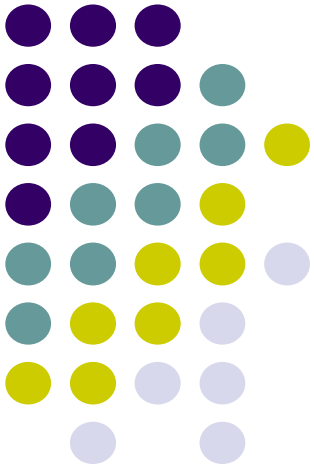
Grammaire d'un document XML

DTD (Document Type Definition)



Pr. Sidi Mohammed Benslimane
École Supérieure en Informatique
08 Mai 1945 – Sidi Bel Abbes –

s.benslimane@esi-sba.dz

Introduction: Préambule

La validation d'un document XML permet de vérifier sa structure et son contenu avant de commencer à le traiter.

La validation d'un document XML se fait à l'aide de deux techniques :

- Une définition de document type, appelée DTD (Document Type Definition), venant de la norme SGML assez simples.
- Un schéma XML (XML-Schemas) plus complet mais plus complexe.

Introduction: Terminologie

- Un document XML est dit "**bien formé**" lorsqu'il répond aux règles de base de XML et ne comporte pas de DTD.
- Un document XML est dit "**valide**" par rapport à une DTD s'il est bien formé et conforme à cette DTD.
- La DTD n'est pas obligatoire.
- La DTD n'a pas la syntaxe XML.

Introduction: Intérêt

- La DTD permet de spécifier une grammaire pour un langage et de tester automatiquement son respect par un document donné. L'avantage est de :
  - faciliter l'échange et la mise en commun de documents produits par différentes personnes;
  - aider les développeurs qui conçoivent des outils automatiques pour traiter les documents respectant la même DTD.

Introduction: Définition

- Le rôle d'une DTD est de définir précisément la structure d'un document à travers un certain nombre de contraintes que doit respecter un document pour être valide.
- Ces contraintes spécifient:
  - les éléments qui peuvent apparaître dans le contenu d'un élément,
  - l'ordre éventuel de ces éléments et leur occurrence.
  - les attributs autorisés et les attributs obligatoires pour chaque élément.
  - les types de contenus qui y sont permis.

Introduction: définition

- Une DTD peut être définie de trois manières :
  - **interne**: la grammaire est incorporée au sein même du document;
  - **externe**: la DTD est un fichier à part. Elle est appelée depuis le document XML;
  - **mixte**: il y a à la fois un fichier externe et des définitions locales.

Rappel

Dans le prologue du fichier XML, si :

- **standalone="yes"**, le document est auto-contenu ⇒ la DTD est interne.
- **standalone="no"**, le document n'est pas auto-contenu ⇒ la DTD est externe. Déclaration par défaut ;

DTD interne: Présentation

- Une DTD interne est une liste de règles définies au début d'un document XML pour permettre sa validation avant sa lecture.
- Elle est déclarée par un élément spécial DOCTYPE juste après le prologue.
- Les déclarations doivent être faites dans l'ordre :

```
<!DOCTYPE élément_racine
 [
        Regle_1
        Regle_2
        ...
        Regle_N
 ]>
 <élément_racine>
 ...
 </élément_racine>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE personne [
<!ELEMENT personne (nom, prenom, adresse)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT adresse (#PCDATA)>
]>
<personne>
  <nom>Mohamed</nom>
  <prenom>Ali</prenom>
  <adresse>rue des oiseaux, Alger</adresse>
<personne>
```

DTD Externe: Présentation

- Le stockage externe de la DTD permet de la partager entre différents documents XML,
- Les DTD externes peuvent être privées ou publiques.
- Les DTD privées sont : type **SYSTEM**
  - accessibles uniquement en local, sur la machine de développement ;
- Les DTD publiques sont : type **PUBLIC**
  - disponibles pour tout le monde, sur un serveur distant accècible via une URI (Uniform Resource Identifier) ;
- Le sous-ensemble externe est placé dans un fichier séparé.
- L'appel à ce fichier, qui doit avoir le même nom que la racine du document, est effectué dans le DOCTYPE après la déclaration XML.
- L'attribut *standalone* prendra dans la déclaration XML la valeur "no".

