# Données Semi-Structurées:

# XML, XPath, DTD

Enseignant: *Dario COLAZZO*
Chargée de TD/TP: *Beatrice NAPOLITANO*

18 Janvier 2020

## 1 Rappel

**Rappel 1 (document XML bien formé)** *:*
*Un document XML (eXstensible Markup Language) bien formé a la bonne syntaxe XML.*
*Les règles de syntaxe sont :*

- *les documents XML doivent avoir un préambule et un seul élément racine*
- *les éléments doivent avoir une balise de fermeture*
- *les éléments doivent être correctement imbriquées*
- *les valeurs d'attribut XML doit être cité*
- *les caractères* `<` *et* `&` *sont strictement illégale et doivent être échappés*

*Ces cinq entités sont définies par défaut :* `<`, `>`, `&`, `'` *and* `"`*. Ils ont des références d'entités prédéfinies :*

| | | |
|---|---|---|
| `<` | `<` | *less than* |
| `>` | `>` | *greater than* |
| `&` | `&` | *ampersand* |
| `&apos;` | `'` | *apostrophe* |
| `&quot;` | `"` | *quotation mark* |

**Rappel 2 (document XML valide)** *:*
*Un document XML est valide quand :*

- *il est bien formé*

*et*

- *il est également conforme aux règles d'une Document Type Definition (DTD).*

*On dit que le document XML est valide par rapport à la DTD. Le DTD peut être externe ou interne.*

**Rappel 3 (syntaxe DTD)** *:*
*DTD externe :*

```
1: <?xml version="1.0"?>
2: <!DOCTYPE bib SYSTEM "bib.dtd">
3: <bib>
4:   ...
5: <bib>
```

*DTD interne :*

```
1: <?xml version="1.0"?>
2: <!DOCTYPE bib [
3: <!ELEMENT bib (book*)>
4: <!ELEMENT book ...>
5: ...
6: ]>
7: <bib>
```

```
8:    ...
9: <bib>
```

**Rappel 4 (Document Type Definition)** *:*
*DTD : ensemble de declaration spécifiant la structure d'un type d'élément ou d'attribut.*

1) *Type élément : tag + structure contenu (expression régulière)*
   `<!ELEMENT tag structure>`

   *Expression régulière :*
   — `A,B` *: un* `A` *suivi d'un* `B`
   — `A*` *: 0 ou plusieurs* `A`
   — `A?` *: 0 ou 1* `A`
   — `B+` *: 1 ou plusieurs* `B`
   — `(A|B)` *: un* `A` *ou* `B` *dans n'importe quel ordre*
   *où* `A` *et* `B` *peut être autres éléments ou #PCDATA (chaine de caractère).*
   *La structure peut aussi être EMPTY soit ANY.*

2) *Type attribut : tag + type contenu + contrainte*
   `<!ATTLIST tag_element tag_attribut type_contenu contrainte>`

   *Le type contenu peut être :*
   — *CDATA*
   — *ID*
   — *IDREF*
   — *s1,....,sn*

   *La contrainte peut être :*
   — *#REQUIRED*
   — *#IMPLIED*
   — *#FIXED value*

***Example :***
— ***#REQUIRED***
*DTD :*
```
1: <!ELEMENT contact EMPTY>
2: <!ATTLIST contact number CDATA #REQUIRED>
```
*XML :*
```
1: <contact number="06-12345678" />
```
*si nous écrivons*
```
1: <contact />
```
*c'est une erreur.*

— ***#IMPLIED***
*DTD :*
```
1: <!ELEMENT contact EMPTY>
2: <!ATTLIST contact fax CDATA #IMPLIED>
```
*XML :*
```
1: <contact fax="555-667788" />
```
*si nous écrivons*
```
1: <contact />
```
*est correcte et la valeur du champ fax est nulle.*

— ***#FIXED value***
*DTD :*

```
1: <!ELEMENT sender EMPTY>
2: <!ATTLIST sender person CDATA #FIXED ‘‘Beatrice’’>
```

*XML :*

```
1: <sender person=‘‘Beatrice’’ />
2: <sender />
```

*sont ok mais si nous écrivons*

```
1: <sender person=‘‘Dario’’ />
```

*c'est une erreur.*

— ***default value***

*DTD :*

```
1: <!ELEMENT square EMPTY>
2: <!ATTLIST square width CDATA "0">
```

*XML :*

```
1: <square width="100" />
```

*la valeur de width est 100, mais si on ne le spécifie pas*

```
1: <square />
```

