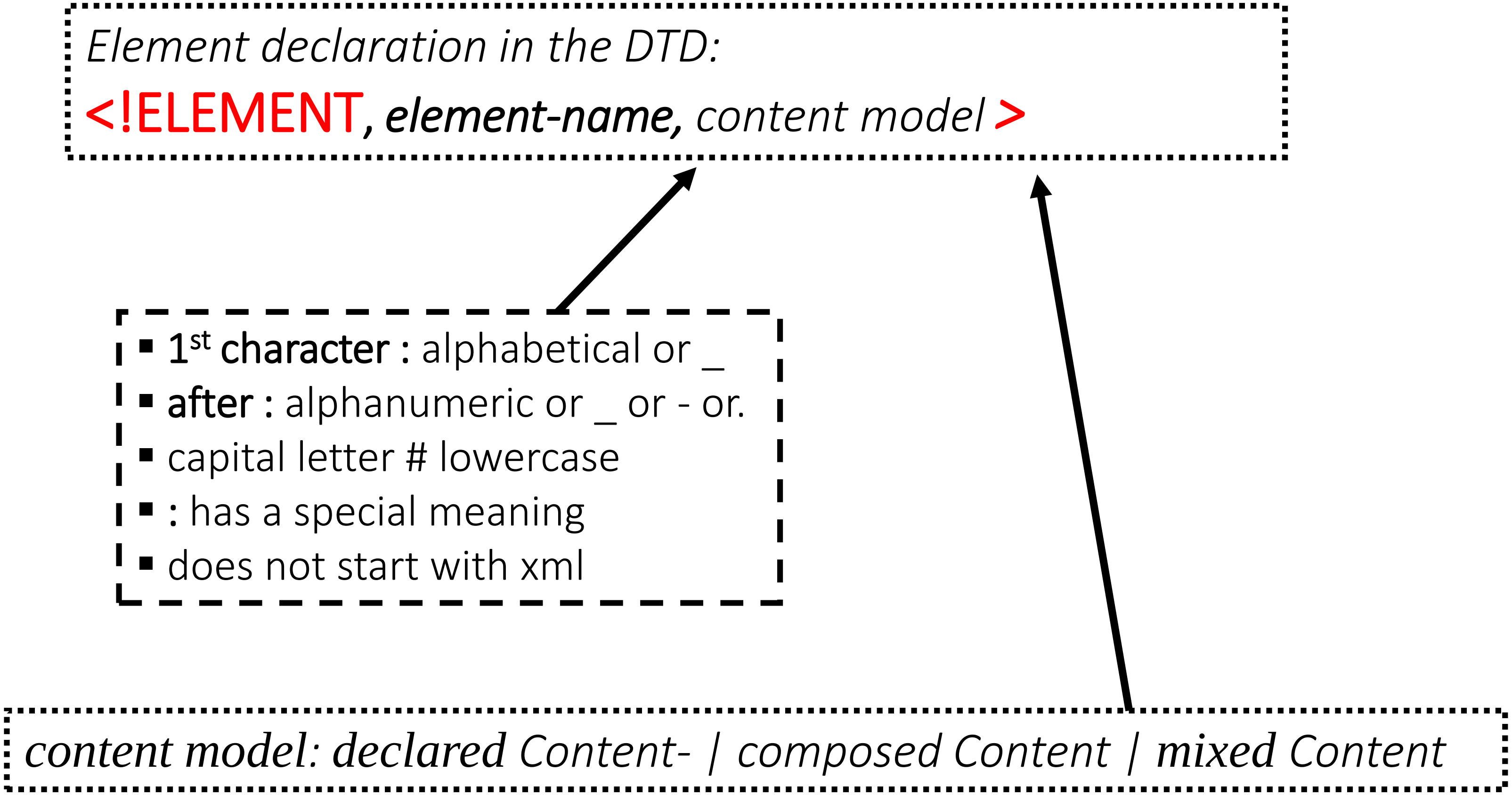


```
<!ENTITY fp1 "Greetings">
<!NOTATION  jpeg SYSTEM "C:\programs\displayjpeg.exe" >
<!ELEMENT    letter      (header, body, end, l-anx?)>
    <!ATTLIST     letter idlettre ID  #REQUIRED>
<!ELEMENT     header    (date, exp, dest*, topic?, titdest?)>
<!ELEMENT     body     (para+)>
<!ELEMENT     para (#PCDATA)>
<!ELEMENT     end (formula, ((signat, nomexp, titexp?)|
                  (signat?, titexp?, nomexp)|
                  (nomexp, titexp?, signat)|
                  (titexp?, nomexp, signat )) )>
<!ELEMENT     l-anx     (ps, ((ps* , pj*) | (pj*,  ps*)))>
<!ELEMENT     signat     EMPTY>
    <!ATTLIST     signat adfichier ENTITY  #REQUIRED>
<!ELEMENT     date (#PCDATA)>
<!ELEMENT     exp (#PCDATA)>
<!ELEMENT     dest (#PCDATA)>
<!ELEMENT     topic (#PCDATA)>
<!ELEMENT     titdest (#PCDATA)>
<!ELEMENT     ps (#PCDATA)>
<!ELEMENT     formula (#PCDATA)>
<!ELEMENT     titexp (#PCDATA)>
<!ELEMENT     nomexp (#PCDATA)>
```

Plan

- Introduction
- Core XML
  - Introduction to XML
  - DTD
  - XML Element
  - XML attribute
  - Reusable Objects: XML Entities
  - XML instance and example
  - Namespaces
  - XML schemas
  - Bibliography
- XML Galaxy
- NOSQL
- Conclusion

XML element (1)



Element declaration in the DTD:

<!ELEMENT, ***element-name,*** *content model* **>**

- **1st character :** alphabetical or _
- **after :** alphanumeric or _ or - or.
- capital letter # lowercase
- : has a special meaning
- does not start with xml

content model: declared Content- | composed Content | mixed Content

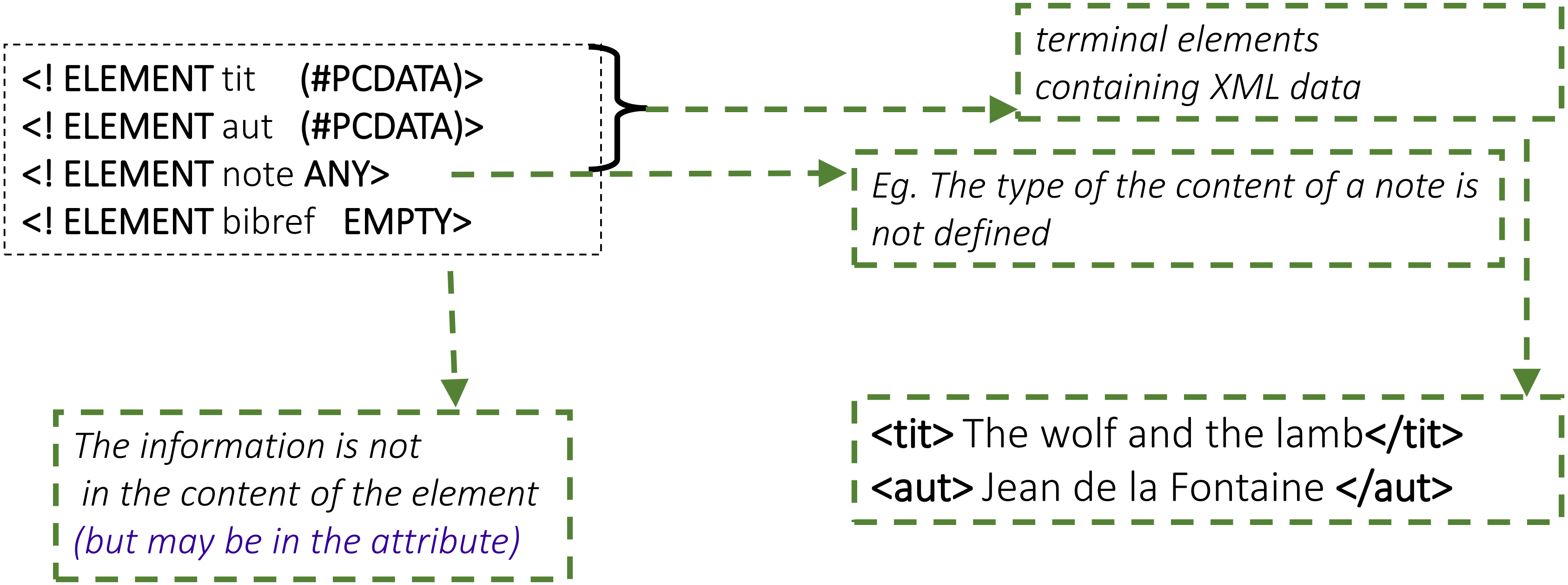
<u>example:</u>

<!ELEMENT **letter** (header, body, end)**>**

XML element (2)

