# TD1

## Exercice 1 : document bien formé

Observez le document XML suivant :

```
<?xml version="1.0"?>
<!-- this is a note -->
<note date=3 janvier>
  <to>Bob</To>
  <from>Alice</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
<note date="5 janvier" <!-- this is another note --> >
  <to>Alice</to>
  <from>Bob
  <body>No problem & see you soon</body>
</note>
<note />
```

1. Ce document est-il bien formé (i.e. respecte-t-il la syntaxe XML) ?
2. S'il ne l'est pas, corrigez les erreurs.

## Exercice 2 : utilisation d'une DTD

On dispose de la DTD cdtheque.dtd :

```
=========================
<!ELEMENT cdtheque (cd*) >

<!ATTLIST cdtheque date CDATA #REQUIRED>
<!ATTLIST cdtheque auteur CDATA #REQUIRED>

<!ELEMENT cd  (titre,artiste,style?)>

<!ELEMENT titre        (#PCDATA)>
<!ELEMENT style        (#PCDATA)>
<!ELEMENT artiste      (#PCDATA)>
=========================
```

1. Créez un document XML vérifiant cette DTD en incluant la DTD dans le document.
2. Créez un document XML vérifiant cette DTD en externalisant la DTD.

## Exercice 3 : création d'une DTD

On souhaite gérer une filmographie. Quelle va être la structure du document XML associé à un film ?
Créer la DTD correspondante.

## Exercice 4 :

Ecrire un document XML modélisant votre Curriculum Vitae (réfléchir à ce qui doit être mémorisé et comment structurer l'information).
Ecrire la DTD associée.

# XML TD1 - proposition de correction

## Exercice 1 : document bien formé

Erreurs dans les lignes :

3 : guillemets
4 : casse
9 : une seul racine à l'arborescence, rajouter un élément général document
10 : commentaire à sortir des tags
13 : & interdit
14 : mismatched tag, en fait l'erreur vient de plus haut, il manque la fin du tag <from>

Le document corrigé :

```
<?xml version="1.0"?>
<!-- this is a note -->
<document>
<note date="3 janvier">
<to>Bob</to>
<from>Alice</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
<note date="5 janvier">
<!-- this is another note -->
<to>Alice</to>
<from>Bob</from>
<body>No problem, see you soon</body>
</note>
```

<note/>
</document>

## Exercice 2 : utilisation d'une DTD

Exo du même genre qu'en TP.
Ici attributs de cdtheque obligatoires (#REQUIRED).
0 ou 1 occurrence de style.

...
<cdtheque date='3 avril 2005' auteur='moi'>
<cd>
<titre>Remasters</titre>
<artiste>Led Zeppelin</artiste>
</cd>
<cd>
<titre>Best Of</titre>
<artiste>Michel Sardou</artiste>
<style>français</style>
</cd>
...

cf. correction du TP pour l'inclusion interne ou externe de la DTD.

## Exercice 3 : création d'une DTD

**Eléments de réponse :**

**Essayer d'amener progressivement les étudiants à un résultat de ce type. Plusieurs formes sont possibles. Discuter le fait d'avoir beaucoup d'éléments ou d'introduire des attributs...**

Objectif : créer une DTD de filmographie. Lorsque vous planifiez votre DTD, il est utile de commencer par griffonner une esquisse simple.
Ce travail préparatoire aide à repérer les relations entre les éléments et les types de données à déclarer, et évite aussi les oublis.
Ne pas trop se soucier de la structure des données en élaborant cette esquisse, vous pourrez réorganiser ces données avant de créer la DTD. Dressez d'abord une liste simple.

*1ère étape : liste des éléments*

Pour notre DTD de filmographie, nous établissons d'abord la liste des éléments clés (éventuellement avec quelques notes concernant la structuration envisageable des données).

Titre
Année
Genre
Réalisateur
Distributeur
Acteurs
Musique
Durée
Pays
Langue
Couleur
Récompenses

*2ème étape : création des éléments XML*

Chacun de ces points peut représenter un élément XML.

Pour chaque élément sont spécifiés son nom, tous les attributs qu'il possède et le type de données qu'il contient.

Déclaration d'élément de base :
<!ELEMENT nom_element (type de données)>

Un élément sera déclaré pour film de la manière suivante :
<!ELEMENT film (titre, annee, genre, realisateur, acteurs, musique)>

**Proposition de correction :**

On fait évoluer certains éléments en attributs pour clarifier la structure.

<!ELEMENT film (titre, annee, genre, realisateur)>
<!ELEMENT titre (#PCDATA)>
<!ATTLIST titre langue CDATA #IMPLIED>
<!ATTLIST titre titre_de_remplacement CDATA #IMPLIED>
<!ATTLIST titre pays CDATA #IMPLIED>
<!ATTLIST titre recompenses CDATA #IMPLIED>
<!ATTLIST titre duree #IMPLIED>
<!ELEMENT annee (#PCDATA)>
<!ATTLIST annee academy_awards CDATA #IMPLIED>
<!ATTLIST annee distributeur CDATA #IMPLIED>
<!ELEMENT genre (#PCDATA)>
<!ATTLIST genre categorie CDATA #IMPLIED>
<!ATTLIST genre medium CDATA #IMPLIED>
<!ELEMENT realisateur (#PCDATA)>
<!ATTLIST realisateur directeur_de_la_photographie CDATA #IMPLIED>
<!ATTLIST realisateur cameraman CDATA #IMPLIED>
<!ATTLIST realisateur monteur CDATA #IMPLIED>
...

