UE Ingénierie documentaire
Master 2ème Année

Fabrice Lefèvre

2021

AVIGNON UNIVERSITÉ

XML

Partie 4 - XPath

Fabrice Lefèvre
fabrice.lefevre@univ-avignon.fr
2021

Référencer des fragments XML

Objectif : permettre le **référencement** et l'**extraction de fragments** (groupe de noeuds) XML d'un document

→ **XPath**

XPath est utilisé par

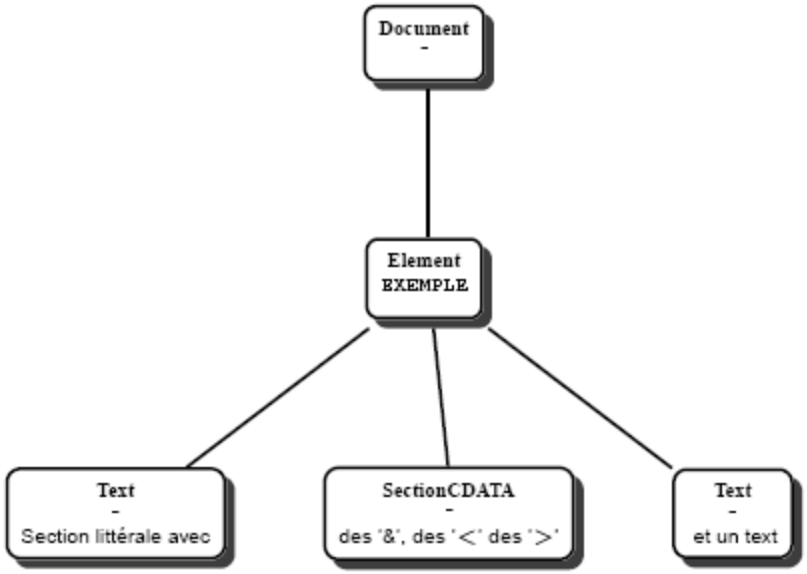
- XML Schéma pour créer des clés et références
- XSLT pour sélectionner des règles de transformation
- XQuery pour l'interrogation de documents XML
- XLink pour créer des liens hypertextes entre documents/fragments XML

Le modèle XPath

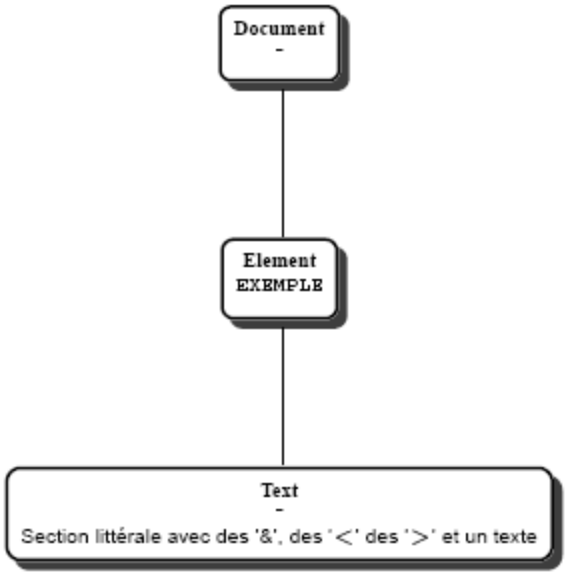
Le langage permet de désigner un ou plusieurs noeuds dans un document XML, à l'aide d'**expressions** de chemin

- XPath est fondé sur la représentation arborescente (DOM) du document XML
- Un typage simplifié par rapport à celui de DOM
  - `pas d'entité`
  - `pas de section littérale`

Exemple, arbre DOM



...puis l'arbre XPath



Expressions

Une **expression** XPath :

- s'évalue en fonction d'un noeud contexte
- désigne un ou plusieurs chemins dans l'arbre à partir du noeud contexte
- a pour résultat
  - `un ensemble de noeuds (éventuellement un singleton)`
  - **`ou`** `une valeur, numérique, booléenne ou alphanumérique`

Syntaxe

Un chemin XPath est une **suite d'étapes** :

$$[/]\text{etape}1/\text{etape}2/\ldots/\text{etape}n$$

Deux variantes : un chemin peut être

- **absolu** : le noeud contexte est alors la racine du document.