Declared content: (#PCDATA) | EMPTY | ANY

- □ EMPTY : empty element (*one tag appears : <Tag />), May have attributes*
- □ (#PCDATA) : "parsable" text *(Can contain text and references to entities)*
- □ ANY : free content *(Usually not set)*



```
<! ELEMENT tit    (#PCDATA)>
<! ELEMENT aut    (#PCDATA)>
<! ELEMENT note ANY>
<! ELEMENT bibref   EMPTY>
```

terminal elements containing XML data

Eg. The type of the content of a note is not defined

The information is not in the content of the element (but may be in the attribute)

<tit> The wolf and the lamb**</tit>**
<aut> Jean de la Fontaine **</aut>**

XML element (3)

composed content: (,
*{{element name, occ-ind? {connect, element-name, occ-ind? }} * |*
*{composed content, occ-ind? {connect, composed-content, occ-ind?} *}} +* ,)

```
<! ELEMENT paragraph (sentence +)>
<! ELEMENT heading (tit-doc, ss-title?, author, summary)>
```

mixed content: (, {#PCDATA , *connect,*
*{Element name, occ-ind? {connect, Element-name, occ-ind? }}} **
|
*{{Element name, occ-ind? {connect, Element-name, occ-ind? } *}*
connect, #PCDATA} ,)

```
<! ELEMENT paragraph (#PCDATA | list-ord * | reference*)>
<! ELEMENT figure (drawing, legend, #PCDATA)>
```

Recall : *the current version does not support XML connector "Aggregate"*
2 possible connections: the sequence "," and choose "| "

Plan

- Introduction
- Core XML
  - Introduction to XML
  - DTD
  - XML Element
  - XML attribute
  - Reusable Objects: XML Entities
  - XML instance and example
  - Namespaces
  - XML schemas
  - Bibliography
- XML Galaxy
- NOSQL
- Conclusion

XML: Attributes

- Attributes provide extra information about elements
- Placed inside the start tag of an element
- Attributes come in name/value pairs
- E.g. `<img src="computer.gif" />`
- the attribute is "src".
  - value of the attribute is "computer.gif".
  - Since the element itself is empty it is closed by a " /"

XML attribute (1)

- The syntax of DTD uses the keyword ATTLIST, followed by the concerned element name, followed by the list of attributes : for each, the name, type, and wether it is optional or not has to be specfied.

- attribute names must be XML names:
  - the first character is any letter or _ (underscore);
  - the following characters can be letters, numbers, underscores (_), hyphens (-) or dots (.);
  - there is no limitation on the length of an XML name.

XML attribute (2)

❖ Attribute list declaration

- XML elements can have zero, one or more **attributes**
  - attributes contain processing information
  - this information is generally not displayed
  - an attribute can be declared separately from the item to which it relates

Attribute list definition: <! ATTLIST, Element-name,
{attribute-name, attribute-type, default?} +>

CDATA	alphanumeric character string
ID	XML element identifier
IDREF | IDREFS	reference(s) to one (or more) ID-s
ENTITY | ENTITIES	reference(s) to one (or more) entities
NMTOKEN | NMTOKENS	XML symbolic name(s) *('Private', 'public', ...)*
Enumeration	possible value list ('Monday' | 'Tuesday' | Thursday')
RATING	notation used for non-XML entities *(JPEG, ...)*

XML attribute (3)

Default Declaration

v-default	Default Attribute value
#FIXED *'V'*	attribute has only one value *'V‘ which* is imposed
#REQUIRED	a value should always be provided
#IMPLIED	attribute is optional

Exemples

```
<! ATTLIST article_loi langue CDATA #FIXED language 'French'>

<! ELEMENT date (#PCDATA)>
<! ATTLIST date format (ANSI | ISO | EN-exp| fr-exp) #REQUIRED >
..........
<date format = "ISO"> 2018-04-01 </date>

<! ELEMENT section ((ali | xref ) +) >
<! ELEMENT ali  (#PCDATA)>
<! ATTLIST section ident ID #IMPLIED >
<! ELEMENT xref  EMPTY>
<! ATTLIST xref ref IDREF #REQUIRED >
..........
<section ident = 'S425'> <ali> Trick: bli bli ... content stuff section </ali> <ali> .... blo blo </ali> </section>
...............
<section> <ali> blah blah ... see section 'trick' </ali> <xref ref = 'S425' /> <ali> .... blu blu </ali> </section>
```

Plan

- Introduction
- Core XML
  - Introduction to XML
  - DTD
  - XML Element
  - XML attribute
  - Reusable Objects: XML Entities
  - XML instance and example
  - Namespaces
  - XML schemas
  - Bibliography
- XML Galaxy
- NOSQL
- Conclusion

XML Entity (1)

- Intuitively, entities define shortcuts (or aliases) that will be used in XML documents related to the DTD.
- Some entities are already defined in XML:
  - `&lt; (<),`
  - `&gt; (>),`
  - `&amp; (&),`
  - `&quot; (")`
  - `&apos; (').`

More about predefined entities at :

https://en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references

XML entity (2)

Entity

- object (XML or not)
- defines a prologue *(Internal or external)*
- reusable in instances *(General entity)*
  in a DTD (***parameter*** *Entity)*
  in an entity definition, ...