*c'est correct e la valeur est 0.*

**Rappel 5 (Expressions XPath)** *:*

`[/]étape /étape /.../étape`

*Une étape : trois composants* `axe::filtre[prédicat_1]...[prédicat_n]`

— *L'axe : sens de parcours des nœuds*
  — *ancestor*
  — *ancestor-or-self*
  — *attribute*
  — *child*
  — *descendant*
  — *descendant-or-self*
  — *following*
  — *following-sibling*
  — *namespace*
  — *parent*
  — *preceding*
  — *preceding-sibling*
  — *self*

— *Le filtre : type des nœuds qui seront retenus*
  — `child::book` *les fils de nœud contexte ayant tag book*
  — `child::node()` *tout les fils du nœud contexte*
  — `child::*` *tout les fils élément du nœud contexte*
  — `child::text()` *tout les fils texte du nœud contexte*

— *Le(s) prédicat(s) : propriétés que doivent satisfaire les nœuds retenus par* `axe::filtre`

# 2 Links

Validateurs en ligne de documents XML :
— http://www.w3schools.com/xml/xml_validator.asp
— http://www.xmlvalidation.com/

# 3 Exercice

**Exercice 1 (XML)** *:*
*Décider si les documents XML suivants sont correctes ? Sinon, trouvez les erreurs et les corriger.*

1.
```
<?xml version="1.0" encoding="UTF-8"?>
<html>
  <head><title>Hello, World</title></head>
  <body><p>Hello, World</p></body>
</html>
```

2.
```
<?xml version="1.0" encoding="UTF-8"?>
<p>This is a test. This is a test of the <em>
   <strong>Emergency</em> Broadcast System.</strong></p>
```

3.
```
<?xml version="1.0" encoding="UTF-8"?>
<note date="12/11/2007">
  <!-- This is a comment -->
  <Message><to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body></message>
  </note>
  <note date="13/11/2007">
  <message><to>Jani</to>
  <from>Tove</from>
  <heading>Re: Reminder</heading>
  <body>Ok!</body></message>
</note>
```

4.
```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="points">
    <xs:complexType>
      <xs:sequence>
         <xs:element maxOccurs="unbounded" name="point">
           <xs:complexType>
             <xs:attribute name="x" type="xs:unsignedShort" use="required" />
             <xs:attribute name="y" type="xs:unsignedShort" use="required" />
           </xs:complexType>
         </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

5.
```
<?xml version="1.0" encoding="UTF-8"?>
<html>
<head><title>Paragraphs</title></head>
<body><p>This is a paragraph.<br/>
<p>This is another paragraph.<br/>
<p>Third paragraph.</body>
</html>
```

6.
```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about=http://www.AcronymFinder.com/>
  <dc:title>Acronym Finder</dc:title>
  <dc:description>The Acronym Finder is a world wide
    web (WWW) searchable database of more than 169,000
    abbreviations and acronyms about computers,
    technology, telecommunications, and military
    acronyms and abbreviations.</dc:description>
  <dc:subject>
    <rdf:Bag>
      <rdf:li>Astronomy</rdf:li>
      <rdf:li>Literature</rdf:li>
      <rdf:li>Mathematics</rdf:li>
      <rdf:li>Music</rdf:li>
      <rdf:li>Philosophy</rdf:li>
    </rdf:Bag>
  </dc:subject>
  </rdf:Description>
</rdf:RDF>
```

7.
```
<?xml version="1.0" encoding="UTF-8"?>
<html><body>
  <p><b><i>This paragraph is bold and italic.</b></i></p><br/>
  <p><i><b>This paragraph is italic and bold.</i></b></p><br/>
</body></html>
```

8.
```
<catalog>
  <work type='prose' date='1906'>
    <title>The Gift Of The Magi</title>
    <author>O. Henry</author>
  </work>
  <work type='poem' date='1845'>
    <title>The Raven</title>
    <author>Edgar Allen Poe</author>
  </work>
  <work type='play' date='1601'>
    <title>Hamlet</title>
    <author>William Shakespeare</author>
  </work>
</catalog>
```

9.
```
<?xml version="1.0" encoding="UTF-8"?>
<letter>
  <date>December 11, 2002</date>
  <addressee>
    <name>Melvile Dewey</name>
    <address_one>
    Columbia University</address_one><address_two>New York, NY
    </address_two>
  </addressee>
  <greeting>Dear Melvile,</greeting>
  <paragraph>I have been reading your ideas concerning nature of
    librarianship, and <italics>I find them very intriguing</italics>.
    I would love the opportunity to discuss with you the role of the
    card catalog in today's libraries considering the advent to World
```

```
    Wide Web. Specifically, how are things like Google and Amazon.com
    changing our patrons' expectations of library services? Mr. Cutter
    and I will be discussing these ideas at the next Annual Meeting,
    and we are available at the follow dates/times:</paragraph>
  <list>
    <item>Monday, 2-4</item>
    <item>Tuesday, 3-5</item>
    <item>Thursday, 1-3</item>
  </list>
  <paragraph>We hope you can join us.</paragraph>
  <closing>Sincerely, S. R. Ranganathan</closing>
</letter>
```

10.