DTD Externe Privée: Syntaxe

Syntaxe:

```
<!DOCTYPE racine SYSTEM "URI">
```

Exemple:

```
<!DOCTYPE bibliotheque SYSTEM "c:\biblio.dtd">
```

DTD Externe Public: Syntaxe

Syntaxe:

<!DOCTYPE racine PUBLIC "*identifiant_public*" "*url*">

- identifiant_public contient les caractéristiques :
- type_enregistrement // proprietaire // DTD description // langue

- type_enregistrement :
  - un signe + qui indique que le document est enregistré auprès de l'ISO, et qu'il est conforme à la norme ISO-9070,
  - un signe - sinon ;
- Propriétaire : nom du propriétaire (entreprise ou personne) ;
- DTD description : une description textuelle, espaces autorisés ;
- langue : un code de langue ISO 639.

Exemple:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

DTD Mixte: presentation

- Il est possible de mélanger les deux notations pour avoir une partie de la DTD dans un fichier séparé et une autre partie embarquée dans le document XML.
- Lorsque la DTD est mixte, tout élément doit être déclaré soit dans la partie interne, soit dans la partie externe mais pas dans les deux.

Exemple:

```
<?xml version="1.0" ?>
<!DOCTYPE cv SYSTEM "curriculum.dtd"
[
Regle_1 Regle_2... Regle_N
]>
<cv>
…
</cv>
```

Déclarer les éléments: Présentation

Modèle de déclaration d'un élément dans une DTD:

<!ELEMENT nom_élément contenu_élément>

- nom_élément : nom de l'élément (nom d'une balise) dans le fichier XML associe ;
- Contenu_élément : type auquel il est associe. Les valeurs possibles sont :
  - Texte
  - Vide
  - Séquence d'éléments
  - choix d'éléments
  - mixte (mélange de texte et d'éléments enfants)
  - libre

Déclarer les éléments: Elément texte

Syntaxe:

```
<!ELEMENT nom_élément (#PCDATA)>
```

- **Cet élément doit contenir du texte.**

Recommandation

Pour éviter les éventuelles erreurs du parseur, mieux vaut mettre le mot clé **#PCDATA** entre parenthèses.

Exemple d'utilisation :

```
<!ELEMENT titre (#PCDATA)>
```

- Se traduira par exemple dans le document XML :

```
<titre> XML et les services Web </titre>
```

Déclarer les éléments: Elément vide

Syntaxe

```
<!ELEMENT nom_élément EMPTY>
```

- L'élément vide n'a aucun contenu : pas de texte, ni même d'autres éléments.
- L'élément vide peut uniquement avoir des attributs.
- C'est une balise auto-fermante **<nom_élément/>**.
- Les éléments vides sont souvent utilisés pour des liens entre éléments (voir **Identificateurs**).

Attention aux blancs entre les éléments !

- <Titre> </Titre>

Déclarer les éléments: Séquence d'éléments

Syntaxe

<!ELEMENT pere (fils$_1$, fils$_2$,…, fils$_n$)>

- Une séquence d'éléments est une liste ordonnée d'éléments (fils) qui apparaîtront comme des éléments enfants de l'élément principal (père).
- Cet élément principal ne pourra contenir aucun autre élément que ceux déclares dans la séquence.
- Les éléments enfants sont placés entre parenthèses et séparés par des virgules.
- Les éléments enfants doivent apparaître dans le fichier XML dans l'ordre de déclaration de la séquence.

Déclarer les éléments: Séquence d'éléments

Exemple d'utilisation :

- `<!ELEMENT auteur (nom, prenom)>`

Se traduira par exemple dans le document XML :

```
<auteur>
<nom> Mohamed </nom>
<prenom> Ali </prenom>
</auteur>
```

```
<auteur>
<nom> Mohamed </nom>
<prenom> Ali </prenom>
<age> 29 </age>
</auteur>
```

```
<auteur>
<prenom> Ali </prenom>
<nom> Mohamed </nom>
</auteur>
```

```
<auteur>
<prenom> Ali </prenom>
</auteur>
```

```
<auteur>
<nom> Mohamed </nom>
<prenom> Ali </prenom>
<prenom> omar </prenom>
</auteur>
```

Déclarer les éléments: contenu alternatif

Syntaxe

<!ELEMENT pere (fils$_1$ | fils$_2$ … | fils$_n$)>

- Un choix d'éléments permet de définir dans une liste les éléments enfants possibles.
- Les éléments enfants sont places entre parenthèses et séparés par des "|".