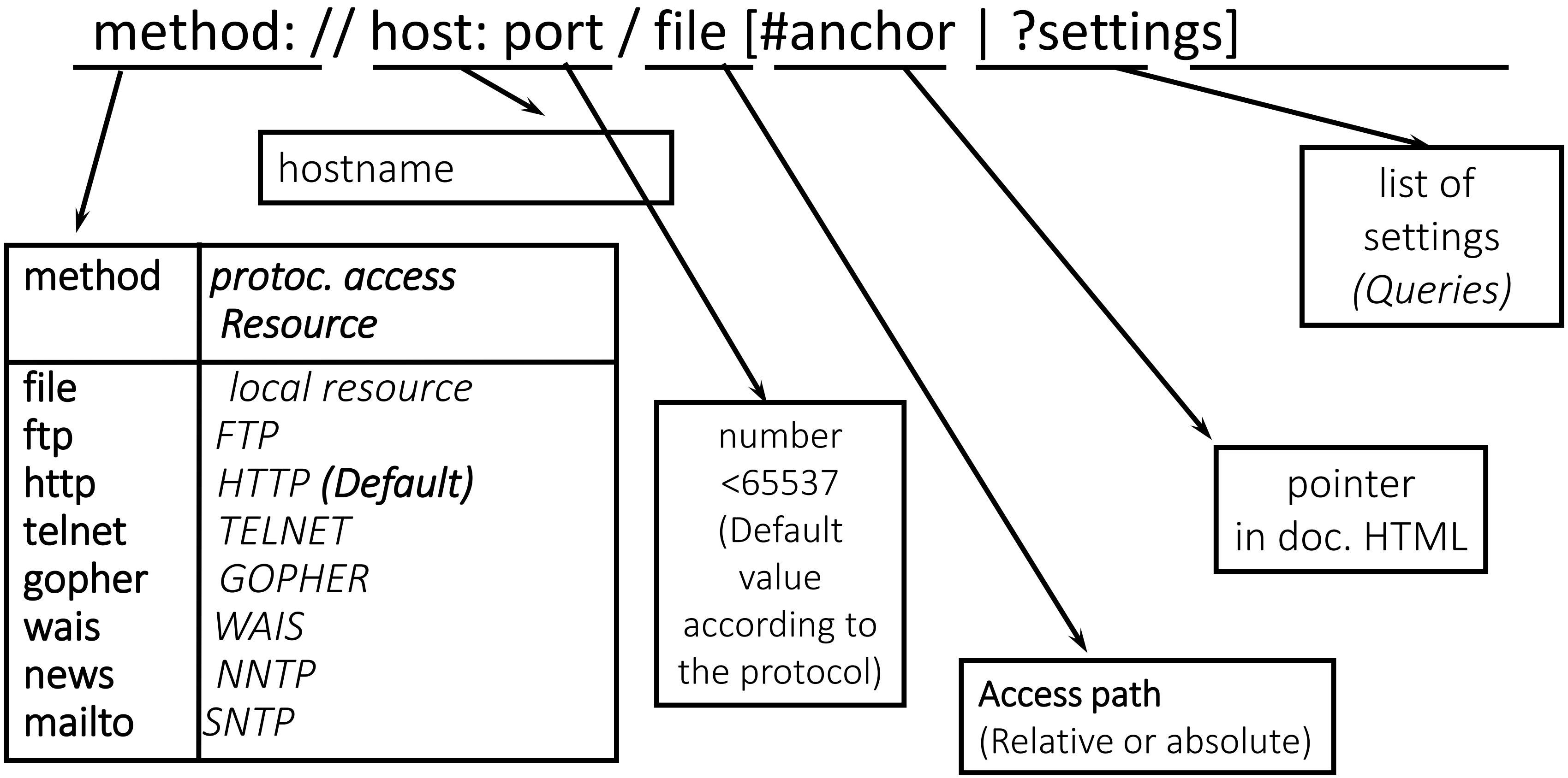
1) Internal general entities:

decl-entity-gen-int : <! ENTITY, *entity-name,* entity-content >
entity-content *XML text in quotation marks or apostrophe*
Reference to an entity: &*name-entity*;

Exemple

in the DTD	<!ENTITY R "Region"> <!ENTITY Ra "Rhone Alpes Auvergne"> <!ENTITY RRA "&R; &Ra ;" >
in the XML instance	**<para>** The **&RRA;** is one of the four dragons of Europe ....**</para>**
generated text	The Region Rhone Alpes Auvergne is one of the four dragons of Europe .....

URL: Universal Resource Locator



method: // host: port / file [#anchor | ?settings]

hostname

method	*protoc. access Resource*
file	*local resource*
ftp	*FTP*
http	*HTTP* ***(Default)***
telnet	*TELNET*
gopher	*GOPHER*
wais	*WAIS*
news	*NNTP*
mailto	*SNTP*

number <65537 (Default value according to the protocol)

Access path (Relative or absolute)

pointer in doc. HTML

list of settings *(Queries)*

example **:** *http: //www.insa-lyon.fr:80/Labos/LIRISthemes/siam.xml#dad*
http: //www.univ-lyon1.fr/cgi-bin/phf-lyon1?Qname=egyed

XML Entity : Parameter Entities

Shortcut for repeating syntax within a DTD

Example of an Employee DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT employee (firstname, surname)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT surname (#PCDATA)>
```

Let’s now create an in-line DTD that uses the above DTD as a parameter entity.

XML Entity : Parameter Entities

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE payroll [
    <!ENTITY %employeedtd SYSTEM "Employee.dtd">
    %employeedtd;
    <!ELEMENT payroll (employee, salary, mailingaddress)>
    <!ELEMENT salary (#PCDATA)>
    <!ELEMENT mailingaddress (employee, address)>
    <!ELEMENT address (#PCDATA)>
]>
<payroll>
    <employee>
        <firstname>Mark</firstname>
        <surname>Collins</surname>
    </employee>
    <salary>£35,000.00</salary>
    <mailingaddress>
        <employee>
            <firstname>Mark</firstname>
            <surname>Collins</surname>
        </employee>
        <address>34, Narrow Lane, SE3 6DY, London</address>
    </mailingaddress>
</payroll>
```

XML entity (8)

- Characters entities
  - &#, code-unicode-decimal, ;
  - &#x, code-unicode-hexadecimal, ;
  - Allow the use of characters no available on the keyboard
    - ex: & # 923; & # 953; & # 947; & # 959; Λίγο
    - Banned in a given context
    - Ex <aut> Smith &#38; al;. </aut> "Smith **&**al .; "

- Comment
  - <!--, character * -->
    - <!-- this is a comment -->

Exemple DTD XHTML

https://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd

Plan

- Introduction
- Core XML
  - Introduction to XML
  - DTD
  - Element XML
  - XML attribute
  - Reusable Objects: XML Entities
  - XML instance and example
  - Nominal fields (namespace)
  - XML schemas
  - Bibliography
- Galaxy XML
- NOSQL
- Conclusion

DTD location (3 cases)

Internal DTD

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE lettre [
<!ELEMENT lettre (entete, corps, fin, l-anx?)>
   <!ATTLIST lettre idlettre CDATA #REQUIRED>
<!ELEMENT entete (date, exp, dest*, sujet?, titdest?)>
<!ELEMENT corps (parag+)>
…
<!ELEMENT titexp (#PCDATA)> ]>
<lettre idlettre="idlettre1">
…
</lettre>
```

External DTD

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE lettre SYSTEM "lettre.dtd" >
<lettre idlettre="lettre-1">
   …
</lettre>
```

DTD location

mixed DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<! ELEMENT    letter (header, body, end, l-anx?)>
<! ATTLIST     letter idlettre ID  #REQUIRED>
<! ELEMENT     header  (date, exp %autres.elements.entete;)> ...
```

DTD

```
<?xml version="1.0" encoding="Utf-8"?>
<! DOCTYPE letter SYSTEM  "LetterToMix.dtd" [

<!- Adding 1 child dest to the element header ->
<! ENTITY %autres.elements.entete  "dest">

  <!- redeclaration Attribute idlettre of the lettre element ->
  <!- The attribute becomes optional ->
<!ATTLIST letter idlettre ID  #IMPLIED>

