

eXtensive Markup Language

Le cas du modèle HyperTopic

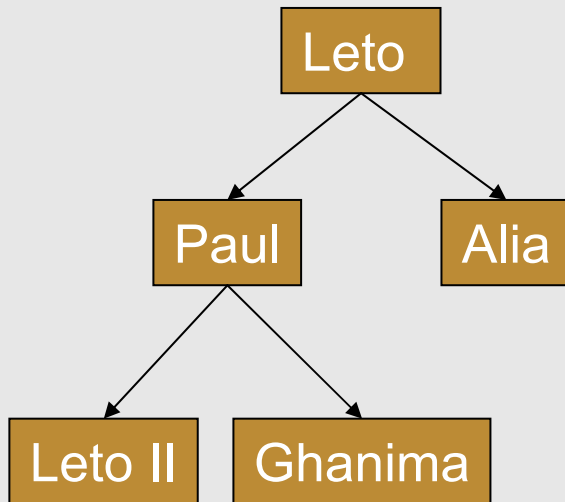
Aurélien Bénel

*Systèmes d'information, management des connaissances
et communication*

Bureau T107, aurelien.benel@utt.fr

- La s erialisation d'un arbre

Ex : Arbre g en alogue



```
<?xml version="1.0" encoding="utf-8" ?>
<enfants de="Leto">
  <enfants de="Paul">
    <enfants de="Leto II">
    </enfants>
    <enfants de="Ghanima" />
  </enfants>
  <enfants de="Alia" />
</enfants>
```

Un peu de vocabulaire

Entête

Nom d'élément

Nom d'attribut

Valeur d'attribut

```
<?xml version="1.0" encoding="utf-8" ?>
<enfants de = "Leto" >
  <enfants de = "Paul" >
    <enfants de = "Leto II" >
      </enfants>
    <enfants de = "Ghanima" />
  </enfants>
  <enfants de = "Alia" />
</enfants>
```

Élément **racine**

Élément

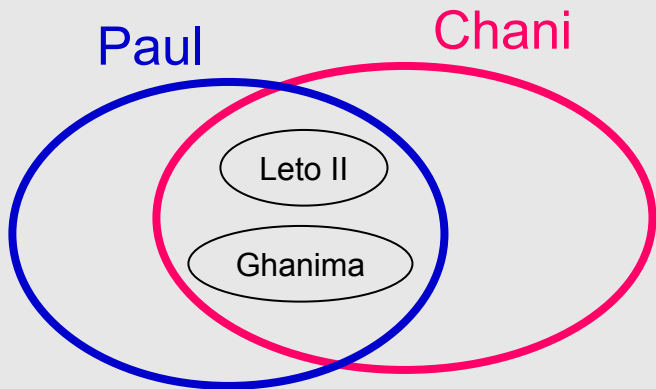
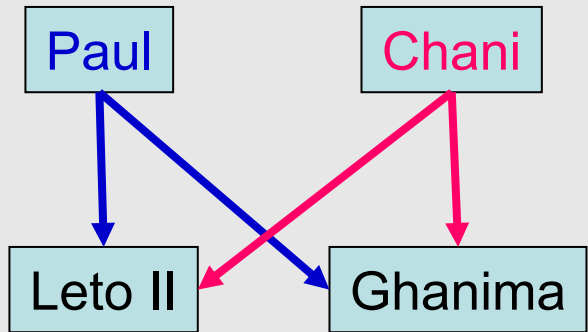
Élément **vide**