Attention

- Chaque élément enfant doit être déclaré dans la DTD.

Exemple

<!ELEMENT elem (elem1 | elem2 | elem3)>

L'élément elem doit contenir un seul des éléments elem1, elem2 ou elem3.

Exemple

<!ELEMENT elem (elem1, (elem2 | elem4), elem3)>

L'élément elem doit contenir un élément elem1, un élément elem2 ou un élément elem4 puis un élément elem3 dans cet ordre.

Déclarer les éléments: élément a contenu mixte

- Certains éléments ont un contenu mixte, c'est à dire qu'ils sont composés d'une alternance, dans un ordre quelconque, de fragments textuels et d'éléments.
- Le mot clé #PCDATA étant nécessairement en première position.
- Dans l'exemple suivant, l'élément **book** possède un contenu mixte. Il peut contenir du **texte** et des éléments **em** et **cite** en nombre quelconque et dans n'importe quel ordre.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE book [
<!ELEMENT book (#PCDATA | em | cite)*>
<!ELEMENT em (#PCDATA)>
<!ELEMENT cite (#PCDATA)>
]>
<book>
Du <em>texte</em>, une <cite>citation</cite> et encore du <em>texte</em>.
</book>
```

Déclarer les éléments: élément libre

Syntaxe

```
<!ELEMENT nom_élément ANY>
```

- Élément qui peut contenir tout élément déclaré dans la DTD et du texte.
- Dans ce cadre il n'y a aucune contrainte sur le contenu d'un élément.

Exemple

```
<!ELEMENT doc ANY>
<doc>
Ceci est une
<i>documentation</i>
très simple
</doc>
```

Indicateurs d'occurrence: Présentation

- Lors de la déclaration de séquence ou de choix d'éléments, à chaque élément enfant peut être attribuée une indication d'occurrence.
- Ces indicateurs permettent de définir les éléments XML qu'un élément peut ou doit contenir.

Indicateurs d'occurrence: Notation

Notation	Utilisation	Exemple
(a, b)	Séquence dans l'ordre	(nom, prenom)
(a|b)	l'un ou l'autre mais pas les deux	(masculin|feminin)
a?	élément optionnel [0,1]	(nom, prenom, adresse, telephone?)
a*	élément répétitif [0,N]	(client, produit*)
a+	élément répétitif [1,N]	(nom, prenom+)

Déclarer les attributs: présentation

Syntaxe

- <!ATTLIST nom_element nom_attribut type_attribut mode>

  - Nom_élément: nom de l'élément auquel cet attribut appartient;
  - Nom_attribut: nom de l'attribut en cours de définition;
  - type_attribut: type de l'attribut permis;
  - mode: la valeur par défaut de l'attribut;

Déclarer les attributs: présentation

Syntaxe

- <!ATTLIST nom_element nom_attribut type_attribut mode>

Type_attribut : type de donnée de l'attribut :

1. chaînes de caractères

- **CDATA:** Les attributs de type CDATA contiennent des chaînes de caractères.

Exemple

```
<!ELEMENT nom (#PCDATA)>
<!ATTLIST nom genre CDATA >
```

Document XML valide :

```
<nom genre="masculin"> Mohamed </nom>
```

Déclarer les attributs: présentation

2. Énumérations

L'attribut prend une valeur dans la liste des valeurs possibles définies dans la déclaration d'attribut.

Syntaxe

```
<!ATTLIST nom_element nom_attribut (val1 | val2 |..| valN) "valdefaut">
```

Exemple

- `<!ATTLIST film type (policier | fiction | aventure) "policier">`

Déclarer les attributs: présentation

3. Identificateurs

Les Identificateurs, s'utilisent pour matérialiser les liens entre les données d'un document XML,

Syntaxe

<!ATTLIST *nom_element nom_attribut* **ID>**

- Un attribut de type ID permet d'identifier de façon unique un élément du document.

<!ATTLIST *nom_element nom_attribut* **IDREF>**

<!ATTLIST *nom_element nom_attribut* **IDREFS>**

- Les types IDREF et IDREFS sont des références ou des liste de références séparées par des espaces, à un identifiant unique précédemment déclaré.

Déclarer les attributs: présentation

Exemple Identificateurs