<!- Declaration of the element recipient ->
<!ELEMENT recipient (#PCDATA)>
  ]>
<letter>
     <header> ...
```

XML instance

DTD drawbacks

- the root element is not specified in the DTD; a document can be valid using any tag defined in the DTD as root;

- the number of occurrences of an element can not be specified precisely, since it has only the quantifiers: ?, * and +
  - we would like to say that an item should appear more than 2 times but still less than 5 times;

- there is no content types for attributes and elements
  - (Name, date, postal code, phone number, URL, email address, etc.);

- we can not constrain the content form
  - (Between 5 and 20 characters, containing an @ sign, etc.);

- the language used to define a DTD is not an XML language!

XML instance (1)

❖ The XML instance is a tagged text that contains the **final data**

- it must respect the rules defined in the DTD *(This can be checked using an "XML parser")*
- and be a Well-formed XML document *(openingTag and closing tag)*
- It is structured in elements (declared in the DTD)
- it can reference entities
- It can contain specific instructions ignored by the XML parser and treated by other related software (DBMS viewer, former). Their form is:
- <?*target*, arg_1*...*, arg_{not} ?>

example

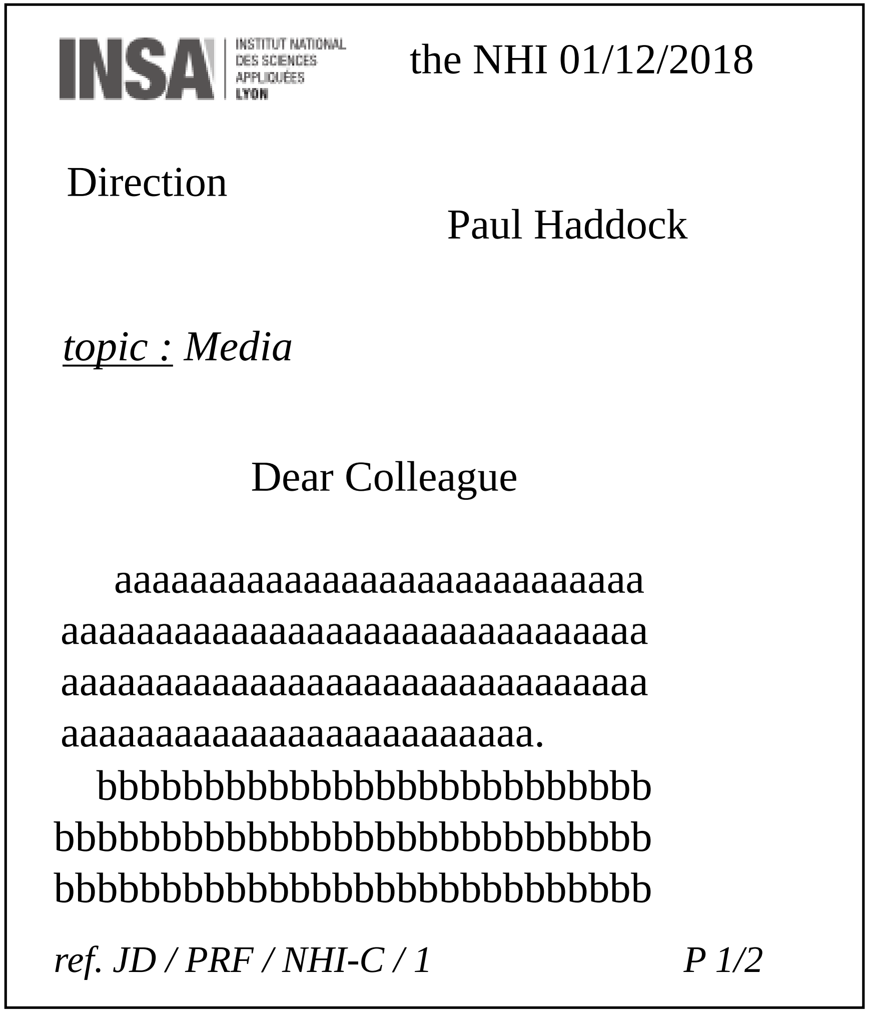
```
<?xml-stylesheet href= "rapportstyle.ccs" type ="text/css"?>
```

XML instance (2)

❖ Well-formed document

- ✓ its structure expressed by its markup can be interpreted as a tree; elements properly nested.
- ✓ the elements can have attributes
- ✓ the DTD of a well formed document may be unknown (none)
- ✓ Consequences
  - □ **an XML fragment**
    - □ is exploitable without its DTD
    - □ is not necessarily valid in the XML sense

XML Example (1)



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

the NHI 01/12/2018

Direction

Paul Haddock

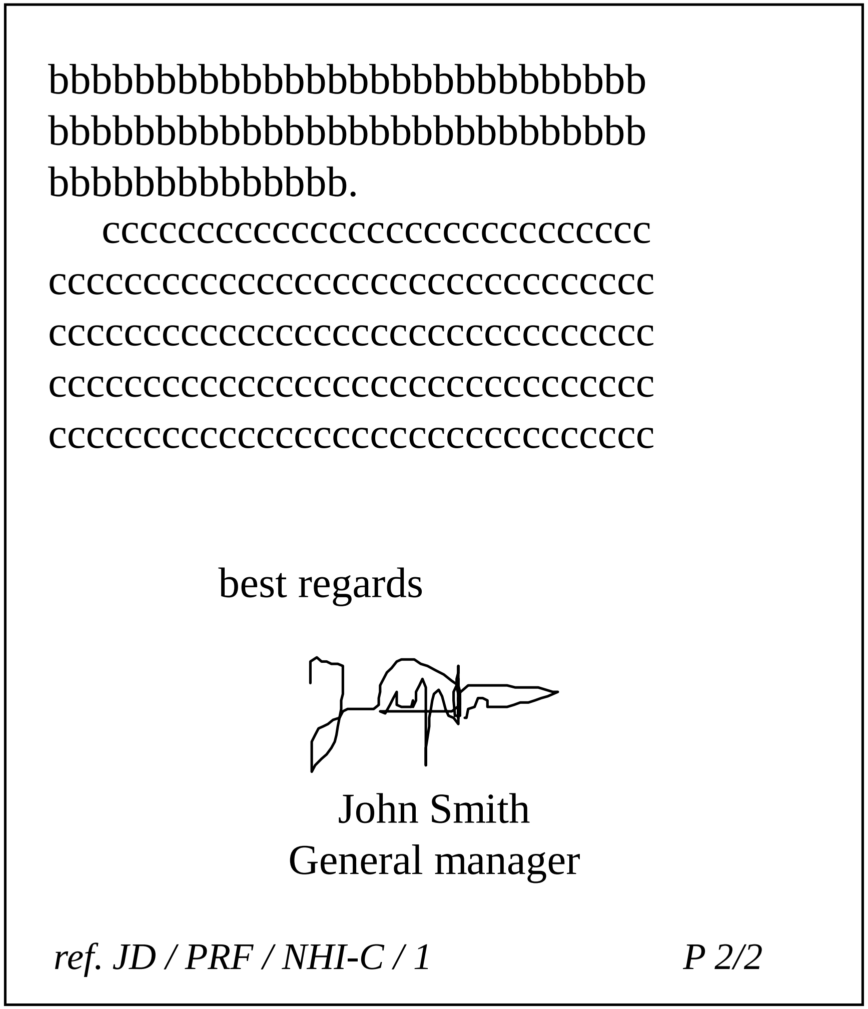
topic : Media

Dear Colleague

aaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaa.

bbbbbbbbbbbbbbbbbbbbbbbbbb bbbbbbbbbbbbbbbbbbbbbbbbbbbb bbbbbbbbbbbbbbbbbbbbbbbbbbbb

ref. JD / PRF / NHI-C / 1 *P 1/2*



bbbbbbbbbbbbbbbbbbbbbbbbbbbb bbbbbbbbbbbbbbbbbbbbbbbbbbbb bbbbbbbbbbbbbb.

ccccccccccccccccccccccccccccc cccccccccccccccccccccccccccccccc cccccccccccccccccccccccccccccccc cccccccccccccccccccccccccccccccc cccccccccccccccccccccccccccccccc

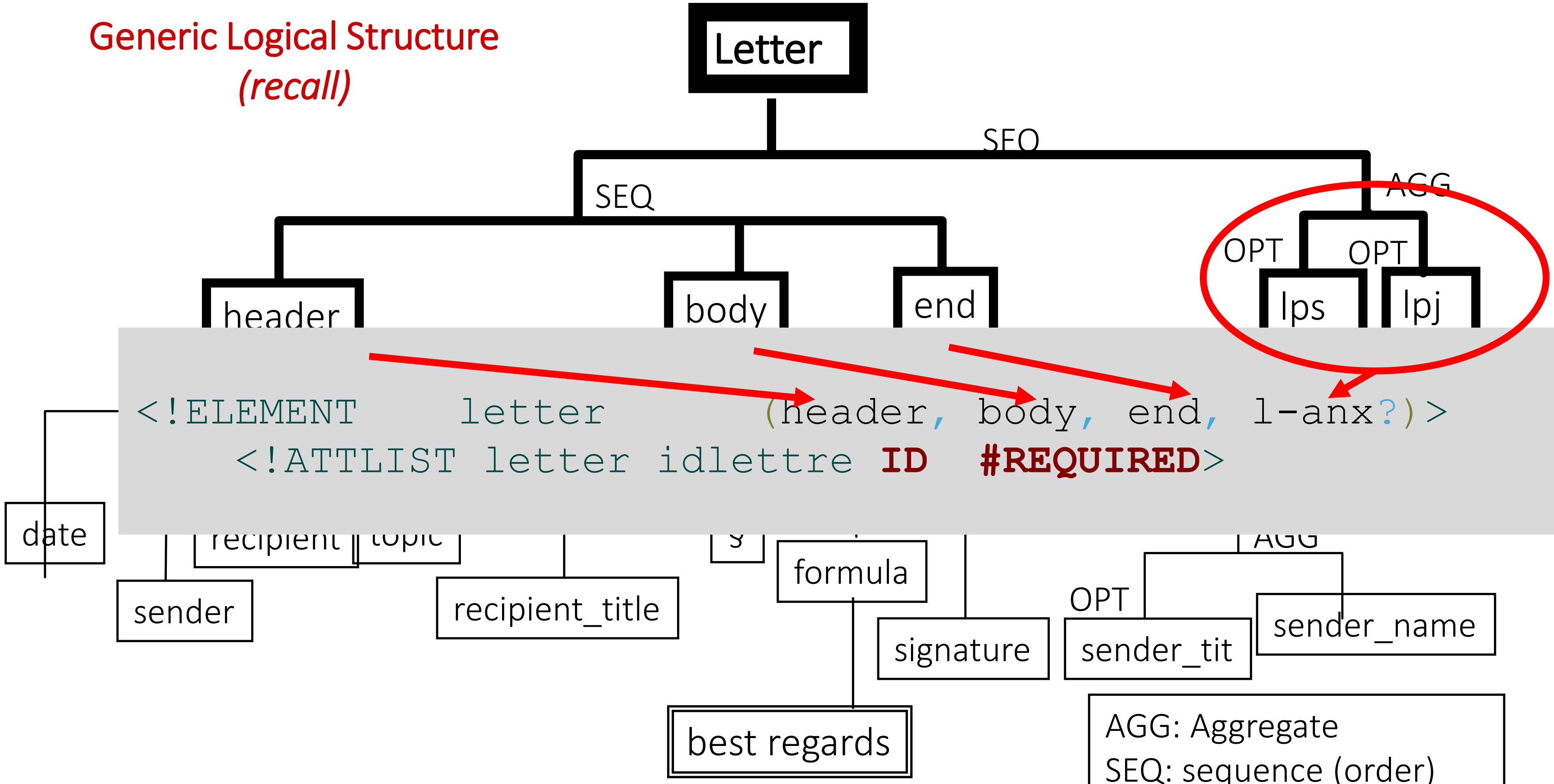
best regards

John Smith
General manager

ref. JD / PRF / NHI-C / 1 *P 2/2*

XML Example (2)

Generic Logical Structure
(recall)