Ainsi, nous aurons la structure suivante pour le document XML filmographique :

<film>

```
   <titre></titre>
   <annee></annee>
   <genre></genre>
   <realisateur></realisateur>
   ...
</film>
```

### Exercice 4 :

Voir l'exemple de CV dans la correction du TP

# TD2 : DOM

## Exercice 1 :

Soit le document XML suivant (sous forme sérialisée) :

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!-- Commentaire --!>
<A>Le texte de A
   <B>Le texte de B</B>
   <D attr1="1" attr2="azerty">
     <C/>
   </D>
   <![CDATA[2x < y ]]>
</A>
```

Dessiner l'arbre DOM correspondant à ce document.

## Exercice 2 :

Ecrire un programme java créant l'arbre d'éléments du document passé en argument selon le DOM Document Object Model, puis parcourir cet arbre pour afficher les noeuds et leurs caractéristiques.
Indications : les différents types de noeuds sont
**`ATTRIBUTE_NODE, CDATA_SECTION_NODE,COMMENT_NODE,`**
**`DOCUMENT_FRAGMENT_NODE, DOCUMENT_NODE, DOCUMENT_TYPE_NODE,`**
**`ELEMENT_NODE,`**
**`ENTITY_NODE, ENTITY_REFERENCE_NODE, NOTATION_NODE,`**
**`PROCESSING_INSTRUCTION_NODE, TEXT_NODE.`**

Utiliser les méthodes :
**getNodeType(), getNodeName(), getNodeValue(),hasChildNodes(),getFirstChild(), getNextSibling()** sur les noeuds.

a) Dans un premier temps, considérer que les noeuds n'ont pas d'attributs.
b) On souhaite prendre en compte les attributs des noeuds. Utiliser **hasAttributes()** et **getAttributes()** qui récupère une liste d'attributs de type **NamedNodeMap**, dans laquelle on peut naviguer item par item tant qu'on ne dépasse pas la taille de la liste (**getLength()**) .

## Exercice 3 :

1) En utilisant le parser DOM, lire un fichier XML contenant la liste de prix d'un café nommé "Cafe Noir".
   2) Insérer dans le DOM un nouveau café, de nom"Cafe Noir" et qui coûte "3.10", juste avant le café qui s'appelle"Columbia".

   Indication : d'abord récupérer la liste des éléments nom et itérer dans la liste pour trouver "Columbia", créer un nouvel objet Node pour le nouvel élément café ainsi que de nouveaux noeuds pour les éléments nom et prix (données caractères). Utiliser la méthode NoeudTrouve.insertBefore(nouveauNoeud, NoeudTrouve).

# TD2 - proposition de correction

### Exercice 1 :

Soit le document XML suivant (sous forme sérialisée) :

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!-- Comentaire --!>
```

```
<A>Le texte de A
  <B>Le texte de B</B>
  <D attr1="1" attr2="azerty">
   <C/>
  </D>
  <![CDATA[2x < y ]]>
</A>
```

Dessiner l'arbre DOM correspondant à ce document.

## Exercice 2 : (cf. page de Didier)

Création de l'arbre d'élément du document selon le DOM Document Object Model, puis parcours de cet arbre pour afficher les noeuds et leurs caractéristiques.