```
<?xml version="1.0"?>
<dictionary>
 <word>
  <update date="2002-12-23"/>
  <name is_acronym="true">XML</Name>
  <description>eXtensible Markup Language</description>
 </word>
 <word>
  <update date="2002-12-23"/>
  <name is_acronym="true">POP</name>
  <definition default>Post Office Protocol</definition>
  <definition>Point Of Purchase</definition>
</dictionary>
```

11.

```
<?xml version="1.0" encoding="UTF-8"?>
<domain type='kvm>
  <name>domain</name><
  <memory>524288</memory>
  <vcpu>2</vcpu>
  <features><acpi/><pae/>
  <clock offset='utc'>
  <disk type='block' device='cdrom'>
    <driver name='qemu' type='raw'/>
    <source file='/path/to/image.iso'/>
    <tar get dev='hdc' bus='ide'/>
    <readonly/></name>
  </disk>
</domain>
```

12.

```
<?xml version="1.0" encoding="UTF-8"?>
<name>Oyster Soup</name>
<author>Eric Lease Morgan</author>
<copyright holder=Eric Lease Morgan>&copy; 2003</copyright>
<ingredients>
<list>
<item>1 stalk of celery
<item>1 onion
<item>2 tablespoons of butter
<item>2 cups of oysters and their liquor
<item>2 cups of half & half
</list><cost>total cost < 36 euro </cost>
</ingredients>
<process><P>Begin by sauteing the celery and onions in butter until soft.
```

```
15: Add oysters, oyster liquor, and cream. Heat until the oysters float.
16: Serve in warm bowls.</p>
17: <p><i>Yummy!</p></i>
18: </process>
```

**Exercice 2 (DTD from XML)** *:*
*Écrivez une DTD pour le document XML suivant :*

```
1: <?xml version="1.0"?>
2: <shiporder orderid="889923">
3:   <orderperson>John Smith</orderperson>
4:   <shipto>
5:     <name>Ola Nordmann</name>
6:     <address>Langgt 23</address>
7:     <city>4000 Stavanger</city>
8:     <country>Norway</country>
9:   </shipto>
10:  <item>
11:    <title>Empire Burlesque</title>
12:    <note>Special Edition</note>
13:    <quantity>1</quantity>
14:    <price>10.90</price>
15:  </item>
16:  <item>
17:    <title>Hide your heart</title>
18:    <quantity>1</quantity>
19:    <price>9.90</price>
20:  </item>
21: </shiporder>
```

**Exercice 3 (XML from DTD)** *:*
*Écrivez un document XML basé sur la DTD suivante :*