  /FILM/RESUME

- **relatif** : le noeud contexte est un noeud quelconque du document (pas forcément la racine)
  - par exemple dans XSLT, il est fixé par l'algorithme de parcours des arbres et les appels de règles

  RESUME/text()

Étapes

Une étape contient trois **composants** :

axe::filtre[prédicat1] [prédicat2]... [prédicat*n*]

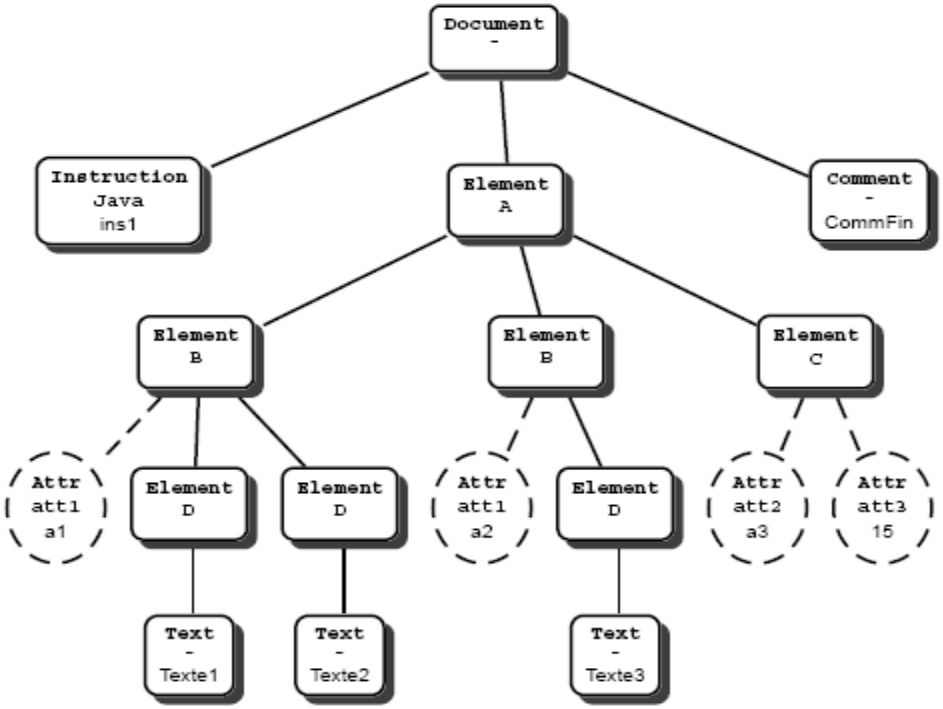
- *l'axe* (optionnel) : donne le sens de parcours des noeuds (par défaut, *child*).
- *le filtre* : donne le type des noeuds ou le nom des éléments qui seront retenus
- *le(s) prédicat(s)* (= *conditions*) que doivent satisfaire les noeuds retenus (optionnel)