### Programme

```
// Dom1.java

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.FactoryConfigurationError;
import javax.xml.parsers.ParserConfigurationException;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;
import java.io.File;
import java.io.IOException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.DOMException;

public class Dom1
{
  public static void main (String argv [])
           throws IOException
    {
     if (argv.length != 1) {
        System.err.println ("Usage: cmd filename");
        System.exit (1);
    }
    DocumentBuilderFactory factory =
                        DocumentBuilderFactory.newInstance();
    // recuperer une fabrique de constructeur de Dom
    try {
      DocumentBuilder builder = factory.newDocumentBuilder();
      // recuperer un constructeur de Dom
      Document document = builder.parse( new File(argv[0]) );
      // parser pour construire le Dom
      Element  rootElement =  document.getDocumentElement() ;
      // recuperer le premier element du Dom (cad root)
      printDomTree (rootElement, "");
    } catch (Throwable t) {
      t.printStackTrace ();
      System.exit (1);
    }
    System.exit (0);
  }
  public static void printDomTree (Node node, String indent)
    // ume methode pour afficher le sous-arbre Dom
    // à partir du noeud node en paramètre
    // avec une indentation conformement à l'arbre d'élément
    { String type;
      switch (node.getNodeType()) {
      // type d'élément XML
        case Node.ATTRIBUTE_NODE :
                        type = "attribut"; break;
        case Node.CDATA_SECTION_NODE :
                        type = "CDATA"; break;
        case Node.COMMENT_NODE :
                        type = "comment"; break;
        case Node.DOCUMENT_FRAGMENT_NODE :
                        type = "document fragment"; break;
        case Node.DOCUMENT_NODE :
                        type = "document"; break;
        case Node.DOCUMENT_TYPE_NODE :
                        type = "document type"; break;
        case Node.ELEMENT_NODE :
                        type = "node"; break;
        case Node.ENTITY_NODE :
                        type = "entity"; break;
        case Node.ENTITY_REFERENCE_NODE :
                        type = "entity reference"; break;
        case Node.NOTATION_NODE :
                        type = "notation"; break;
        case Node.PROCESSING_INSTRUCTION_NODE :
                        type = "processing instruction"; break;
        case Node.TEXT_NODE :
                        type = "text"; break;
        default : type = "none";
      }
      System.out.println(indent+"type : " + type);
      System.out.println(indent+"noeud name  : "
                              +node.getNodeName());
      // nom de l'élément ou de l'attribut ou ..
      System.out.println(indent+"value : "
                              +node.getNodeValue());
      // valeur de l'attribut ou texte d'un élément texte ou ..
      if (node.hasChildNodes())  {
      // comme les Tree-s JAVA .....
        Node nextFils = node.getFirstChild();
       while (nextFils != null) {
           printDomTree (nextFils,  indent+" ");
           nextFils = nextFils.getNextSibling();
      }
      }
    }
  }
```

### commandes et résultats

```
$ java Dom1 dialogue7.xml
type : node
noeud name  : dialogue
value : null
 type : node
 noeud name  : situation
```

```
 value : null
  type : text
  noeud name  : #text
  value : acte I, Scene 1 : madame pernelle et flipote sa servante,elmire, mariane,
dorine, damis, cléante.
 type : node
 noeud name  : replique
 value : null
  type : node
  noeud name  : personnage
  value : null
   type : text
   noeud name  : #text
   value : madame pernelle
  type : node
  noeud name  : texte
  value : null
   type : text
   noeud name  : #text
   value : Allons, Flipote, allons, que d'eux je me délivre.

....
```

### Et les attributs ...

### Programme

```
// Dom15.java
//
// afficher le Dom avec les attributs

import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.NamedNodeMap;
.................
// meme programme que Dom1.java

      System.out.println(indent+"value : "
                         +node.getNodeValue());
      if (node.hasAttributes())  {
        NamedNodeMap attList = node.getAttributes();
        // recupere la liste des attributs
        for (int i=0; i<attList.getLength(); ++i)
           printDomTree (attList.item(i),  indent+" ");
      }
      if (node.hasChildNodes())  {
        Node nextFils = node.getFirstChild();
       while (nextFils != null) {
           printDomTree (nextFils,  indent+" ");
           nextFils = nextFils.getNextSibling();
      }
      }
    }
  }
```

### commandes et résultats

```
$ java Dom15 dialogue9.xml
java Dom15 dialogue9.xml
type : node
noeud name  : dialogue
value : null
 type : node
 noeud name  : situation
 value : null
  type : text
  noeud name  : #text
  value : acte I, Scene 1 : madame pernelle et flipote sa servante,elmire, mariane,
dorine, damis, cléante.
 type : node
 noeud name  : replique
 value : null
  type : node
  noeud name  : personnage
  value : null
   type : attribut
   noeud name  : attitude
   value : pressée
   type : attribut
   noeud name  : geste
   value : marchant vite
   type : text
   noeud name  : #text
   value : madame pernelle
  type : node
  noeud name  : texte
  value : null
   type : attribut
   noeud name  : ton
   value : fort
   type : text
   noeud name  : #text
   value : Allons, Flipote, allons, que d'eux je me délivre.
```

## Exercice 3 :

1) En utilisant le parser DOM, lire un fichier XML contenant la liste de prix d'un café nommé "Cafe Noir".
2) Insérer dans le DOM un nouveau café, de nom"Cafe Noir" et qui coûte "3.10", juste avant le café qui s'appelle"Columbia".

Indication : d'abord récupérer la liste des éléments nom et itérer dans la liste pour trouver "Columbia", créer un nouvel objet Node pour le nouvel élément café ainsi que de nouveaux noeuds pour les éléments nom et prix (données caractères). Utiliser la méthode NoeudTrouve.insertBefore(nouveauNoeud, NoeudTrouve).