```
<!ELEMENT    letter        (header, body, end, l-anx?)>
   <!ATTLIST letter idlettre ID  #REQUIRED>
```

AGG: Aggregate
SEQ: sequence (order)
REP: repeat (1-n)
OPT: optional (0-1)

XML Example (3)

```
<?xml version="1.0" encoding="UTF-8"?>
<!NOTATION  jpeg SYSTEM "C:\programs\displayjpeg.exe" >
<!ENTITY fp    "Greetings">
<!ELEMENT     letter        (header, body, end, l-anx?)>
    <!ATTLIST     letter idlettre ID  #REQUIRED>
<!ELEMENT      header     (date, sender, recipient* topic?,
recipient_title?)>
<!ELEMENT      body      (para+)>
<!ELEMENT      end (formula, ((signat, sender_name,
sender_tit?)|(signat, sender_tit?, sender_name)|(sender_name,
sender_tit?, signat)|(sender_tit?, sender_name, signat ))  )>
<!ELEMENT      l-anx      (ps, ((ps* , pj*) | (pj*,  ps*)))>
<!ELEMENT      signat      EMPTY>
    <!ATTLIST     signat adfichier ENTITY  #REQUIRED>
<!ELEMENT      date  (#PCDATA)>
<!ELEMENT      sender  (#PCDATA)>
<!ELEMENT      dest  (#PCDATA)>
<!ELEMENT      topic  (#PCDATA)>
<!ELEMENT      sender_name  (#PCDATA)>
<!ELEMENT      ps  (#PCDATA)>
...
```

DTD (file : *lettre.dtd*)

XML Example (4)

```
<?xml version="1.0" encoding="Utf-8" standalone="No."?>
<! DOCTYPE letter SYSTEM "Lettre.dtd" [
<! ENTITY sdupont SYSTEM "D:\signat\dupont.jpg" NDATA jpeg> ]>
<letter idlettre="Letter-1">
  <header>
    <date>02/01/2015 </date>
    <exp> Direction </exp>
    <recipient> Paul Haddock </recipient>
    <topic> Subject: Media </topic>
    <recipient_title> Dear Colleague </recipient_title>
  </header>
  <body>
    <para> aaaaaaaaa..............aaaaaaaaaaaa</para>
    <para> bbbbbbbbbbbb..............bbbbbbbbbbbb</para>
    <para> ccccccccccccc................ccccccccccccccc</para>
  </body>
  <end>
    <formula>&fp;</formula>
    <signat adfichier="sdupont"/>
    <sender_tit>General manager </sender_tit>
    <sender_name>John Smith </sender_name>
  </end>
</letter>
```

a letter instance

- *The logo, presentation, page top and bottom will be processed by XSL*

XML Example (5)

Best practices for a DTD

- 1) modularity: reusable parts in entities
- 2) group entity declaration at the beginning of the DTD
- 3) use parameter entities in an convenient manner
  - for content models
  - for attribute declaration
- 4) simplify content models
- 5) proper use of comments
- 6) adapted DTD:
  - avoid too general DTD full of useless XML objects
  - prefer more specific DTD-s

Plan

- Introduction
- Core XML
  - Introduction to XML
  - DTD
  - Element XML
  - XML attribute
  - Reusable Objects: XML Entities
  - XML instance and example
  - Namespaces
  - XML schemas
  - Bibliography
- XML Galaxy
- NOSQL
- Conclusion

XML Namespaces

Reasons for using namespaces:

- Distinguish between elements and attributes from different applications with the same name.
- Group all related elements and attributes from one application to recognise them easier.

- Distinguish between elements and attributes from different applications with the same name.
- Group all related elements and attributes from one application to recognise them easier.

- One of the main feature of XML is to combine markup from various XML applications. Hence, XML Namespaces make it easier for software to understand the nature of the markup been used.

Namespaces (1)

- Namespaces *(espaces nominaux)* enable to qualify in a unique way all the XML objects *(Elements, attributes, ...).*
    - Namespaces allow coexistence in the same document of objects with the same name but with a different connotation: like:
        - *Kitchen table)* → **kitchen:table**
        - *Table (SQL)* → **sql:table**
    - advantages:
        - *Cooperation of various XML standards, HTML, XSL, DTD, schemas, ...*
        - *Sharing different documentary tools (different DTD)*
        - *concepts and objects traceability (where was defined the attribute "time"?)*
        - *Defining a coherent vocabulary*
            - *Getting closer to ontologies*

Namespaces (2)

Example: XML allows reusing DTD fragments defined elsewhere