Les axes XPath

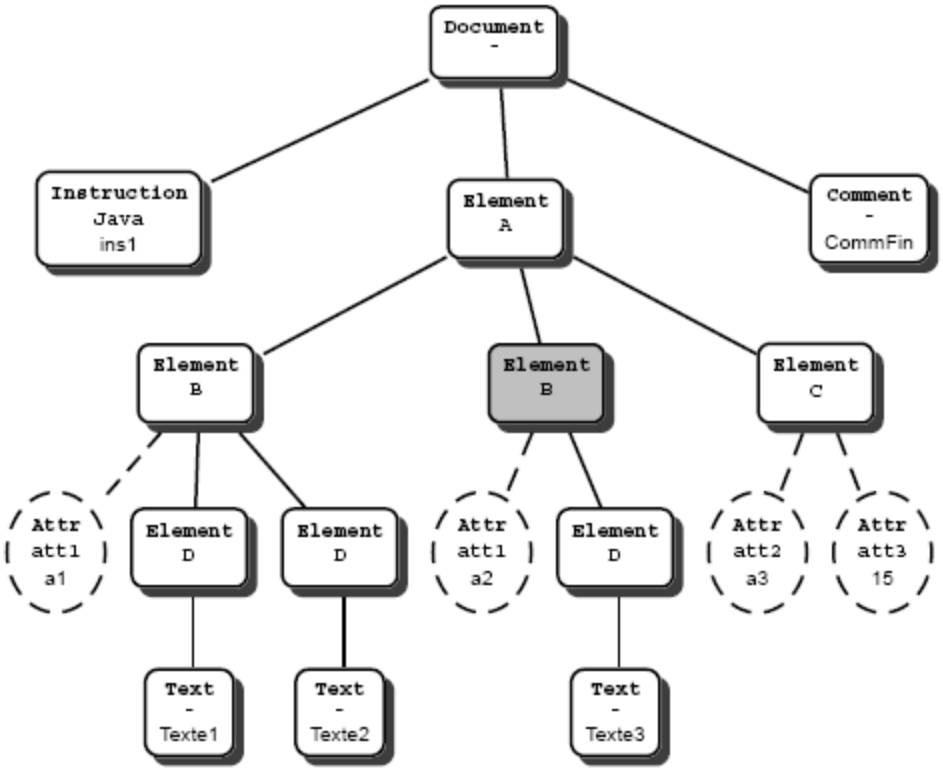
Un axe XPath recouvre les deux notions suivantes :

- un **sous-ensemble** des noeuds de l'arbre relatif au noeud contexte
- l'**ordre de parcours** de ces noeuds à partir du noeud contexte

