Syntaxe XML

Forme sérialisée et forme arborescente

Il existe deux représentations d'un document XML.

- **Forme sérialisée** : c'est la forme courante, où le contenu est marqué par des balises.
- **Forme arborescente** : elle met en évidence la structure du document.

Il est plus facile de raisonner sur la forme arborescente pour concevoir des traitements

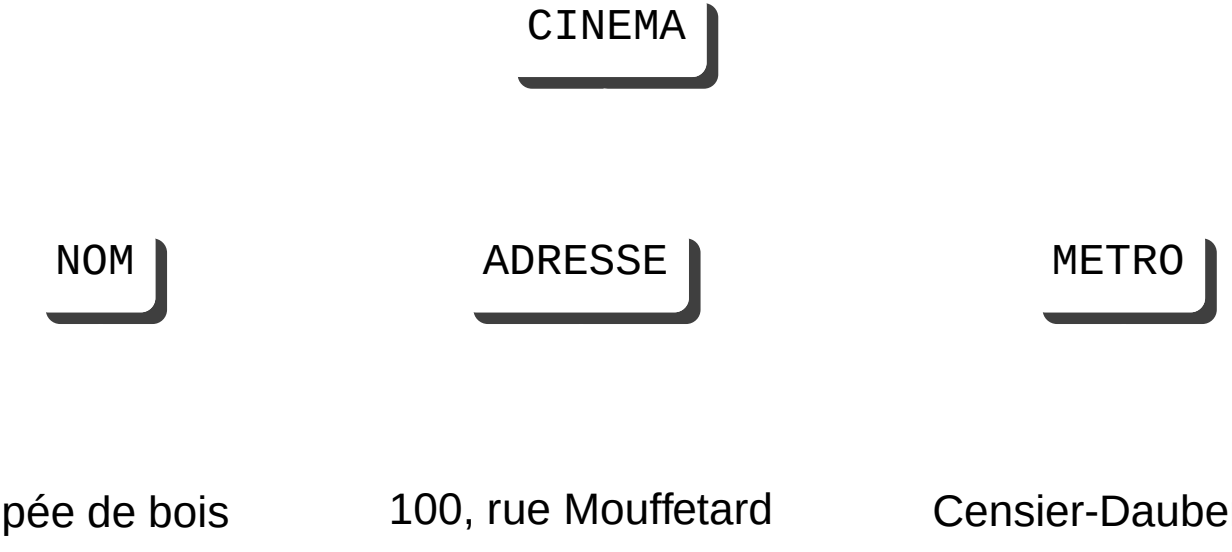
Un document sous forme sérialisée

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<CINEMA>
  <NOM>
    Epée de Bois
  </NOM>
  <ADRESSE>
     100, rue Mouffetard
   </ADRESSE>
  <METRO>
     Censier-Daubenton
  </METRO>
</CINEMA>
```

Cette représentation permet le stockage et l'échange de documents.

Le même, sous forme arborescente

Un arbre, constitué de nœuds typés (éléments, commentaires, etc)



La structure des arbres XML est définie par le *Document Object Model* (DOM)

En pratique

On utilise la forme sérialisée :

- pour stocker un document dans un fichier
- pour échanger des documents

La forme arborescente :

- permet de spécifier des manipulations de données XML ;
- utilisée par certaines applications qui gèrent l'ensemble du document en mémoire (exemple : éditeurs XML)

Exemple : un document sérialisé

Elle permet de marquer, par des balises, la structure d'un document.

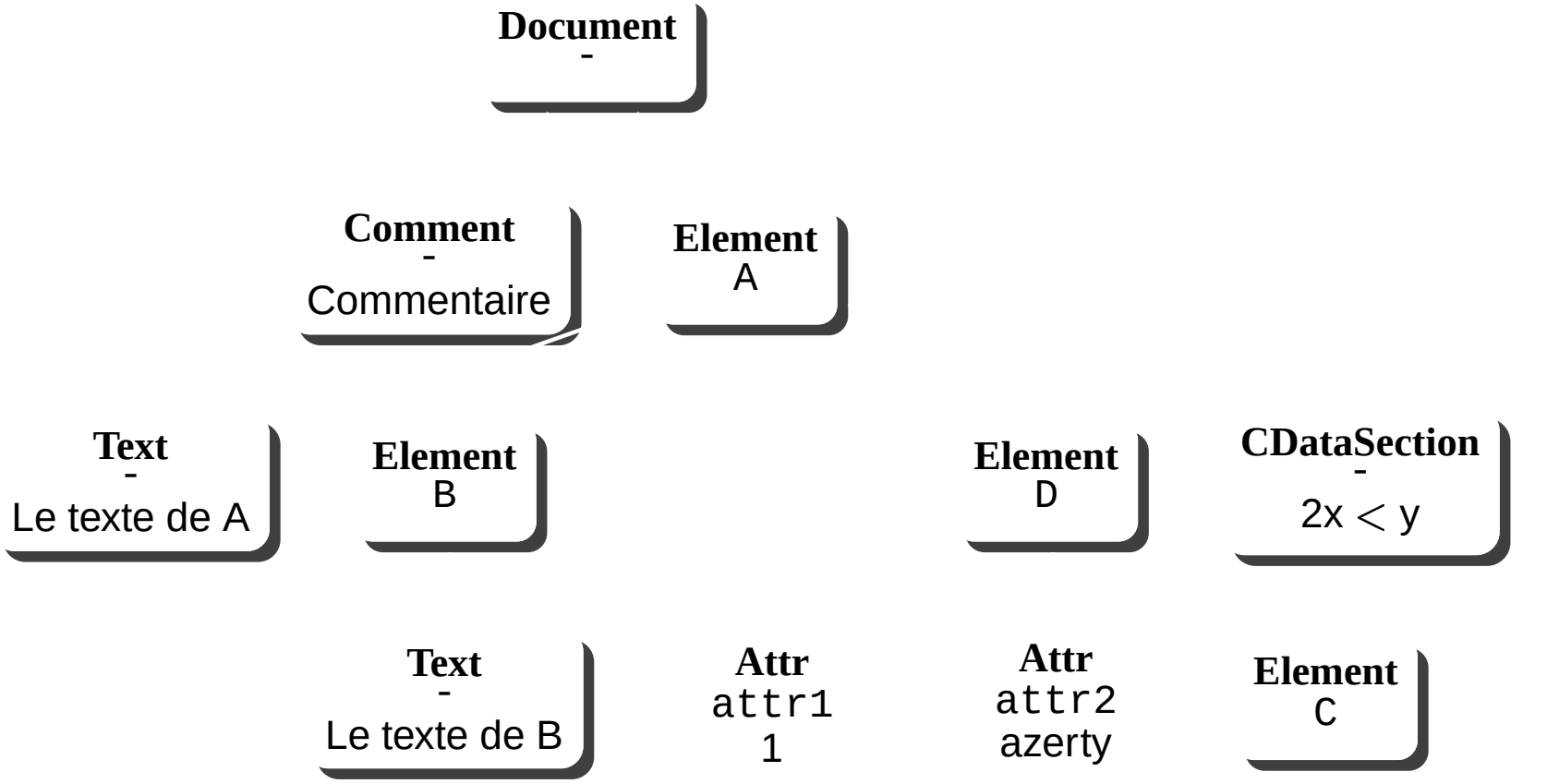
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Commentaire -->
<A>Le texte de A
  <B>Le texte de B</B>
  <D attr1="1" attr2="azerty">
   <C/>
  </D>
  <![CDATA[2x < y]]>