Correction :

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
Document document = builder.parse("liste_prix.xml");

```
NodeList list = document.getElementsByTagName("nom");
Node NoeudCourant = list.item(0);
```

bouclage dans la liste pour trouver le bon noeud (cf cours)

```
if (noeud.getData().equals("Columbia"))  //le nouveau noeud va être inséré avant Columbia
{
   Node newNode = document.createElement("cafe");
   Node nameNode = document.createElement("nom");
   TextNode textNode = document.createTextNode("Cafe Noir");
   nameNode.appendChild(textNode);

   Node priceNode = document.createElement("prix");
   TextNode textprixNode = document.createElement("3.10");
   priceNode.appendChild(textprixNode);

   newNode.appendChild(nameNode);
   newNode.appendChild(priceNode);

   NoeudCourant.insertBefore(newNode, NoeudCourant);
}
```

## TD3 - SAX

### Exercice

Considérons le fichier xml suivant donnant des informations sur la population d'une ville.

```
<?xml version="1.0" ?>
<population ville="Amiens" date="2005">

  <tranche age="0" nbFemmes="564" nbHommes="644"/>
   <tranche age="1" nbFemmes="320" nbHommes="430"/>
   <tranche age="2" nbFemmes="674" nbHommes="617"/>

   ...

   <tranche age="98" nbFemmes="272" nbHommes="372"/>
   <tranche age="99" nbFemmes="527" nbHommes="557"/>
   <tranche age="100" nbFemmes="103" nbHommes="150"/>
</population>
```

Utiliser SAX pour afficher le nombre de femmes et d'hommes ayant entre 18 et 25 ans dans la ville d'Amiens.

## TD3 - SAX - corrigé

### Exercice

Considérons le fichier xml suivant donnant des informations sur la population d'une ville.

```
<?xml version="1.0" ?>
<population ville="Amiens" date="2005">

<tranche age="0" nbFemmes="564" nbHommes="644"/>
<tranche age="1" nbFemmes="320" nbHommes="430"/>
<tranche age="2" nbFemmes="674" nbHommes="617"/>
...
<tranche age="98" nbFemmes="272" nbHommes="372"/>
<tranche age="99" nbFemmes="527" nbHommes="557"/>
<tranche age="100" nbFemmes="103" nbHommes="150"/>
</population>
```

Utiliser SAX pour afficher le nombre de femmes et d'hommes ayant entre 18 et 25 ans dans la ville d'Amiens.

Idée de correction :

Il faut redéfinir startElement pour traiter les valeurs de l'attribut age qui nous intéresse quand on arrive sur un element tranche.

Grosso-modo (voir cours pour le détail de la syntaxe) :

On prend deux variables globales compteur_femmes et compteur_hommes (initialisées globalement ou au début de l'élément population dont l'attribut ville a la valeur Amiens.

```
public void startElement(String name, AttributeList attrs) throws SAXException {

if (name.equals("tranche")
    if (attrs.getLength()==3)
        if ((attrs.getName(1).equals("age"))
```

```
        if ((attrs.getName(2).equals(nbFemmes)
            compteur_femmes+=Integer.parseInt(attrs.getValue(2));
        if ((attrs.getName(3).equals(nbHommes)
            compteur_hommes+=Integer.parseInt(attrs.getValue(3));

}
```

Et dans le endElement(...), on vérifie que c'est l'élément "population" d'attribut "Amiens" qu'on termine (drapeau à l'entrée par exemple) et on fait afficher les deux variables globales joliment.

```
System.out.println("Il y a + compteur_femmes + "femmes et " + nb_hommes + "hommes à " + attrs.getName(1));
```

# TD4

## Exercice 1 : écrire des chemins avec XPATH

Pour les arbres XPATH suivants, écrire les chemins permettant de localiser les noeuds demandés. Localiser ces noeuds dans l'arbre.

```
 <AAA>
        <BBB/>
        <CCC/>
        <BBB/>
        <BBB/>
        <DDD>
            <BBB/>
        </DDD>
        <CCC/>
</AAA>
```

1) l'élément racine AAA
2) tous les éléments CCC qui sont enfants de l'élément racine AAA
3) tous les éléments BBB qui sont enfants de DDD, qui sont enfants de l'élément racine AAA

```
<AAA>
        <BBB/>
        <CCC/>
        <BBB/>
        <DDD>
            <BBB/>
        </DDD>
        <CCC>
            <DDD>
                <BBB/>
                <BBB/>
            </DDD>
        </CCC>
    </AAA>
```

4) tous les éléments BBB
5) tous les éléments BBB qui sont enfants de DDD

```
<AAA>
        <XXX>
            <DDD>
                <BBB/>
                <BBB/>
                <EEE/>
                <FFF/>
            </DDD>
        </XXX>
        <CCC>
            <DDD>
                <BBB/>
                <BBB/>
                <EEE/>
                <FFF/>
            </DDD>
        </CCC>
        <CCC>
            <BBB>
                <BBB>
                    <BBB/>
                </BBB>
            </BBB>
        </CCC>
    </AAA>
```

6) tous les éléments inclus dans les éléments /AAA/CCC/DDD
7) tous les éléments BBB qui ont trois ancêtres
8) tous les éléments

```
<AAA>
        <BBB/>
        <BBB/>
        <BBB/>
        <BBB/>
    </AAA>
```

9) le premier élément BBB, fils de l'élément racine AAA

10) le dernier élément BBB, fils de l'élément racine AAA

```
<AAA>
        <BBB id = "b1"/>
        <BBB id = "b2"/>
        <BBB name = "bbb"/>
        <BBB/>
    </AAA>
```