Élément **vide**

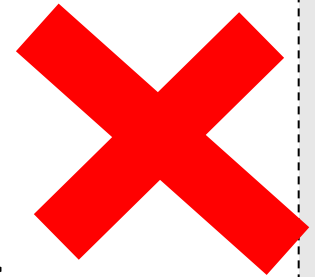
Élément vide

« Bien formé » : emboîtement des éléments

4

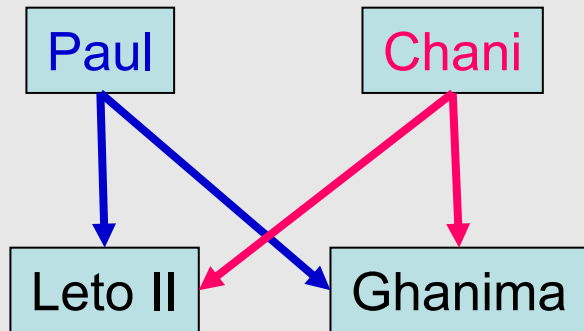


```
...  
<homme nom="Paul">  
<femme nom="Chani">  
  <homme nom="Leto II" />  
  <femme nom="Ghanima" />  
</homme>  
</femme>  
...
```

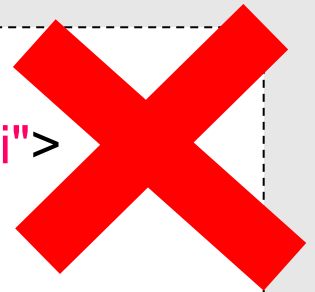


« Bien formé » : un attribut n'apparaît qu'une fois par élément

5



```
...  
<enfants de="Paul" de="Chani">  
  <enfants de="Leto II" />  
  <enfants de="Ghanima" />  
</enfants>  
...
```



Et le texte dans tout ça ?

6

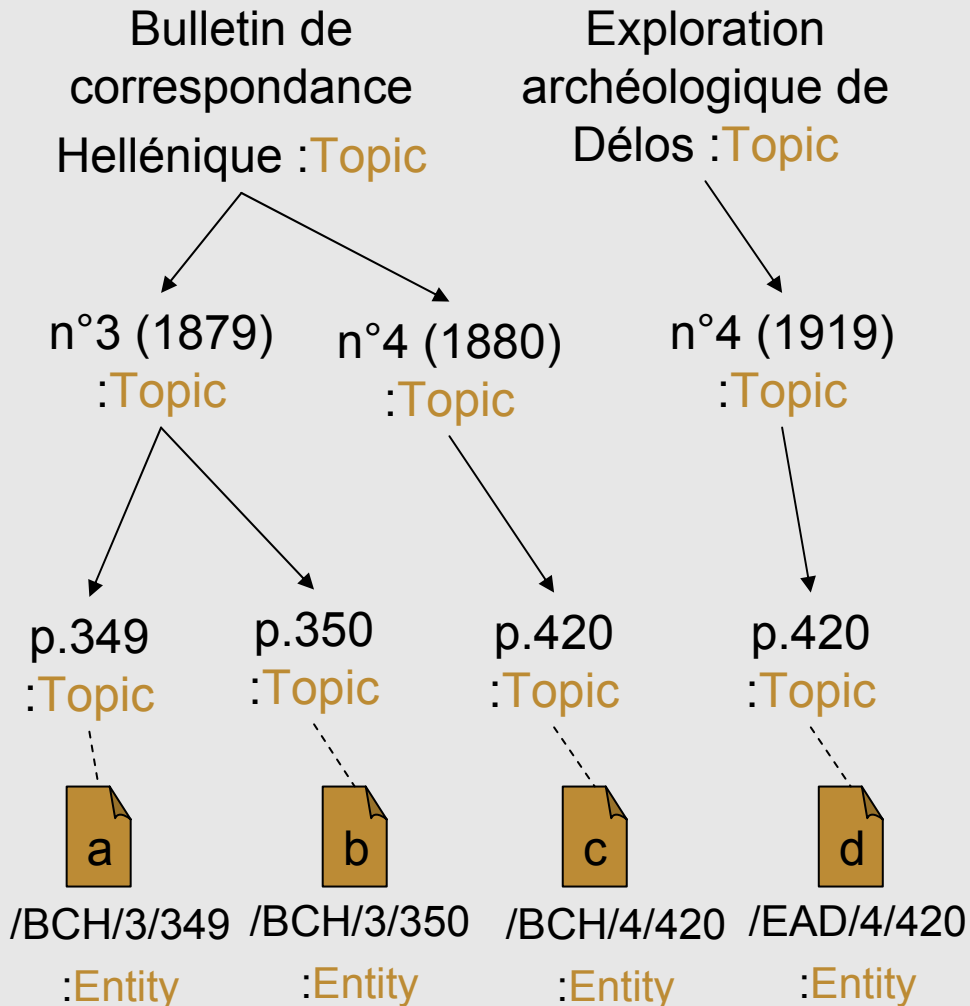
PCDATA :
Données textuelles
analysées

```
...  
<enfants de="Paul" avec="Chani" >  
  Leur premier fils fut tué par les  
  Harkonnens. Ensuite naquirent deux  
  jumeaux : un garçon (  
  <enfants de="Leto II" />  
  ) et une fille (  
  <enfants de="Ghanima" />  
  ).  
</enfants>  
...
```