</A>
```

Passage à la représentation DOM

Le document sérialisé est analysé, et une représentation arborescente est créée :

- le nœud racine est de type **Document**
- les catégories syntaxiques (commentaires, balises, texte) se traduisent par différents types de nœuds (**Comment**, **Element**, **Text**)
- les nœuds constituent un arbre qui reflète l'imbrication des éléments dans la forme sérialisée

La représentation DOM



Nom, valeur et contenu d'un nœud

Notions essentielles à retenir :

- certains nœuds ont un **nom** (éléments), d'autres pas (texte, commentaires)
- certains nœuds ont une **valeur** (attributs, texte), d'autres pas (éléments).
- certains nœuds ont un **contenu** (éléments), d'autres pas (texte, attributs).

Un **contenu** est un sous-arbre, une **valeur** est une chaîne de caractères sans balisage.

Présentation de la syntaxe XML

Un ensemble de catégories syntaxiques :

- les éléments (et leurs attributs)
- les commentaires
- les instructions de traitement
- les sections de texte
- les sections littérales

Ainsi que quelques règles sur la structure d'un document

La déclaration XML

Tout document XML doit être précédé par une déclaration :

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

- l'attribut `encoding` indique le jeu de caractères utilisé dans le document
- l'attribut optionnel `standalone` indique si le document fait référence à d'autre documents

Déclaration de type

On peut indiquer qu'un document est conforme à une *DTD*, et déclarer des *entités*.