11) tous les attributs id
12) tous les éléments BBB qui ont un attribut id
13) tous les éléments BBB qui ont un attribut name
14) tous les éléments BBB qui ont un attribut
15) tous les éléments BBB qui n'ont pas d'attribut
16) tous les éléments BBB ayant un attribut id dont la valeur est b1
17) tous les éléments BBB ayant un attribut name dont la valeur est bbb
18) tous les éléments BBB ayant un attribut name dont la valeur est bbb. Les espaces de début et de fin sont supprimés avant la comparaison

```
 <AAA>
        <CCC>
            <BBB/>
            <BBB/>
            <BBB/>
        </CCC>
        <DDD>
            <BBB/>
            <BBB/>
        </DDD>
        <EEE>
            <CCC/>
            <DDD/>
        </EEE>
    </AAA>
```

19) les éléments ayant deux enfants BBB
20) les éléments ayant deux enfants
21) les éléments ayant trois enfants

```
 <AAA>
        <BCC>
            <BBB/>
            <BBB/>
            <BBB/>
        </BCC>
        <DDB>
            <BBB/>
            <BBB/>
        </DDB>
        <BEC>
            <CCC/>
            <DBD/>
        </BEC>
    </AAA>
```

22) tous les éléments qui ont le nom BBB, équivalent à //BBB
23) tous les éléments dont le nom commence par la lettre B
24) tous les éléments dont le nom contient la lettre C

```
<AAA>
        <Q/>
        <SSSS/>
        <BB/>
        <CCC/>
        <DDDDDDDD/>
        <EEEE/>
    </AAA>
```

25) tous les éléments qui ont un nom dont le nombre de caractères est exactement trois
26 tous les éléments qui ont un nom de un ou deux caractères
27) tous les éléments qui ont un nom dont le nombre de caractères est strictement supérieur à trois

```
<AAA>
        <BBB/>
        <CCC/>
        <DDD>
            <CCC/>
        </DDD>
        <EEE/>
    </AAA>
```

28) tous les éléments CCC et BBB
29) tous les éléments EEE qui sont enfants de l'élément racine AAA et tous les éléments BBB.

## Exercice 2 : comprendre un chemin XPATH

Donner la signification des expressions XPATH suivantes et localiser les noeuds dans l'arbre correspondant.

1)

```
    <AAA>
        <BBB/>
        <CCC/>
    </AAA>
```

/AAA
/child::AAA
/AAA/BBB
/child::AAA/child::BBB
/child::AAA/BBB

2)

```
<AAA>
    <BBB>
        <DDD>
            <CCC>
                <DDD/>
                <EEE/>
            </CCC>
        </DDD>
    </BBB>
    <CCC>
        <DDD>
            <EEE>
                <DDD>
                    <FFF/>
                </DDD>
            </EEE>
        </DDD>
    </CCC>
</AAA>
```

/descendant::*
/AAA/BBB/descendant::*
//CCC/descendant::*
//CCC/descendant::DDD

3)

```
<AAA>
    <BBB>
        <DDD>
            <CCC>
                <DDD/>
                <EEE/>
            </CCC>
        </DDD>
    </BBB>
    <CCC>
        <DDD>
            <EEE>
                <DDD>
                    <FFF/>
                </DDD>
            </EEE>
        </DDD>
    </CCC>
</AAA>
```

//DDD/parent::*
/AAA/BBB/DDD/CCC/EEE/ancestor::*
//FFF/ancestor::*

4)

```
<AAA>
    <BBB>
        <CCC/>
        <DDD/>
    </BBB>
    <XXX>
        <DDD>
            <EEE/>
            <DDD/>
            <CCC/>
            <FFF/>
            <FFF>
                <GGG/>
            </FFF>
        </DDD>
    </XXX>
    <CCC>
        <DDD/>
    </CCC>
</AAA>
```

/AAA/BBB/following-sibling::*
//CCC/following-sibling::*
/AAA/XXX/preceding-sibling::*
//CCC/preceding-sibling::*

5)

```
<AAA>
    <BBB>
        <CCC/>
        <ZZZ>
            <DDD/>
            <DDD>
                <EEE/>
            </DDD>
        </ZZZ>
```

```
            <FFF>
                <GGG/>
            </FFF>
        </BBB>
        <XXX>
            <DDD>
                <EEE/>
                <DDD/>
                <CCC/>
                <FFF/>
                <FFF>
                    <GGG/>
                </FFF>
            </DDD>
        </XXX>
        <CCC>
            <DDD/>
        </CCC>
    </AAA>
```

```
/AAA/XXX/following::*
//ZZZ/following::*
/AAA/XXX/preceding::*
//GGG/preceding::*
```