```
1: <?xml version="1.0"?>
2: <!DOCTYPE stock[
3: <!ELEMENT stock (new-car | used-car)*>
4: <!ELEMENT new-car (model, price)>
5: <!ELEMENT used-car (model, price, mileage, condition?)>
6: <!ELEMENT model (#PCDATA)>
7: <!ELEMENT price (#PCDATA)>
8: <!ELEMENT mileage (#PCDATA)>
9: <!ELEMENT condition (#PCDATA)>
10: ]>
```

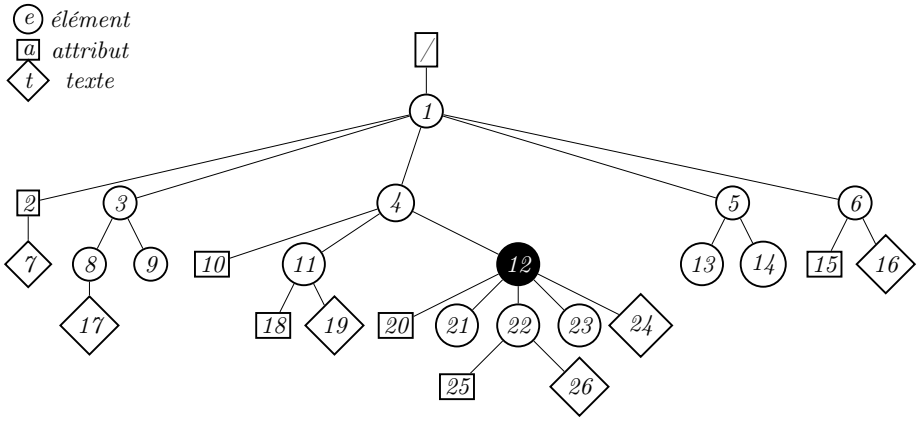
**Exercice 4 (XML & DTD)** *:*
*Envisager une application dans laquelle les résultats des matchs de football doivent être représentés en XML.*
*Pour chaque* ***jeu****, nous voulons être en mesure de représenter* ***les deux équipes*** *impliquées, l'équipe qui jouait* ***chez eux****, quels* ***joueurs*** *ont marqué des buts (dont certains peuvent avoir été* ***pénalités****) et le* ***moment*** *où chacun a été marqué, et quels joueurs ont reçu des* ***cartons jaunes*** *et* ***rouges****.*

*Écrivez un document XML et une DTD pour cette application. Vous pouvez utiliser des atributes.*

*Assurez-vous que votre document est bien écrit et valide à l'aide d'un programme qui valide votre document.*

**Exercice 5 (XPath – axe)** *:*

*Regardez l'arbre XML ci-dessous. Notez les nœuds couverts par chaque axe, à partir de noeud 12.*




— *ancestor :*
— *ancestor-or-self :*
— *attribute :*
— *child :*
— *descendant :*
— *descendant-or-self :*
— *following :*
— *following-sibling :*
— *parent :*
— *preceding :*
— *preceding-sibling :*
— *self :*

**Exercice 6 (XPath)** *:*
*Expliquez la signification des expressions suivantes :*
— `//COURS[INTITULE='XML']`
— `//COURS[INTITULE=XML]`
*Donnez une instance pour laquelle le résultat est identique.*

**Exercice 7 (XPath)** *:*
*Expliquez la différence entre les deux expressions suivantes, et donnez un document pour laquelle on n'obtient pas le même résultat :*
— `//B[position() = 1]`
— `/descendant::B[position() = 1]`

**Exercice 8 (XPath)** *:*
*Considèrez la DTD suivante pour le document Films.xml dècrivant des donnèes cinèmatographiques :*

```
1: <!DOCTYPE FILMS [
2: <!ELEMENT FILMS (FILM+, ARTISTE+)>
3: <!ELEMENT FILM (TITRE, GENRE, PAYS, MES, ROLES, RESUME?)>
4: <!ELEMENT TITRE (#PCDATA)>
```

```
<!ATTLIST FILM Annee CDATA #REQUIRED>
<!ELEMENT GENRE (#PCDATA)>
<!ELEMENT PAYS (#PCDATA)>
<!ELEMENT MES (#PCDATA)>
<!ATTLIST MES id_mes IDREF #IMPLIED>
<!ELEMENT ROLES (ROLE*)>
<!ELEMENT ROLE (PRENOM, NOM, INTITULE)>
<!ELEMENT PRENOM (#PCDATA)>
<!ELEMENT NOM (#PCDATA)>
<!ELEMENT INTITULE (#PCDATA)>
<!ELEMENT RESUME (#PCDATA)>
<!ELEMENT ARTISTE (ACTNOM, ACTPNOM, ANNEENAISS)>
<!ATTLIST ARTISTE id_art ID #REQUIRED>
<!ELEMENT ACTNOM (#PCDATA)>
<!ELEMENT ACTPNOM (#PCDATA)>
<!ELEMENT ANNEENAISS (#PCDATA)>
]>
```

*Appliquez à ce document les expressions XPath pour rechercher les informations suivantes :*

1. *La liste des titres de films.*
2. *Les titres des films parus en 1990.*
3. *Le résumé d'Alien.*
4. *Les titre des films avec Bruce Willis.*
5. *Les titres des films qui ont un résumé.*
6. *Les titres des films qui n'ont pas de résumé.*
7. *Les titres des films vieux de plus de trente ans.*
8. *Quel rôle joue Harvey Keitel dans Reservoir dogs ?*
9. *Quel est le dernier film du document ?*
10. *Quel est le titre du film qui précède immédiatement Shining (dans l'ordre du document) ?*
11. *Qui a mis en scène Vertigo ?*
12. *Les titres des films qui contiennent un « V » (utiliser la fonction contains())*
13. *Les noeuds qui ont exactement trois descendants (utiliser la fonction count()).*
14. *Les noeuds dont le nom contient la chaîne « TU » (fonction name())*