```
<!ELEMENT e1 (e2 | e3 | e4)*>
<!ELEMENT e2 (#PCDATA)>
<!ELEMENT e3 (#PCDATA)>
<!ELEMENT e4 (#PCDATA)>
<!ATTLIST e2 code ID #REQUIRED>
<!ATTLIST e3 ref IDREF #IMPLIED>
<!ATTLIST e4 refs IDREFS #IMPLIED>
```

```
<e1>
    <e2 code="id1">...</e2>
    <e2 code="id2">...</e2>
    <e3 ref="id1">...</e3>
    <e4 refs="id1 id2">...</e4>
</e1>
```

Déclarer les attributs: présentation

Exemple Identificateurs

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE DOCUMENT [
<!ELEMENT DOCUMENT(PERSONNE*)>
<!ELEMENT PERSONNE (#PCDATA)>
<!ATTLIST PERSONNE PNUM ID #REQUIRED >
<!ATTLIST PERSONNE MERE IDREF #IMPLIED>
<!ATTLIST PERSONNE PERE IDREF #IMPLIED>
                    ]>
<DOCUMENT>
<PERSONNE PNUM = "P1">Zohra</PERSONNE>
<PERSONNE PNUM = "P2">Mohamed</PERSONNE>
<PERSONNE PNUM = "P3" MERE="P1" PERE="P2">Ali</PERSONNE>
<PERSONNE PNUM = "P4" MERE="P1" PERE="P2">Fatima</PERSONNE>
</DOCUMENT>
```

Déclarer les attributs: présentation

Syntaxe

- `<!ATTLIST nom_element nom_attribut type_attribut mode>`

Mode: peut prendre l'une des quatre formes suivantes:

- **#REQUIRED** = obligatoire, i.e. la valeur d'attribut doit être spécifiée lorsque l'élément est utilisé dans le document.
- **#IMPLIED** = facultatif, i.e. la valeur d'attribut peut rester non spécifiée
- **#FIXED 'val'** = fixée à 'val' i.e. la valeur de l'attribut est fixe et ne peut pas être modifiée par l'utilisateur
- **'valeur'** = l'attribut a cette valeur par défaut, mais celle-ci peut être modifiée

Déclarer les attributs: présentation

Exemple mode

```
<!ELEMENT coordonnees>
<!ELEMENT coordonnees (#PCDATA)>
<!ATTLIST coordonnees telephone CDATA #REQUIRED>
<!ATTLIST coordonnees email CDATA #IMPLIED>
<!ATTLIST coordonnees pays CDATA #FIXED "Algérie" >
```

Oubien

```
<!ELEMENT coordonnees>
<!ELEMENT coordonnees (#PCDATA)>
<!ATTLIST coordonnees telephone CDATA #REQUIRED
                      email CDATA #IMPLIED
                      pays CDATA #FIXED "Algérie">
```

```
<coordonnees telephone="048-54-54-54" email="nom@domaine.dz" pays="Algérie">
   Benamar Ali
</coordonnees>
```

Les entités : présentation

- Le principe sur lequel est fondé le système des entités est celui du remplacement d'une chaîne de caractères par un symbole, puis de l'utilisation du symbole à la place de cette chaîne.
- On distingue deux types d'entités: paramètres et générales.
- La différence réside dans le contexte d'utilisation :
  1. Les entités **générales** sont définies dans la DTD et utilisées dans les documents correspondants.
  2. Les entités **paramètres** sont définies dans la DTD et utilisées dans la DTD elle-même.

Les entités générales : présentation

- Une entité générale interne est définie à l'intérieur d'une DTD.
- Elle correspond à une version abrégée d'un texte long lorsqu'il doit être évoqué à l'identique un grand nombre de fois.
- Une entité générale est toujours invoquée sous la forme **&symbole;** au sein d'un document là où devrait apparaître le texte de remplacement associé.
- Les entités générales peuvent être de deux types:
  - **internes**: définies dans l'entité document elle-même,
  - **externes**: dépendent d'une source externe au document XML.