6)

```
 <AAA>
        <BBB>
            <CCC/>
            <ZZZ>
                <DDD/>
            </ZZZ>
        </BBB>
        <XXX>
            <DDD>
                <EEE/>
                <DDD/>
                <CCC/>
                <FFF/>
                <FFF>
                    <GGG/>
                </FFF>
            </DDD>
        </XXX>
        <CCC>
            <DDD/>
        </CCC>
    </AAA>
```

```
/AAA/XXX/descendant-or-self::*
//CCC/descendant-or-self::*
/AAA/XXX/DDD/EEE/ancestor-or-self::*
//GGG/ancestor-or-self::*
```

7)

```
<AAA>
        <BBB>
            <CCC/>
            <ZZZ/>
        </BBB>
        <XXX>
            <DDD>
                <EEE/>
                <FFF>
                    <HHH/>
                    <GGG>
                        <JJJ>
                            <QQQ/>
                        </JJJ>
                        <JJJ/>
                    </GGG>
                    <HHH/>
                </FFF>
            </DDD>
        </XXX>
        <CCC>
            <DDD/>
        </CCC>
    </AAA>
```

```
//GGG/ancestor::*
//GGG/descendant::*
//GGG/following::*
//GGG/preceding::*
//GGG/self::*
//GGG/ancestor::* | //GGG/descendant::* | //GGG/following::* | //GGG/preceding::* |
//GGG/self::*
```

8)

```
 <AAA>
        <BBB/>
        <BBB/>
        <BBB/>
        <BBB/>
        <BBB/>
        <BBB/>
        <BBB/>
        <BBB/>
```

```
        <CCC/>
        <CCC/>
        <CCC/>
    </AAA>
```

```
//BBB[position() mod 2 = 0 ]
//BBB[ position() = floor(last() div 2 + 0.5) or position() = ceiling(last() div 2 + 0.5) ]
//CCC[ position() = floor(last() div 2 + 0.5) or position() = ceiling(last() div 2 + 0.5) ]
```

## TD5

**Exercice :**

Voici la dtd utilisée pour définir la structure d'une cdthèque et une feuille de style associée à cette cdthèque.

*cdtheque.dtd :*

```
<!ELEMENT cdtheque (cd*) >

<!ATTLIST cdtheque date CDATA #REQUIRED>
<!ATTLIST cdtheque auteur CDATA #REQUIRED>

<!ELEMENT cd  (artiste,titre,style?,piste*)>

<!ELEMENT artiste      (#PCDATA)>
<!ELEMENT titre        (#PCDATA)>
<!ELEMENT style        (#PCDATA)>
<!ELEMENT piste   (ptitre,pextras)>
<!ELEMENT ptitre (#PCDATA)>
<!ELEMENT pextras (#PCDATA)>

<!ATTLIST cd id CDATA #IMPLIED>
```

*cdtheque.xsl :*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="html" encoding="ISO-8859-1" />
<xsl:template match="/cdtheque">
  <html>
  <body>
  <xsl:apply-templates select="/cdtheque/cd"/>
  </body>
  </html>
  </xsl:template>
<xsl:template match="artiste" >
  <xsl:text>Artiste: </xsl:text><xsl:value-of select="text()" />
</xsl:template>
<xsl:template match="titre" >
  <xsl:text>Titre: </xsl:text><xsl:value-of select="text()" />
</xsl:template>
<xsl:template match="/cdtheque/cd">
  <xsl:text>==Un CD==</xsl:text><br/>
  <xsl:apply-templates select="artiste" /><br/>
  <xsl:apply-templates select="titre" /><br/>
  <xsl:text>--Fin du CD--</xsl:text><br/><br/>
</xsl:template>
</xsl:stylesheet>
```

1. A quoi ressemble la page HTML générée pour cette feuille de style ?

2. En vous inspirant de cette feuille de style, créez une feuille de style affichant la position du cd dans l'arbre au début de chaque ligne.

3. Créez une feuille de style affichant affichant la liste des cds triés par artiste

4. Créez une template permettant d'afficher les CDs de l'artiste passé en paramètre

## TP1

Les fichiers suivants représentent un dialogue extrait du Tartuffe de Molière.

**Version sans attribut :** dialogue1.xml

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<!-- fichier dialogue1.xml -->


<!-- extrait de Tartuffe de Molière -->
<dialogue>
<situation>acte I, Scene 1 : madame pernelle et flipote sa servante,elmire, mariane,
dorine, damis, cléante.</situation>
<replique>
<personnage>madame pernelle</personnage>
<texte>Allons, Flipote, allons, que d'eux je me délivre.</texte>
</replique>
<replique>
<personnage>elmire</personnage>
<texte>Vous marchez d'un tel pas qu'on a peine à vous suivre.</texte> </replique>
<replique>
<personnage>madame pernelle</personnage>
<texte>Laissez, ma bru, laissez, ne venez pas plus loin: Ce sont toutes façons dont
je n'ai pas besoin.</texte>
```

```
</replique>
<replique>
<personnage>elmire</personnage>
<texte>De ce que l'on vous doit envers vous on s'acquitte, Mais ma mère, d'où vient
que vous sortez si vite;</texte>
</replique>
</dialogue>
```