`<!DOCTYPE` *`nomDocument`* `SYSTEM "`*`sourceExt`*`" [`*`decLoc`*`]>`

- *`nomDocument`* est le nom du « type »
- *`sourceExt`* est la source extérieure contenant la DTD
- *`decLoc`* sont des déclarations locales (pour les entités principalement)

Entités et références à des entités

Les *entités* servent à factoriser des parties du document.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE A SYSTEM "minimal.dtd" [
  <!ENTITY  monTexte "texte simple">
  <!ENTITY  maSignature SYSTEM "signature.xm
]>
<A>
  Du &monTexte;, sans caractères réservés:
  ni &lt; ni &gt; ni &amp; ni &apos; ni &quo
  &maSignature;
</A>
```

Entités prédéfinies

Déclaration entité	Référence	Car.
`<!ENTITY lt "&#60;">`	`&lt;`	<
`<!ENTITY gt "&#62;">`	`&gt;`	>
`<!ENTITY amp "&#38;">`	`&amp;`	&
`<!ENTITY apos "&#39;">`	`&apos;`	'
`<!ENTITY quot "&#34;">`	`&quot`	"

TAB. 1 –

Commentaires et instructions de traitement

- Les commentaires : à utiliser avec parcimonie
  Exemple :
  ```
  <!-- Ceci est un commentaire -->
  ```
- Instructions de traitement : lié au système qui traite le document.
  Exemple :
  ```
  <?xml-stylesheet href="prog.xslt"
  type="text/xslt">
  ```

Éléments

Dans la forme sérialisée :

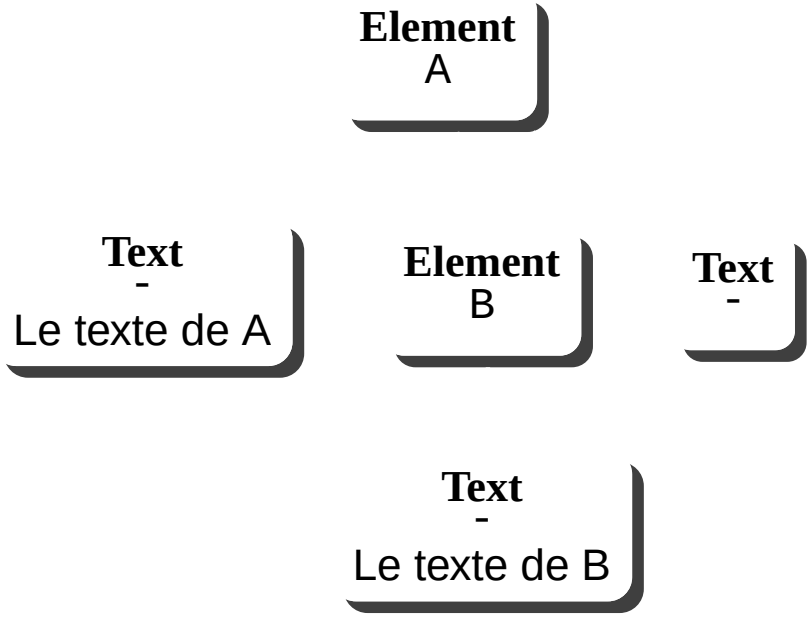
- C’est une **balise ouvrante** avec un nom, puis un **contenu**, puis une **balise fermante**

Dans la forme arborescente

- C’est un nœud, avec un nom ;
- Le contenu est un sous-arbre de ce nœud.

Exemple : un élément avec contenu

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<A>Le texte de A
  <B>Le texte de B</B>
</A>
```



Quelques remarques sur les éléments

- Un nom d'élément ne contient pas de blanc, ni de caractère accentué
- Les majuscules sont distinguées des minuscules
- Il existe une forme abrégée pour les éléments sans contenu : `<C></C>` peut s'écrire `<C/>`.
- **Tout document comprend un et un seul élément racine**

Attributs

Les attributs constituent un **autre** moyen de représenter de l'information.

```
<A att1='1' att2='2'>
```

- l'ordre des attributs n'est pas important
- il doit toujours y avoir une valeur, encadrée par des apostrophes simples ou doubles (différent de HTML)
- il ne peut pas y avoir deux attributs avec le même nom dans un élément.

Attributs : exemples

- `<A att1=’1’ att2=’2’>` est équivalent à `<A att2=’2’ att1=’1’>`
- `<A att=a>` n’est pas bien formé : pas d’apostrophe
- `<A att1=’1’ att1=’2’>` : interdit
- `<A att1=’1’> <B att1=’2’>` : autorisé (deux éléments différents)

Les sections littérales

À priori, on n’a pas le droit de placer dans le contenu d’un document XML des caractères comme ’<’, ’>’, ou ’&’.
Exemple incorrect :

```
<?xml version=’1.0’?>
<PROGRAMME>
if ((i < 5) && (j > 6))
   printf("error");
</PROGRAMME>
```

Solution : sections CDATA

Elles permettent d'inclure du texte qui n'est pas analysé par le parseur :

```
<?xml version='1.0'?>
<PROGRAMME>
<![CDATA[if ((i < 5) && (j > 6)) printf("err
</PROGRAMME>
```

Résumé : structure d'un document XML

Un document XML comprend trois parties :

- le **prologue**, avec la déclaration XML, la DTD, des commentaires, des instructions de traitements (optionnels)
- un **élément racine** avec son contenu
- un **épilogue** avec des commentaires, ou des instructions de traitements (optionnels)

Le contenu du document proprement dit est le contenu de l'élément racine.