Les entités générales: Définition d'une entité interne

Synthaxe

<!ENTITY nom_entité "texte associé">

La référence à une entité se fait en préfixant son nom avec &
et le postfixant avec ;

Exemple

```
<!DOCTYPE exemple[
<!ELEMENT exemple (#PCDATA)>
<!ENTITY DTD "Définition d'un Type de Documents (DTD)">
]>
<exemple> Une &DTD; est une grammaire pour valider un type de
documents XML
</exemple>
```

Les entités générales: Définition d'une entité externe

Si le code associé à une entité devient très important, il peut être intéressant de le détacher dans un fichier à part.

Synthaxe

<!ENTITY nom_entité SYSTEM "fichier associé">

Dans ce cas, le fichier associé doit être un fichier XML bien formé,

Exemple

Dans la DTD :

```
<!ENTITY adr SYSTEM "adresse.txt">
```

Dans le document XML:

&adr; doit être remplacé par le texte contenu dans le fichier adresse.txt

Les entités générales: Définition d'une entité externe

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE livre [
<!ELEMENT livre (titre, chapitre*)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT chapitre (titre, section+)>
<!ATTLIST chapitre numero CDATA #REQUIRED>
<!ELEMENT section (#PCDATA)>
<!ENTITY ch01 SYSTEM "chapitre_01.xml">
<!ENTITY ch02 SYSTEM "chapitre_02.xml">
<!ENTITY ch03 SYSTEM "chapitre_03.xml">
<!ENTITY ch04 SYSTEM "chapitre_04.xml">
]>

<!-- début du document -->
<livre>
    <titre>XML pour vous</titre>
    &ch01;
    &ch02;
    &ch03;
    &ch04;
</livre>
```

Les quatre fichiers chapitre xx.xml sont construits à l'image de celui-ci.

```
<chapitre numero="1">
    <titre>premier chapitre</titre>
    <section>première</section>
    <section>deuxième</section>
    <section>troisième</section>
</chapitre>
```

```
<!-- début du document -->
<livre>
    <titre>XML pour vous</titre>
      <chapitre numero="1">
          <titre>premier chapitre</titre>
          <section>première</section>
          <section>deuxième</section>
          <section>troisième</section>
      </chapitre>
      ...

<livre>
```

Les entités paramètres: Définition

- Comme pour une entité générale, la définition d'une entité paramètre revient à :
  1. définir le nom de l'entité, qui doit être un nom XML et sert de symbole utilisé, plus loin, dans la DTD,
  2. définir le texte de remplacement qui sera invoqué par l'intermédiaire de **%nom_entité**;

Synthaxe

<!ENTITY % *nom_entité* "*texte de remplacement*">

- Cependant, comme elle est utilisée dans la DTD même où elle est définie, Une entité paramètre doit obligatoirement être définie avant d'être utilisée.

- Le symbole **%** est utilisé dans la définition d'une entité paramètre pour la distinguer de la définition d'une entité générale.

Les entités paramètres: Exemples

Déclaration d'élément:

```
<!ENTITY % affiliation "(nom, prenom, date_naissance)">
<!ELEMENT etudiant (num_inscription, %affiliation;, groupe, section)>
<!ELEMENT enseignant(matricule, %affiliation;, grade, specialité)>
```

Déclaration d'attributs:

```
<!ENTITY % listeMarques "marque (Samsung|Apple|Condor) #REQUIRED">
<!ATTLIST telephone %listeMarques; >
```

Exemple de DTD et Document XML associé

Exemple