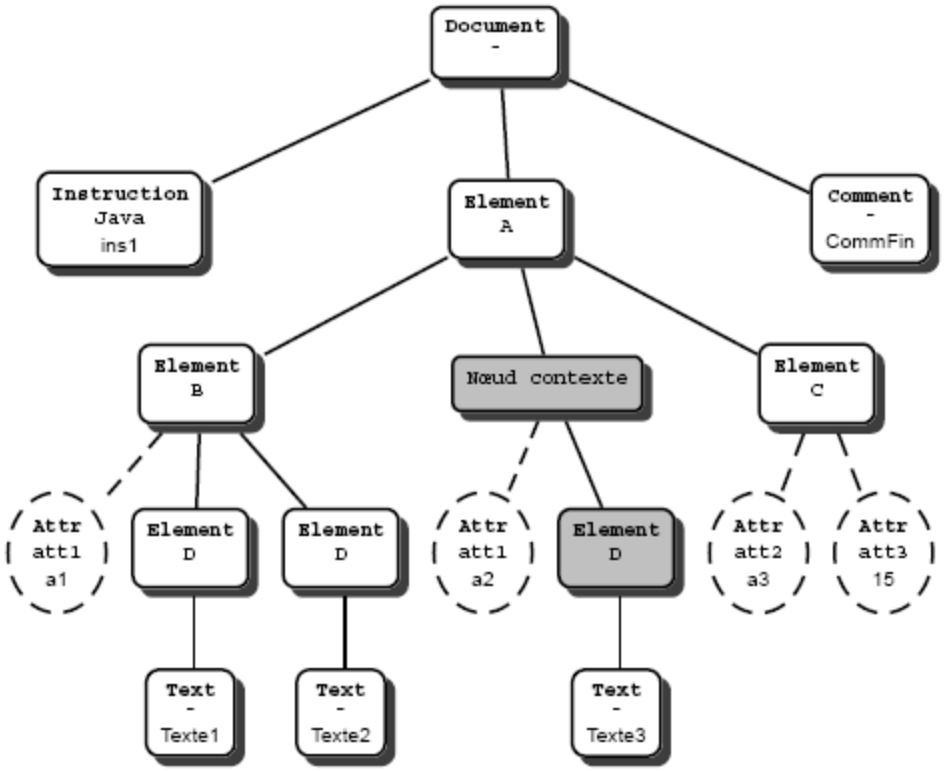
Exemple de référence



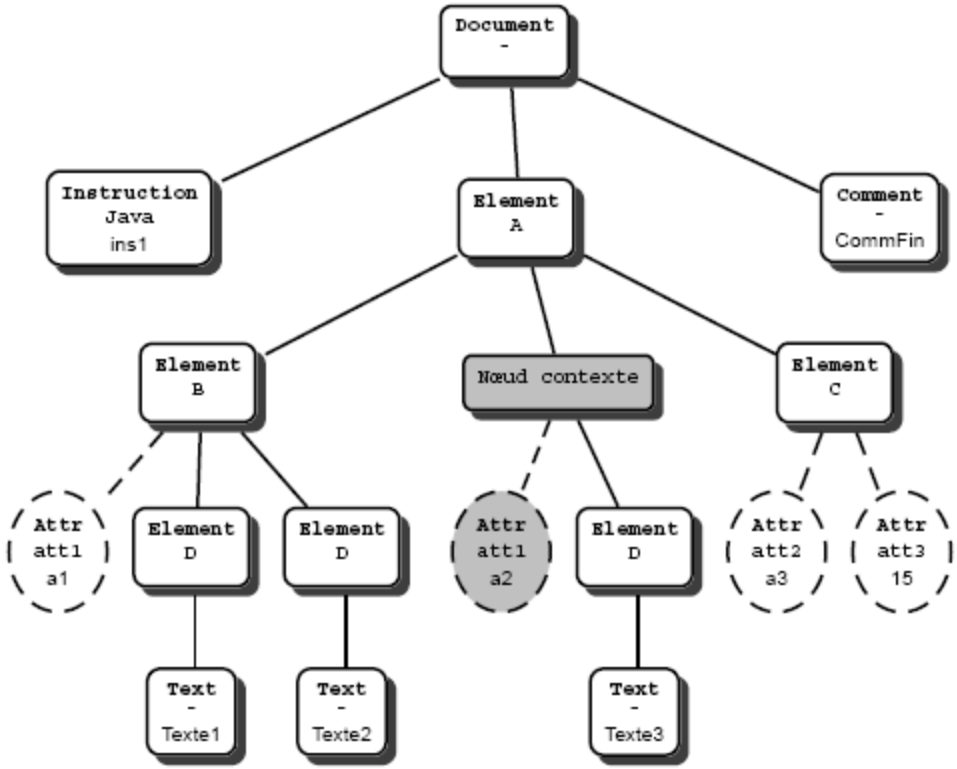
Exemple : le noeud contexte



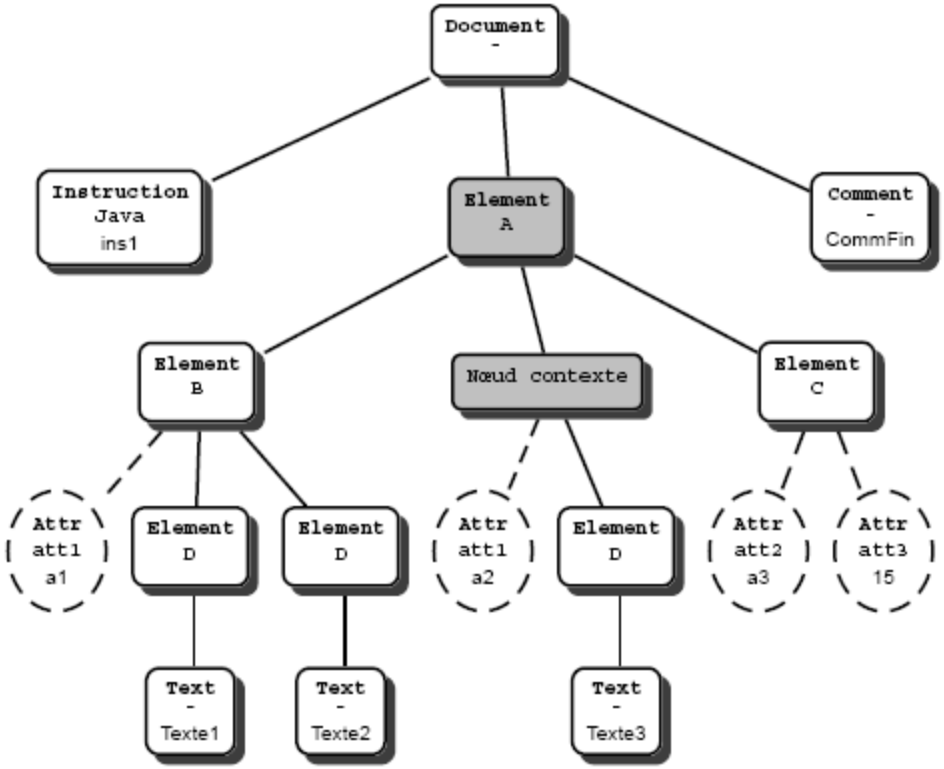
child::D ou D


Document
-
Instruction
Java
ins1
Element
A
Comment
-
CommFin
Element
B
Nœud contexte
Element
C
Attr
att1
a1
Element
D
Element
D
Attr
att1
a2
Element
D
Attr
att2
a3
Attr
att3
15
Text
-
Texte1
Text
-
Texte2
Text
-
Texte3