**Version avec attributs :** dialogue2.xml

```
<?xml version='1.0' encoding='ISO-8859-1' standalone='yes' ?>
<!-- fichier dialogue2.xml -->

<!-- extrait de Tartuffe de Molière -->
<dialogue>
<situation>acte I, Scene 1 : madame pernelle et flipote sa servante,elmire, mariane,
dorine, damis, cléante.</situation>
<replique>
<personnage attitude='pressée' geste='marchant vite' >madame pernelle</personnage>
<texte ton='fort' >Allons, Flipote, allons, que d'eux je me délivre.</texte>
</replique>
<replique>
<personnage attitude='essouflée' >elmire</personnage>
<texte>Vous marchez d'un tel pas qu'on a peine à vous suivre.</texte> </replique>
<replique>
<personnage attitude='agacée'>madame pernelle</personnage>
<texte>Laissez, ma bru, laissez, ne venez pas plus loin: Ce sont toutes façons dont
je n'ai pas besoin.</texte>
</replique>
<replique>
<personnage attitude='étonnée' >elmire</personnage>
<texte>De ce que l'on vous doit envers vous on s'acquitte, Mais ma mère, d'où vient
que vous sortez si vite;</texte>
</replique>
</dialogue>
```

*Remarque : vous pouvez éditez vos fichiers avec emacs.*

## Exercice 1 :

Modifier ce fichier xml afin de mettre les différents composants de la situation en attribut de dialogue.

*Remarque : en affichant vos fichiers xml avec Mozilla, vous verrez si votre fichier est bien formé ou non. S'il est bien formé, le navigateur affiche l'arborescence du fichier, sinon il indique où se trouve l'erreur de syntaxe.*

## Exercice 2 :

a) Ecrire la DTD interne correspondant au fichier XML précédent.
b) Modifiez-la en DTD externe.

## Exercice 3 :

Utilisez une entité pour récupérer le dialogue dans un fichier extérieur.

## Exercice 4 : Validation - Outils

Vérifiez que le document XML est valide, c'est-à-dire correspond bien à la DTD que vous avez écrite :

- Vous utiliserez pour cela la commande xmllint.

- Si xmllint n'est pas installé sur votre poste, vous pouvez utiliser un valideur en ligne (attention : fonctionne seulement avec DTD interne !!!) :

http://www.stg.brown.edu/service/xmlvalid/

## Exercice 5 :

Ecrire un document XML modélisant votre Curriculum Vitae (réfléchir à ce qui doit être mémorisé et comment structurer l'information).
Ecrire la DTD associée et vérifier que votre document XML est valide par rapport à cette DTD.

# XML TP1 - proposition de correction

<>

**<>Exercice 1 : attributs**

<>

<>
<?xml version='1.0' encoding='ISO-8859-1' ?>
<!-- fichier dialogue1.xml -->

<!-- extrait de Tartuffe de Moli\'ere -->
<dialogue acte='I' Scene='1' Personnages='madame pernelle, flipote sa servante, elmire, mariane, dorine, damis, cléante'>
<replique>
<personnage>madame pernelle</personnage>
<texte>Allons, Flipote, allons, que d'eux je me délivre.</texte>
</replique>
<replique>
<personnage>elmire</personnage>
<texte>Vous marchez d'un tel pas qu'on a peine à vous suivre.</texte> </replique>
<replique>
<personnage>madame pernelle</personnage>
<texte>Laissez, ma bru, laissez, ne venez pas plus loin: Ce sont toutes façons dont je n'ai pas besoin.</texte>
</replique>
<replique>
<personnage>elmire</personnage>
<texte>De ce que l'on vous doit envers vous on s'acquitte, Mais ma mère, d'où vient que vous sortez si vite;</texte>
</replique>
</dialogue>

## Exercice 2 : utiliser une dtd

1)

<?xml version='1.0' encoding='ISO-8859-1' standalone='yes' ?>
<!-- fichier dialogue4.xml -->

<!DOCTYPE dialogue [
<!ELEMENT dialogue (situation?, replique+) >
<!ELEMENT situation (#PCDATA) >
<!ELEMENT replique (personnage, texte) >
<!ELEMENT personnage (#PCDATA) >
<!ELEMENT texte (#PCDATA) >
]>

<!-- extrait de Tartuffe de Molière -->
<dialogue>
.....

2)

**Sans attributs :**

**le document xml**

<?xml version='1.0' encoding='ISO-8859-1' standalone='no' ?> <!--Attention ne pas oublier de changer standalone='no'-->
<!-- fichier dialogue3.xml -->

<!DOCTYPE dialogue SYSTEM "dia.dtd">

<!-- extrait de Tartuffe de Molière -->
<dialogue>
...

**et la dtd : dia.dtd**

<?xml version='1.0' encoding='ISO-8859-1' ?>
<!-- fichier dia.dtd -->
<!-- dtd dialogue -->

<!ELEMENT dialogue (situation?, replique+) >
<!ELEMENT situation (#PCDATA) >
<!ELEMENT replique (personnage, texte) >
<!ELEMENT personnage (#PCDATA) >
<!ELEMENT texte (#PCDATA) >

**Idem avec des attributs :**

**le document xml**

<?xml version='1.0' encoding='ISO-8859-1' standalone='no' ?>
<!-- fichier dialogue6.xml -->

<!DOCTYPE dialogue SYSTEM "dia.dtd">

<!-- extrait de Tartuffe de Molière -->
<dialogue>
...