```
<!ELEMENT personne (nom, prenom+, tel?, email, adresse >
<!ELEMENT nom (#PCDATA) >
<!ELEMENT prenom (#PCDATA) >
<!ELEMENT tel (#PCDATA) >
<!ELEMENT email (#PCDATA) >
<!ELEMENT adresse ANY >
```

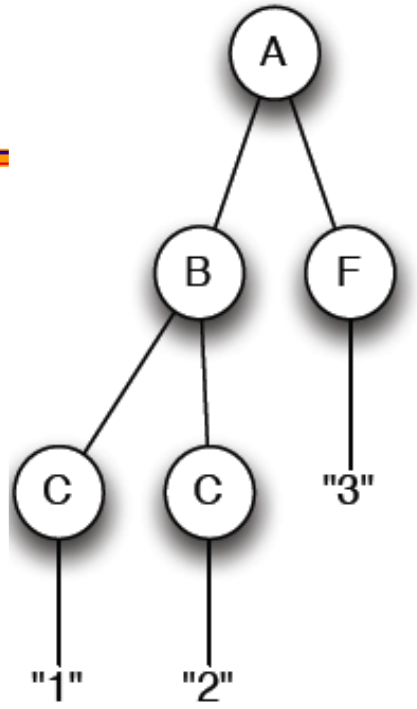
Document XML associé