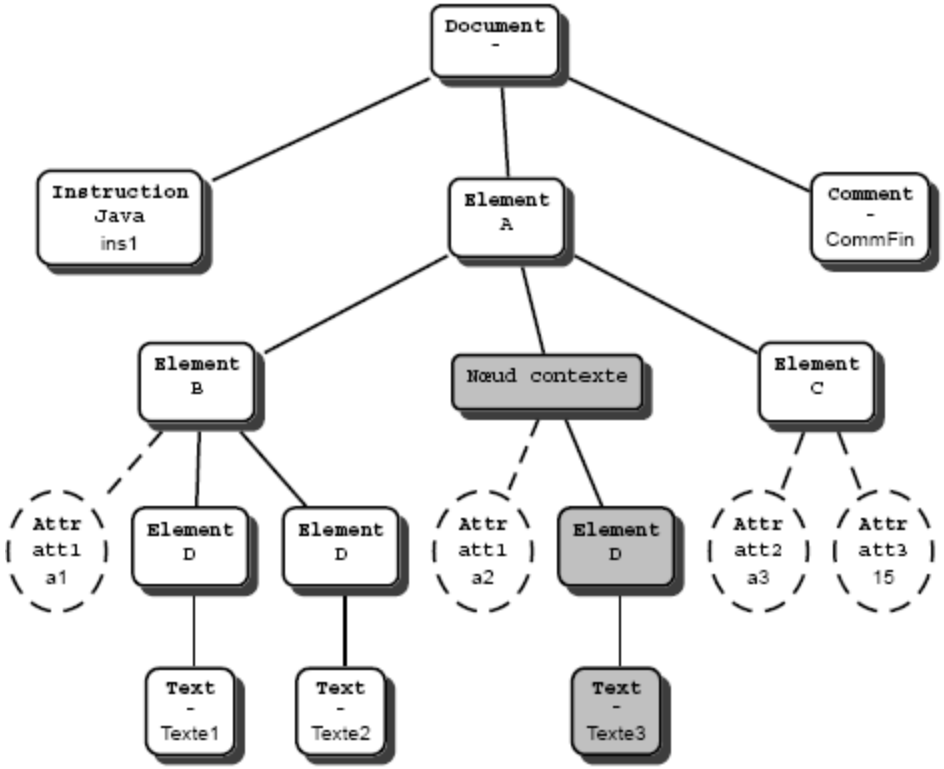
attribute::att1 ou @att1



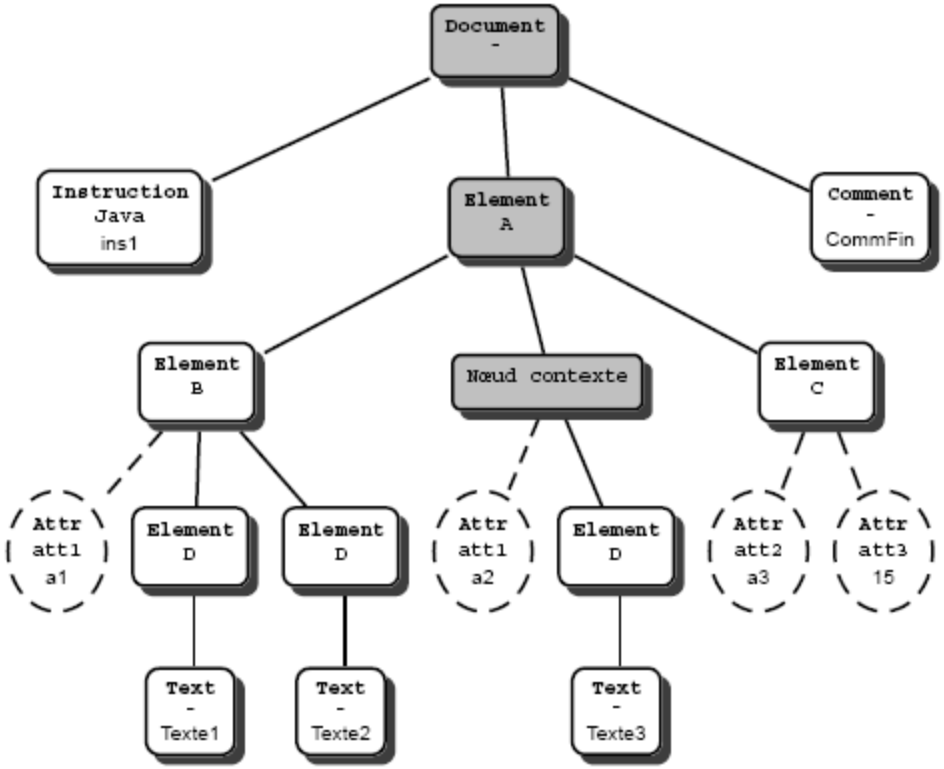
parent::A


Document
-
Instruction
Java
ins1
Element
A
Comment
-
CommFin
Element
B
Nœud contexte
Element
C
Attr
att1
a1
Element
D
Element
D
Attr
att1
a2