Export : L'exemple d'HyperTopic

7

Point de vue de l'imprimeur : **ViewPoint**

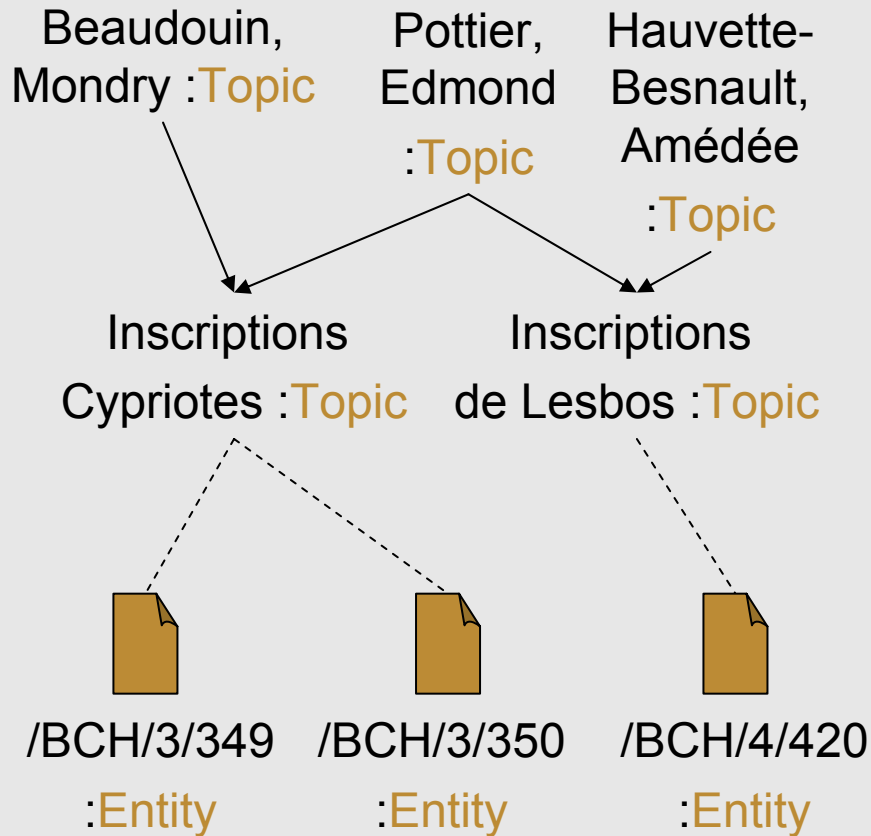


```
<viewpoint name="Point de vue de l'éditeur">
  <topic name="Bulletin de correspondan...">
    <topic name="n°3 (1879)">
      <topic name="p.349">
        <entity href="http://.../BCH/3/349/" />
      </topic>
      <topic name="p.350">
        <entity href="http://.../BCH/3/350/" />
      </topic>
    </topic>
    <topic name="n°4 (1880)">
      <topic name="p.420">
        <entity href="http://.../BCH/4/420/" />
      </topic>
    </topic>
  </topic>
  <topic name="Exploration archéologiqu...">
    <topic name="n°4 (1919)">
      <topic name="p.420">
        <entity href="http://.../EAD/4/420/" />
      </topic>
    </topic>
  </topic>
</viewpoint>
```