**et la dtd : dia.dtd**

<?xml version='1.0' encoding='ISO-8859-1' ?>
<!-- fichier dia.dtd -->
<!-- dtd dialogue -->

<!ELEMENT dialogue (situation?, replique+) >
<!ELEMENT situation (#PCDATA) >

```
<!ELEMENT replique (personnage, texte) >
<!ELEMENT personnage (#PCDATA) >
<!ATTLIST personnage attitude  CDATA #REQUIRED
          geste CDATA #IMPLIED >
<!ELEMENT texte (#PCDATA) >
<!ATTLIST texte ton (normal | fort | faible) "normal">
```

### Exercice 3 : Document XML utilisant des entités

fichier dialogue5.xml

```
<?xml version='1.0' encoding='ISO-8859-1' standalone='no' ?>
<!-- fichier dialogue5.xml -->

<!DOCTYPE dialogue SYSTEM "dia2.dtd"[
 <!ENTITY prl "madame pernelle">
   entité interne
<!ENTITY elm "elmire">
 <!ENTITY dialogue_a SYSTEM "dialogue5a.xml">
   entité externe
 <!ENTITY dialogue_b SYSTEM "dialogue5b.xml">
]>

<!-- extrait de Tartuffe de Molière -->
<dialogue>
&dialogue_a;
   référence à une entité
&dialogue_b;
</dialogue>
```

fichier dialogue5a.xml

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<!-- fichier dialogue5a.xml -->

<situation>acte I, Scene 1 : &prl; et flipote sa servante,&elm;, mariane, dorine,
damis, cléante.</situation>
<replique>
<personnage>&prl;</personnage>
   référence à une entité
<texte>Allons, Flipote, allons, que d'eux je me délivre.</texte>
</replique>
<replique>
<personnage>&elm;</personnage>
<texte>Vous marchez d'un tel pas qu'on a peine à vous suivre.</texte>
</replique>
```

fichier dialogue5b.xml

....

### Exercice 5 : CV

cv.dtd

```
<!ELEMENT cv
(coordonnees,lesFormations,lesExperiences?,activites?,langues?,connaissancesInfo?) >

<!ELEMENT coordonnees (nom,prenom,adresse,dateDeNaissance,tel,mel) >
<!ELEMENT nom (#PCDATA) >
<!ELEMENT prenom (#PCDATA) >
<!ELEMENT dateDeNaissance (#PCDATA) >
<!ELEMENT tel (#PCDATA) >
<!ELEMENT mel (#PCDATA) >

<!ELEMENT adresse (numero,voie,codepostal,ville) >
<!ELEMENT numero (#PCDATA) >
<!ELEMENT voie (#PCDATA) >
<!ELEMENT codepostal (#PCDATA) >
<!ELEMENT ville (#PCDATA) >



<!ELEMENT lesFormations (formation*) >

<!ELEMENT formation (annee,diplome,lieu,mention?) >

<!ELEMENT annee (#PCDATA) >
<!ELEMENT diplôme (#PCDATA) >
<!ELEMENT lieu (#PCDATA) >
<!ELEMENT mention (#PCDATA) >



<!ELEMENT lesExperiences (experience*) >

<!ELEMENT experience (annee, libelle, lieu, descriptif?) >

<!ELEMENT libelle (#PCDATA) >
<!ELEMENT descriptif (#PCDATA) >


<!ELEMENT activites (#PCDATA) >
<!ELEMENT langues (langue*) >
<!ELEMENT langue EMPTY>
<!ATTLIST langue nom CDATA #REQUIRED>
<!ATTLIST langue parle (oui|non) #IMPLIED>
```

```
<!ATTLIST langue ecrit (oui|non) #IMPLIED>
```

cv.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE cv SYSTEM "cv.dtd">

<cv>

  <!-- Coordonnees -->
  <coordonnees>
    <nom>Dupont</nom>
    <prenom>Jean</prenom>

    <adresse>
      <numero> 82 </numero>
      <voie> Rue du vieux moulin </voie>
      <codepostal>80000</codepostal>
      <ville>Amiens</ville>
    </adresse>


    <dateDeNaissance>07 septembre 1980</dateDeNaissance>

    <tel>03.22.xx.xx.xx</tel>
    <mel>jean.dupont@u-picardie.fr</mel>

  </coordonnees>


  <!-- formations -->
  <lesFormations>
    <formation>
      <annee>1998</annee>
      <diplome>Baccalaureat S</diplome>
      <lieu>Amiens</lieu>
      <mention>Bien</mention>
    </formation>

    <formation>
      <annee>2000</annee>
      <diplome>Deug MIAS</diplome>
      <lieu>Amiens</lieu>
    </formation>
  </lesFormations>

</cv>
```