```
<?xml version = '1.0' encoding= ISO-8859-1 ' standalone= 'No'?>
<! DOCTYPE article [
   <! ELEMENT item ANY>
   <! ENTITY % Basic SYSTEM 'http://-------/basic-text.dtd'
   <! ENTITY % math SYSTEM 'http://-------/MathML.dtd'
   <------
  %Basic;
  %math;
   -------]
```

Library containing XML objects related to mathemathics (Eg fn element: function)

Library of XML objects related to basic word processing (Eg fn element. Foot note)

Namespaces (3)

reusing statements (DTD parameter entities) may cause name conflicts :

- ex1: <fn> In the DTD "basic-text.dtd" means a "footnote"

  ```
  <fn id = 'Note 10'> example of footnote on page </fn>
  ```

- ex2 <fn> In the DTD "Mathml.dtd" refers to a "function"

  ```
  <fn>
        <apply>
              <int/>
              <bvar> <ci> x </ci> </bvar>
              <lowlimit> <cn> 0 </cn> </lowlimit>
              <uplimit> <cn> 1 </cn> </uplimit>
        </apply>
  </fn>
  ```

$$\int_0^1 dx$$

Namespaces (4)

- Solution: declare name spaces

  - Attribute **xmlns**
  - Syntax of the declaration: xmlns, prefix = URI
    - *URI =* ***U****niforme* ***R****esource* ***I****dentifier*
  - to associate a *prefix* to a URI
    - a DTD
    - a DTD fragment
    - an XML schema
    - a schema fragment, etc.
  - syntax use: *prefix* ":" tag

Namespaces (5)

- example:

```
<report>
xmlns math = "http://www.w3.org/...../REC-MathML.dtd"
xmlns bt = "http://foo.bar.org/xml/...../Basic-text.dtd" >
<!-- further  -->
     <bt:fn id = "Note 10">
                  example footnote
              </bt:fn>
<!--  further  -->
   <math:fn>
     <math:apply>
      <math:int/>
             <math:bvar> <math:this> X </math:this> </math:bvar>
             <math:lowlimit> <math:cn> 0 </math:cn> </math:lowlimit>
             <math:uplimit> <math:cn> 1 </math:cn></math:uplimit>
    </math:apply>
   </math:fn>
<!-- further  -->
</report>
```

Namespaces (6)

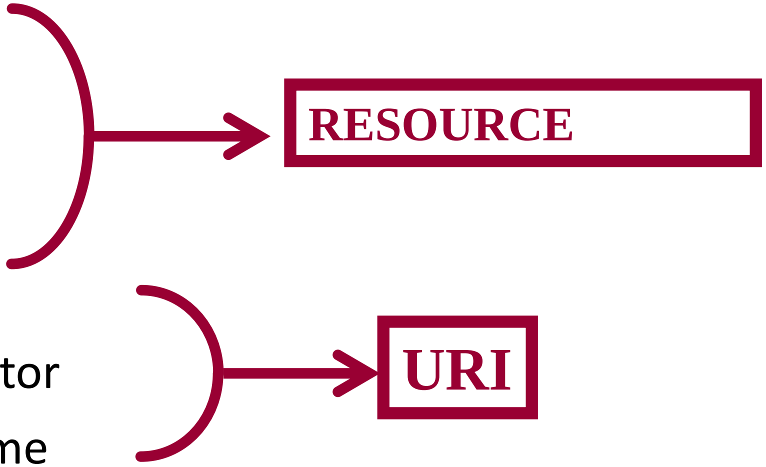
- Scope of a namespace declaration
  - a namespace is valid
    - for the element in which it is declared
    - for all its children
  - default namespaces
    - declared without prefix
    - alleviates writing tags
    - valid for the element in which it is declared and all its children

example

```
<lab-report xmlns = "Http://servif1.insa-lyon.fr/dpt-if/rapport.dtd"
                xmlns math = "http://www.w3.org/...../REC-MathML.dtd"
                xmlns bt = http://foo.bar.org/xml/...../Basic-text.dtd >
   - - - - - -
<Title-rap> Title TP Report </ title-rap>
              <- - the title-rap element is defined in rapport.dtd - ->
```

Namespaces (7)

- *URI* = *Universal Resource Identifier*
  - XML allows to establish links between documents
    - stored on any server;
    - accessible by different protocols *(HTTP, FTP, GOPHER, ...);*
  - These documents are:
    - files,
    - DB query results,
    - the results of programs ....
  - these resources are identified:
    - physically: **U**niversal **R**ESOURCE **L**ocator
    - logically: **U**niversal **R**ESOURCE **N**ame



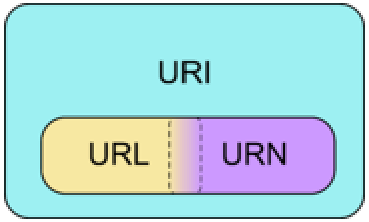
RESOURCE

URI

Namespaces (7)

- *URI* = *Universal Resource Identifier*

A URI has two specializations known as URL and URN



Source: wikipedia

Syntax of URL
scheme: subdomain/domain-name.Top-level-domain/sub-folder

Example of URL
https://www.geekflare.com/articles
mailto:mary@jane.website.com
file:///localhost/8.8.8.8

Syntax of URN
urn:<nid>:<nss></nss></nid>

Example of URN
urn:nbn:de:101:3-2019075675872913
urn:uuid:6r4bc420-9c3a-12i9-97d9-0665700c9a66
ISBN 1-446-2776877-40

https://geekflare.com/difference-between-url-uri-and-urn/

Namespaces

- Some namespace URI’s are standard, and may be recognised by some browsers
  Example:

  http://www.w3.org/1999/xhtml
  http://www.w3.org/1999/XSL/Transform
  http://www.w3.org/2001/svg
  http://www.w3.org/2001/MathML

- Some prefixes are used by convention
  Example:

  xmlns:**html**="http://www.w3.org/1999/xhtml"
  xmlns:**svg**="http://www.w3.org/2001/svg"
  xmlns:**xsl**="http://www.w3.org/1999/XSL/Transform"
  xmlns:**xsd**="http://www.w3.org/2001/XMLSchema"

  However, prefixes can be user-defined.

  Hence, you can change it but make sure it is meaningful to both a human reader and an XML parser.

Some classic namespaces

- XML http://www.w3.org/XML/1998/namespace
- schemas http://www.w3.org/2001/XMLSchema
- schema instances http://www.w3.org/2001/XMLSchema-instance
- XSLT http://www.w3.org/1999/XSL/Transform
- XHTML http://www.w3.org/1999/xhtml
- XLink http://www.w3.org/1999/xlink
- MathML http://www.w3.org/1998/Math/MathML
- SVG http://www.w3.org/2000/svg
- DocBook http://docbook.org/ns/docbook
- Dublin Core http://purl.org/dc/elements/1.1/

Plan

- Introduction
- Core XML
  - Introduction to XML
  - DTD
  - Element XML
  - XML attribute
  - Reusable Objects: XML Entities
  - XML instance and example
  - Namespaces
  - XML schemas
  - Bibliography
- XML Galaxy
- NOSQL
- Conclusion

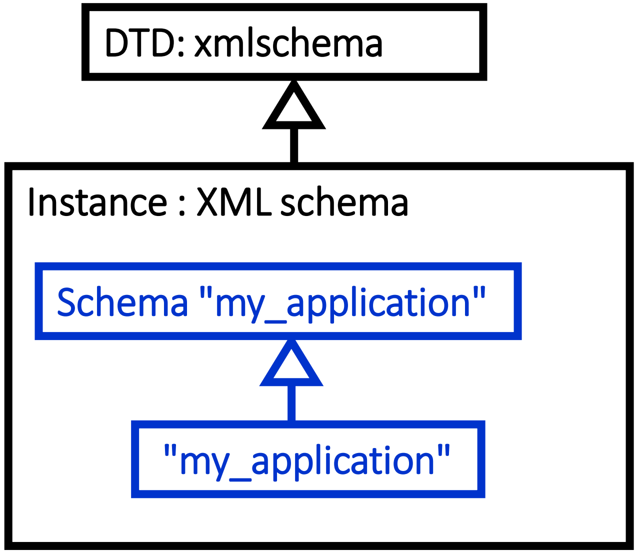
XML schemas

- Why this new concept?
  - DTD: presentation oriented document modeling.
    - ➢ weak typing of final elements:
      - ▪ One predefined type: PCDADA
    - ➢ inaccurate cardinality
    - **Inadequate for structured data manipulation in advanced information systems**

  - 3 concerns grow beyond the DTD:
    - Increase constraint expressivity
    - Take advantage of object oriented data modeling formalisms
    - Adopt an XML syntax to represent the schema itself

XML schemas

- Contribution of XML schemas
  - more accurate and more flexible data modelization than a DTD
  - a schema is an XML document that defines a class of XML documents as a component structure:
    - the items *(usage, meaning, relationships, content)*
    - the attributes and values
    - data types
    - the entities
    - notations



XML schemas

- Writing a schema
  - a schema is composed of elements
    - represented by tags *(DTD was standardized: xsd)*
    - can have attributes
    - associated data types
      - simple types: *contains only text*
      - complex types: *can contain*
        - *attributes*
        - *other elements*
  - elements and attributes are constrained
    - occurrence constraints
      - *optional / mandatory*
      - *default value*
  - ability to group attributes
    - *to facilitate maintenance and readability*

XML schemas

- simple types *(The most common)*
  - string : string of characters , *ex. Villeurbanne*
  - boolean : true| false | 0 | 1
  - decimal : decimal number, *ex. 1.23, -100.25, 0, 10, .15*
  - float : Single precision floating number (32 bits) *ex. 12.78E-2*
  - double : Double precision floating point number (64 bits), *ex. 12.78E-2*
  - timeInstant : *ex. 2003-02-17T16: 14: 45.010 + 01: 00*
    (February 17, 2003 at 16h 14mn 45s 10ms Universal Time + 1 hour)
  - timePeriod : *ex. 2003-02-17T16: 14*
    (February 17, 2003 at 16h 14mn 16h15mn)
  - timeDuration : *ex. P1Y2M3DT10H30M12.3S*
    (Term 1 year 2 months 3 days 10 hours 30 minutes 12.3 seconds)
  - month : *ex. 2003-02: February 2003*
  - recurring date: *ex. - - 05-20: every May 20*
  - recurring day: *ex. - - - - 20: every 20th of each month*
  - binary: Binary string *ex. 01110000111*
  - uriReference : *ex. http: //www.w3.org/1999/XMLSchema*

XML schemas

- simple types *(The most common)*
    - name : Name conforms to XML, *ex. para*
    - QName : Qualified name (with prefix) *ex math: fn, mt: obs*
    - NCName : Prefix unnamed *ex. para*
    - integer: Whole, *ex. 100*
    - PositiveInteger: > 0
    - nonPositiveInteger: < or = 0
    - NegativeInteger: < 0
    - nonNegativeInteger: > or = 0
    - byte: Byte, *ex. 100*
    - date:, *ex. 2003-02-17*
    - time:, *ex. 13: 20: 10 045*
    - etc.

Complete Listing : http://www.w3.org/TR/xmlschema-0/#CreatDt

XML schemas

- types derived from simple types
  - **facets *(Simple type restriction)***
    - **period, duration** address the simple types related to time and designate a period
    - **maxInclusive, minInclusive, maxExclusive, minExclusive** carry on ordered types and designate high or low limits
    - **precision, scale** cover numeric types and respectively specify the number of significant digits and the number of fractional digits
    - **length, minlength, maxlength** indicate lengths (Strict, min or max) channels (print, binary, etc.)
    - **enumeration** list of possible values. *Ex "3IF", "4IF", "5if"*
    - **pattern** to define a shape (regular expression)

      Ex \D{5}: 5 decimals
    - etc.

example

eg. type "monetaryAmount" > or = 0, expressed as a unit or in cents