```
<personne>
<nom> Bennani </nom>
<prenom> Mohammed </prenom>
<prenom> Ali </prenom>
<tel> 0683000000 </tel>
<email> bennani@domaine.dz </email>
<adresse> 25 rue Pasteur <ville> Sidi Bel Abbes </ville></adresse>
</personne>
```

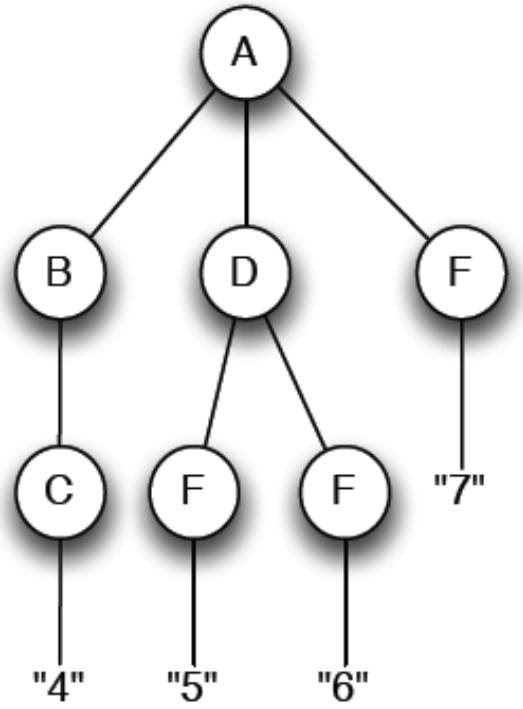
Limites des DTD

- La DTD n'est pas écrite dans une syntaxe XML.
- Le nombre d'apparitions d'un élément ne peut pas être contraint précisément. On ne dispose que des quantifieurs ?, * et +.On ne peut pas dire qu'un élément doit apparaître plus de 3 fois mais toujours moins de 7.
- On ne peut pas contraindre la forme de ces contenus (par exemple, entre 5 et 20 caractères, contenant un signe @, ...).
- Pas de concepts de classe et d'héritage.
- Pas de typage: les éléments terminaux contiennent que des chaînes de caractères.