Export : L'exemple d'HyperTopic (suite)

8

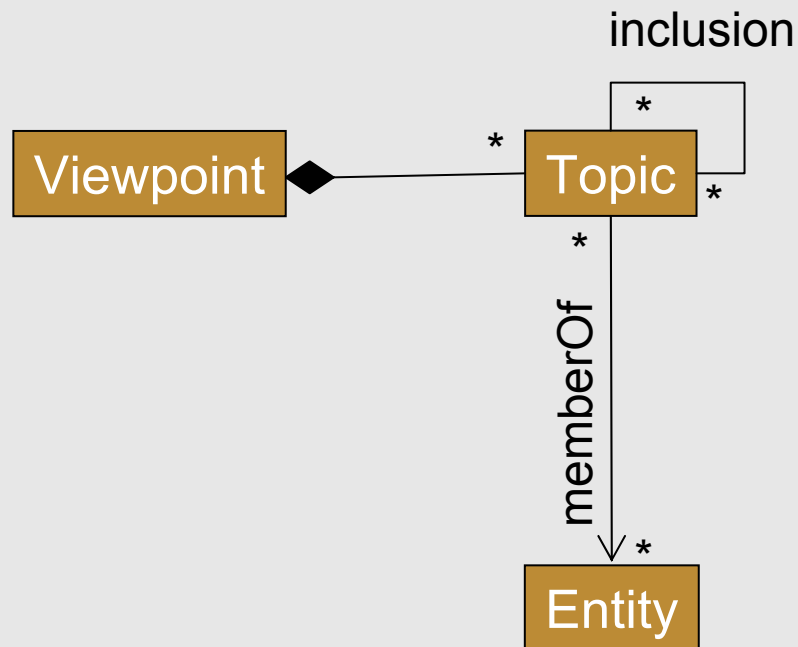
Point de vue du bibliothécaire : **ViewPoint**



```
<viewpoint name="Point de vue du bibliothécaire">
  <topic name="Beaudouin, Mondry">
    <topic id="c1" name="Inscriptions Cypriotes" >
      <entity href="http://.../BCH/3/349"/>
      <entity href="http://.../BCH/3/350"/>
    </topic >
  </topic >
  <topic name="Hauvette-Besnault...">
    <topic id="c2"
      name="Inscriptions de Lesbos">
      <entityRef href="http://.../BCH/4/420"/>
    </topic >
  </topic >
  <topic name="Pottier, Edmond"
    subtopics="c1 c2"
  />
</viewpoint>
```

■ « Document Type Definition »

```
<!ELEMENT viewpoint (topic)*>
<!ELEMENT topic      (topic|entity)*>
<!ELEMENT entity     EMPTY>
<!ATTLIST viewpoint name CDATA #IMPLIED>
<!ATTLIST topic
  id ID #IMPLIED
  name CDATA #IMPLIED
  subtopics IDREFS #IMPLIED
>
<!ATTLIST entity
  href CDATA #REQUIRED
>
```



- Contrôle pour chaque instance d'élément :
 - De sa définition en termes d'éléments
 - De sa définition en termes d'attributs
 - Du type de ses d'attributs (chaîne, identifiant, énumération...)
 - De l'unicité de ses ID dans le document
 - De l'intégrité référentielle de ses IDREF(S) dans le document
- Mais :
 - Typage très faible
 - Références globales
 - Pas de contraintes plus complexes

Alternative du W3C : "XML Schema"

11

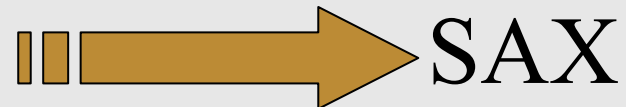
```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="entity">
    <xs:complexType>
      <xs:attribute name="href" type="xs:anyURI" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="topic">
    <xs:complexType>
      <xs:choice>
        <xs:element ref="topic" />
        <xs:element ref="entity" />
      </xs:choice>
      <xs:attribute name="subtopics" type="xs:IDREFS" use="optional" />
      <xs:attribute name="name" type="xs:string" use="optional" />
      <xs:attribute name="id" type="xs:ID" use="optional" />
    </xs:complexType>
  </xs:element>
  <xs:element name="viewpoint">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="topic" />
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" use="optional" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Alternative de l'OASIS : “Relax NG” (ISO)