```
<xs:simpleType name= "monetaryAmount">
    <xs:restriction base = "xs:decimal ">
          <xs:scale value = "2" />        (two decimal places → cents)
          <xs:minInclusive value= "0" />
    </xs:restriction>
</xs:simpleType>
```

XML Schemas: Facets of Simple Types

- Facets = additional properties restricting a simple type
- 15 facets defined by XML Schema

Examples

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

XML schemas

Types derived from simple types

- **lists**
  - **defined using the facet enumeration**

eg list of French departments

```
<xs:simpleType name= "deptsFrance">
    <xs:restriction base = "xs:string" >
        <xs:enumeration value = "Ain" />
        <xs:enumeration value = "Aine" />
    - - - - -
    </xs:restriction> </xs:simpleType>
```

- **Pattern: Regular Expression**
  - **defined using the facet pattern**

eg definition of a postcode

```
<xs:simpleType name= "Postal code" >
     <xs:restriction base = "xs:string" >
              <xs:pattern value = "\d {5} "/> (5 decimals)
      </xs:restriction>
</xs:simpleType>
```

XML schemas

Complex types

- A complex type is either:
  - an element composed of other elements
  - an element having attributes
- it is defined with the element **complexType** who owns
  - an mandatory attribute **name**
  - an optional attribute **content** that can take the following values:
    - **elementOnly** : content will be one or more elements (default)
    - **textOnly** : the content will be of type string (~ #PCDATA)
    - **empty** : the content is empty
    - **mixed** : the content will be mixed

XML schemas

Attribute groups

- is defined with the keyword **attributeGroup**
  - is a child of the **schema** element
  - has a mandatory attribute **name**
  - has a child : **attribute** element
- each **attribute** element has
  - two mandatory attributes
    - **name :** *name of the elements*
    - **type:** *type of element*
  - optional attributes, e.g.
    - use (required| default |**optional** | fixed | ...).
    - value (returns the fixed or default)

XML schemas

- Attribute group example

```
.....
<xs:attributeGroup name = "Metadata">
     <xs:attribute name = "Author" type =“xs:string" use = "required"/>
     <xs:attribute name = "Creation Date" type ="timePeriod"/>
     <xs:attribute name = "dateValid" type ="timePeriod"/>
     <xs:attribute name = "xs:language" type ="NMTOKEN" use ="default"
           value ="French"/>
......................
</xs:attributeGroup>
```

- use

```
 ...............

<xs:element name = "Report">

     <xs:attributeGroup ref = "Metadata"/>

......................

</xs:element>
```

XML schemas

Attribute types

- simple element types
- types derived from simple types of element (Facets, lists, forms, etc..)
- **types coming from XML DTD** *(Reserved for attributes)*
  - **ID**
  - **IDREF or IDREFS**
  - **ENTITY or ENTITIES**
  - **RATING**
  - **NMTOKEN or NMTOKENS**
  - **language** *eg en-GB, en-US, fr*
  - *etc.*

XML schemas

Deriving complex type

- **a complex type may be derived from a simple or complex type**
  - **by extension (***enriches the primitive type by adding elements and / or attributes***)**
  - **by restriction (***force the primitive type, an optional attribute is made mandatory or minOccurs is increased or maxOccurs decreased etc..***)** ***Use with caution !!***
- **indicated by the value of the attribute derivedBy**

```
<xs:element name = "Price">
        <xs:complexType base = "montantMonétaire" derived by = "extension">
                <xs:attribute name= "Currency" type = "listeDevises"/>
        </xs:complexType>
</xs:element>
```

XML schemas

XML Schema Example

```
<?xml version = "1.0" encoding= "ISO-8859-1"?>
<xs:schema          xmlns: xs= "Http://www.w3.org/1999/XMLSchema"
                    targetNamespace= "Http: // ------------- /meteoSchema"
                    xmlns: mt= "Http: // ------------- / meteoSchema"
                    xmlns = "Http: // ------------- /meteoSchema">
```

root

```
   <xs:annotation>
           <xs:documentation> Sample Schema for BDA courses
            </xs:documentation>
   </xs:annotation>
```

child-1