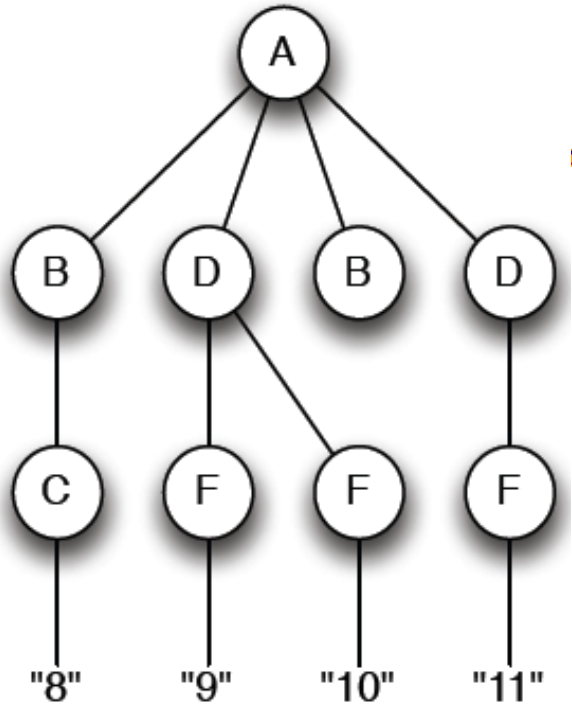
Exercices d’application



a_1



a_2

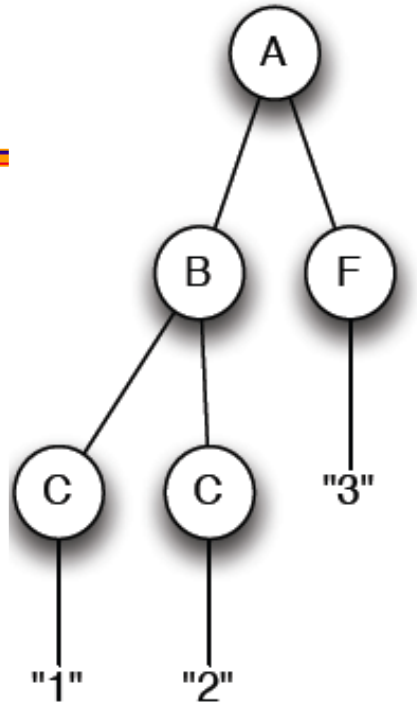


a_3

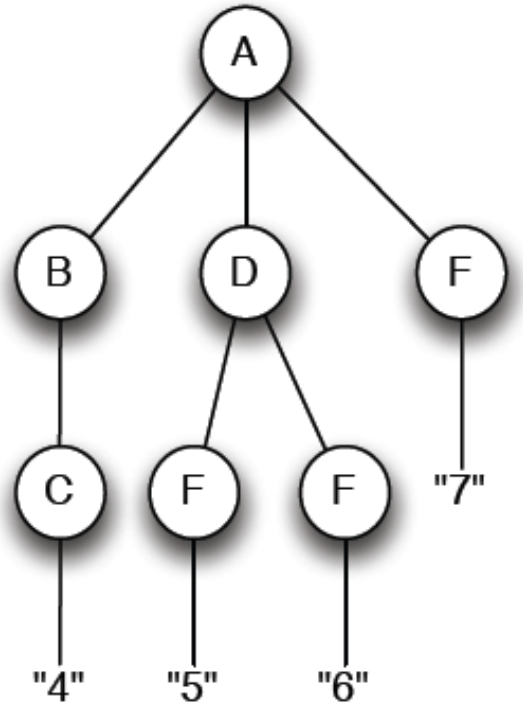
```
<!ELEMENT A ((B|D)+,F)>
<!ELEMENT B (C*)>
<!ELEMENT C (#PCDATA)>
<!ELEMENT D (F*)>
<!ELEMENT F (#PCDATA)>
```

d1.dtd

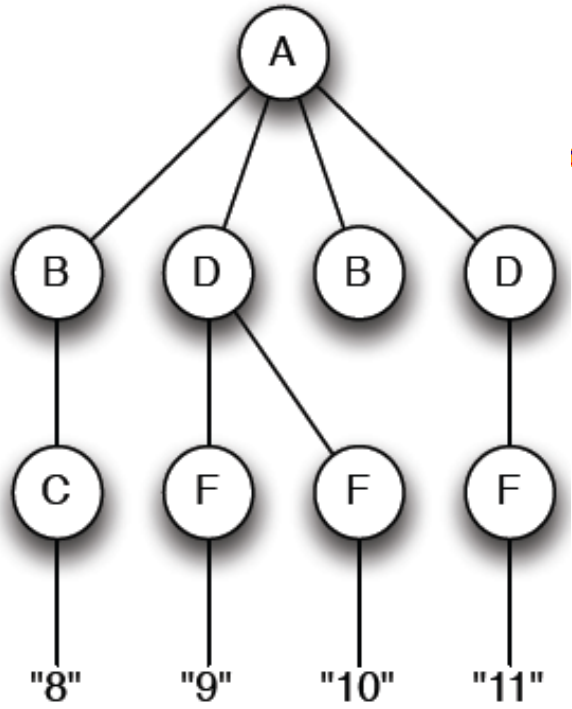
Solutions
a1 a2



a_1



a_2

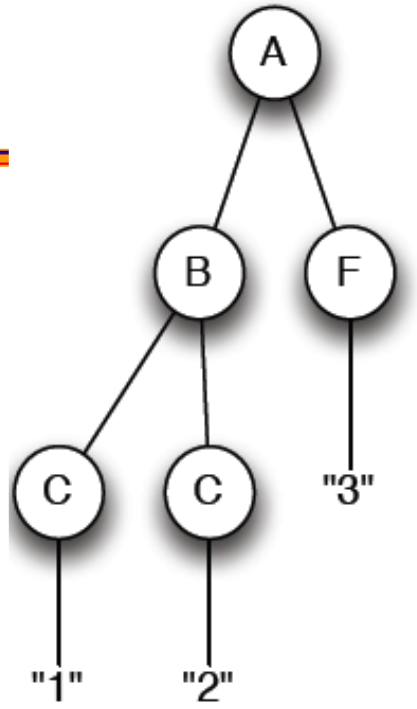


a_3

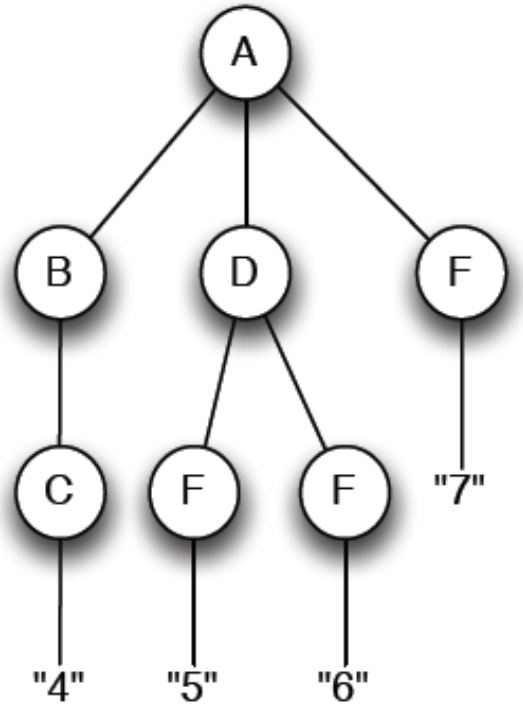
```
<!ELEMENT A ((B,D)+,F?)>
<!ELEMENT B (C*)>
<!ELEMENT C (#PCDATA)>
<!ELEMENT D (F*)>
<!ELEMENT F (#PCDATA)>
```

d2.dtd

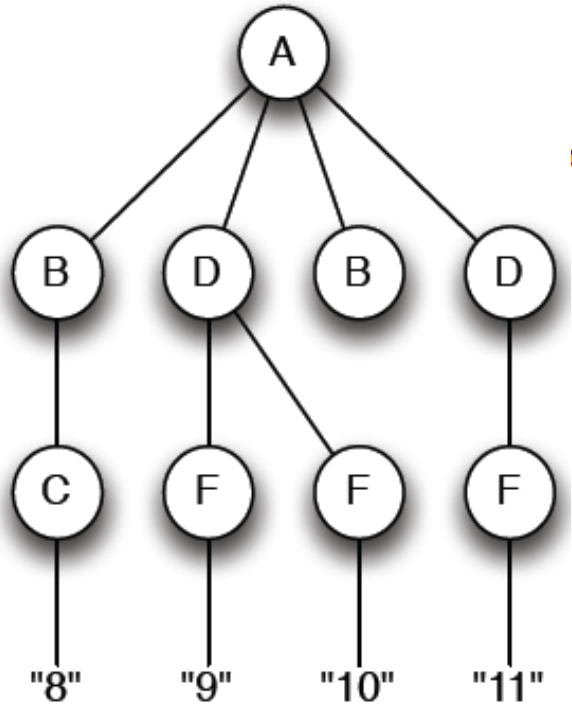
Solutions
a2 a3



a_1



a_2



a_3

```
<!ELEMENT A (B+,D+,F*)>
<!ELEMENT B (C*)>
<!ELEMENT C (#PCDATA)>
<!ELEMENT D (F*)>
<!ELEMENT F (#PCDATA)>
```

d3.dtd

Solution
a2

Questions ?