```
start = element viewpoint {
    attribute name { text }?,
    topic*
}
topic = element topic {
    attribute id { xsd:ID }?,
    attribute name { text }?,
    attribute subtopics { xsd:IDREFS }?,
    (topic | entity)*
}
entity = element entity {
    attribute href { xsd:anyURI },
    empty
}
```

- Deux API sont disponibles dans la plupart des langages :
 - *Simple API for XML (SAX)*
Génère des événements au cours de la lecture (linéaire) du fichier XML.
 - *Document Object Model (DOM)*
Permet de "naviguer" dans une représentation "objet" du fichier XML.

- Fichiers pouvant être volumineux
 - Éviter le chargement en mémoire
- Navigation
 - Trouver le père (une structure de pile suffit)
 - Trouver les éléments référencés (une structure de dictionnaire –Map– suffit)
 - Le reste est linéaire



Analyse avec SAX (en Java)

15

```
...
SAXParserFactory parserFactory = SAXParserFactory.newInstance();

SchemaFactory schemaFactory =
SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
parserFactory.setSchema(
    schemaFactory.newSchema("hypertopic.xsd")
);

SAXParser parser = parserFactory.newSAXParser();
parser.setContentHandler(new Importer());
parser.parse(file);
...
```

Détail de l'analyse avec SAX

```
class Importer extends DefaultHandler {  
    //////////////////////////////////////  
    public void startElement (String u, String n, String name, Attributes attributes) throws SAXException {  
        if (name.equals("topic" ) ) {  
            String label = attributes.getValue("name");  
            ...  
        } else if (name.equals("entity")) {  
            String ressource = attributes.getValue("resource");  
            ...  
        }  
    }  
    //////////////////////////////////////  
}
```

Importer.java

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="dlib">
    <viewpoint name="Revue">
      <xsl:apply-templates select="series"/>
    </viewpoint>
  </xsl:template>

  <xsl:template match="series">
    <topic name="{@title}" >
      <xsl:apply-templates />
    </topic>
  </xsl:template>
```

```
<xsl:template match="issue">
  <topic name="n°{@number} ({@year})" >
    <xsl:apply-templates />
  </topic>
</xsl:template>
```

```
<xsl:template match="page">
  <topic name="p.{@number}">
    <entity href="{@fac-simile}" />
  </topic>
</xsl:template>
```

```
...
</xsl:stylesheet>
```

En savoir plus... sur XPATH

19

<code>/truc</code>	Élément racine "truc"
<code>..</code>	Élément père de l'élément courant
<code>.</code>	Élément courant
<code>truc</code>	Élément "truc", fils de l'élément courant
<code>@machin</code>	Attribut "machin" de l'élément courant
<code>truc/machin</code>	Élément "machin", fils de "truc", lui-même fils de l'élément courant
<code>./machin</code>	Élément "truc", descendant de l'élément courant
<code>truc[@chose="machin"]</code>	Élément "truc", fils de l'élément courant, ayant un attribut "chose" dont la valeur est "machin"
<code>id(@machin)</code>	Élément ayant pour identifiant la valeur de l'attribut "machin"

<pre><xsl:value-of select="monExpressionXpath" ></pre>	Insérer la valeur (texte sans balise) d'une expression Xpath
<pre><xsl:for-each select="monExpressionXpath"> Corps de la boucle </xsl:for-each></pre>	Répéter le corps de la boucle pour chaque noeud correspondant à l'expression Xpath
<pre><xsl:sort select="monExpressionXpath" data-type="text number" order="ascending descending" ></pre>	Trier des éléments (juste après le début d'un for-each)