```
   <xs:element name= "weather" type ="mt:meteoType />
```

child-2

```
   <xs:complexType name= "meteoType">
           <xs:element name= "obs" Type ="mt:observation"
                            minOccurs= "1" maxOccurs= "unbounded"/>
    </xs:complexType>
```

XML schemas

```
<xs:complexType name= "observation">
        <xs:attribute name= "num" type ="xs:string"Use ="required"/>
        <xs:element name= "loc" type ="xs:string"/>
        <xs:element name= "Instant" type = "xs:timeInstant"/>
        <xs:element name= "temp" type ="mt: tempType"
                                            minOccurs= "0" maxOccurs= "1" />
        <xs:element name= "anemo" type ="xs:nonNegativeInteger"
                                                    minOccurs= "0" maxOccurs=
"1" />
        <xs:element name= "pluvio" type ="xs:nonNegativeInteger"
                                                    minOccurs= "0" maxOccurs=
"1" />
        <xs:element name= "nebulo" type ="mt: Octal"
                                    minOccurs= "0" maxOccurs= "1" />
        <xs:element name= "hygro" type ="mt: percent"
                                    minOccurs= "0" maxOccurs= "1" />
        <xs:element name= "Message" type = "xs: string"
                                            minOccurs= "0" maxOccurs= "
Unbound"/>
</xs:complexType>
```

XML schemas

```
<xs:simpleType name= "percent"
     <xsd restriction base = "xs:nonNegativeInteger">
              <xs:maxInclusive value = "100" />
     </xsd restriction>
</xs:simpleType>

<xs:simpleType name= "octal"
       <xsd restriction base = "xs:nonNegativeInteger">
            <xs:maxExclusive value = "8" />
     </xsd restriction>
 </xs:simpleType>

<xs:complexType name= "tempType"Base ="xs:decimal" derived by = "extention">
            <xs:minInclusive value = "- 50.0" />
            <xs:maxInclusive value = "50.0" />
            <xs:attribute name= "Unit" type = "mt: tempUnit"/>
</xs:complexType>

<xs:simpleType name= "tempUnit">
     <xs:restriction base = "xs:string">
             <xs:enumeration value = "celsius"/>
             <xs:enumeration value = "kelvin" />
             <xs:enumeration value = "farenheight"/>
    </xs:restriction>
   </xs:simpleType>
</xs:schema>
```

XML schemas

- Weather data, XML Instance

```
<?xml version = "1.0" encoding= "ISO-8859-1"?>
<weather      xmlsns = « http: // ---------- /meteoSchema">
  <obs num= "LB54476VZ32">
       <loc> Lyon-Bron </loc>
       <date> 2003-02-17T17: 30: 15 </date>
       <temp unit = "celsius"> -5.3 </temp>
       <hygro> 58 </hygro>
       <nebulo> 2 </nebulo>
       <anemo> 48 </anemo>
       <pluvio> 6 </pluvio>
  </obs>
  <obs num= "PO63276ST44">
       <loc> Paris-Orly </loc>
       <date> 2003-02-17T17: 31:
       <temp unit = "celsius"> -
       <hygro> 62 </hygro>
       <nebulo> 6 </nebulo>
       <pluvio> 0 </pluvio>
       <message> anemometer repa
  </obs>
</weather>
```

"But to be fair, there's a fifty percent chance of just about anything."

XML schemas

- non-XML content inclusion
  - A notation declaration is used to declare a type of non-XML object (sound, video, image, ...). A notation declaration has 2 attributes
    - **name :** name used in the proceeding to describe this notation *(Eg. JPEG)*
    - one of the following two attributes:
      - **public:** MIME Type Identifier *(Eg. Image / jpeg or video/ Mpeg2)*
      - **or system:** URL component that supports treatment.
  - a notation can be referenced in a complex type declaration

XML schemas

- *Example*

```
<! - notation Declaration ->
<xs:notation name = Jpeg public = 'image / jpeg' />
............................
<! - Declaring an element containing a non-XML object ->
<xs:element name = 'Photography'>
   <xs:complexType base = "xs:binary" derivedBy = "Extension">
          <xs:attribute name = "Phototype" type = "xs:notation"/>
           <xs:encoding value = 'hex'/>
   </xs:complexType
</xs:element
........................
```

In the instance we can have

```
<photography PicType = "Jpeg">
       O2A1 97BE AABE FE07 ..... AABF
      ..............................................
      EEBE O221 9743 6390 FE08 ..... </ photography>
```

You can also use the entity type attributes

XML schemas

- Many other concepts (not covered in this course)
  - items and attribute wildcards
  - mixed content, sequence, choice and item groups
  - items and abstract types
  - classes of equivalent elements
  - inclusion scheme
  - import types and elements
  - etc.

: consult the bibliography

XML schemas sum up

- A Schema is an XML document (a DTD is not)

- Because it is an XML document, it must have a root element
    - The root element is `<schema>`

- Within the root element, there can be
    - Any number and combination of
        - Inclusions
        - Imports
        - Re-definitions
        - Annotations

    - Followed by any number and combinations of
        - Simple and complex data type definitions
        - Element and attribute definitions
        - Model group definitions
        - Annotations

DTD and XML Schemas

- The two technologies coexist
- DTD easier and faster
- Schemes are more comprehensive but verbose

Xhtml, DTD and schema :

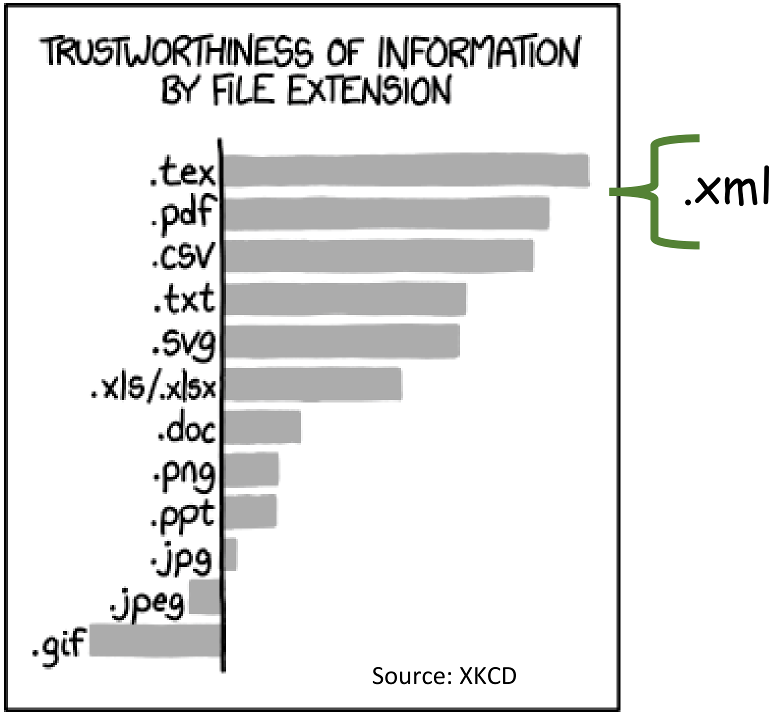
https://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd

https://www.w3.org/2002/08/xhtml/xhtml1-strict.xsd

http://www.w3.org/TR/2002/WD-SVG11-20020108/SVG.xsd

Plan

- Introduction
- Core XML
  - Introduction to XML
  - DTD
  - Element XML
  - XML attribute
  - Reusable Objects: XML Entities
  - XML instance and example
  - Namespaces
  - XML schemas
  - Bibliography
- XML Galaxy
- NOSQL
- Conclusion



Source: XKCD

Bibliography

- The reference : **http://www.w3.org/TR/**

- A site to understand and take control quickly on some notions : **http: //www.w3schools.com/**

- Online courses :
  - **https://www.tutorialspoint.com/xml/xml_schemas.htm**
  - **https://www.irif.fr/~carton/Enseignement/XML/Cours/support.pdf**