Element
D
Attr
att2
a3
Attr
att3
15
Text
-
Texte1
Text
-
Texte2
Text
-
Texte3

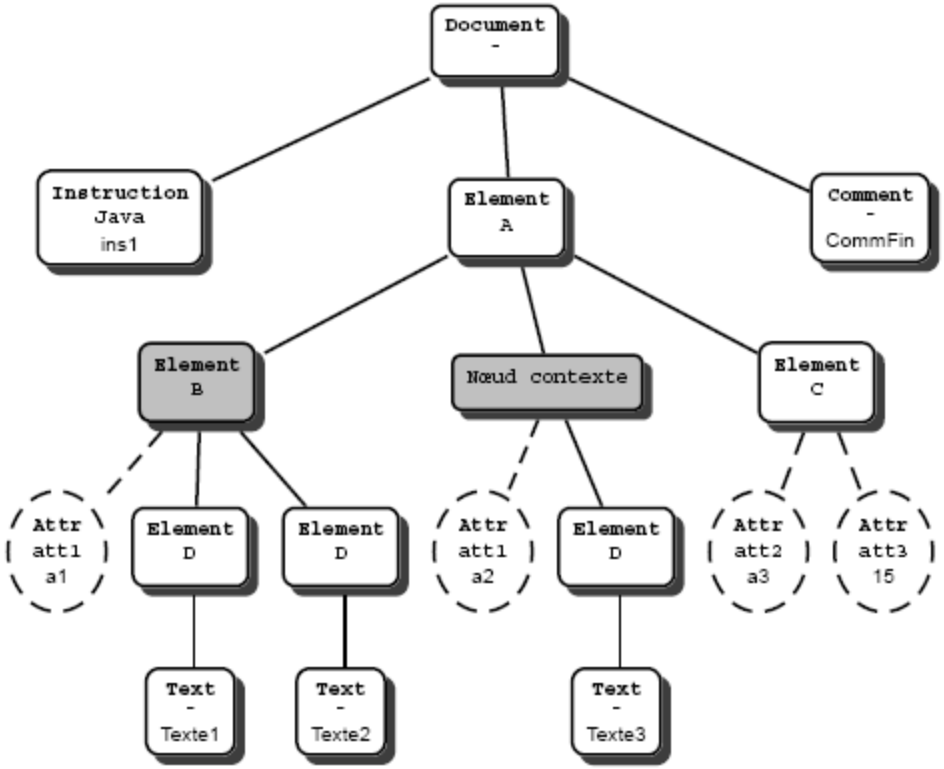
descendant::node()



ancestor::node()



preceding-sibling::node()



Axes nommés

Langage XPath prévoit des noms pour certains axes génériques :

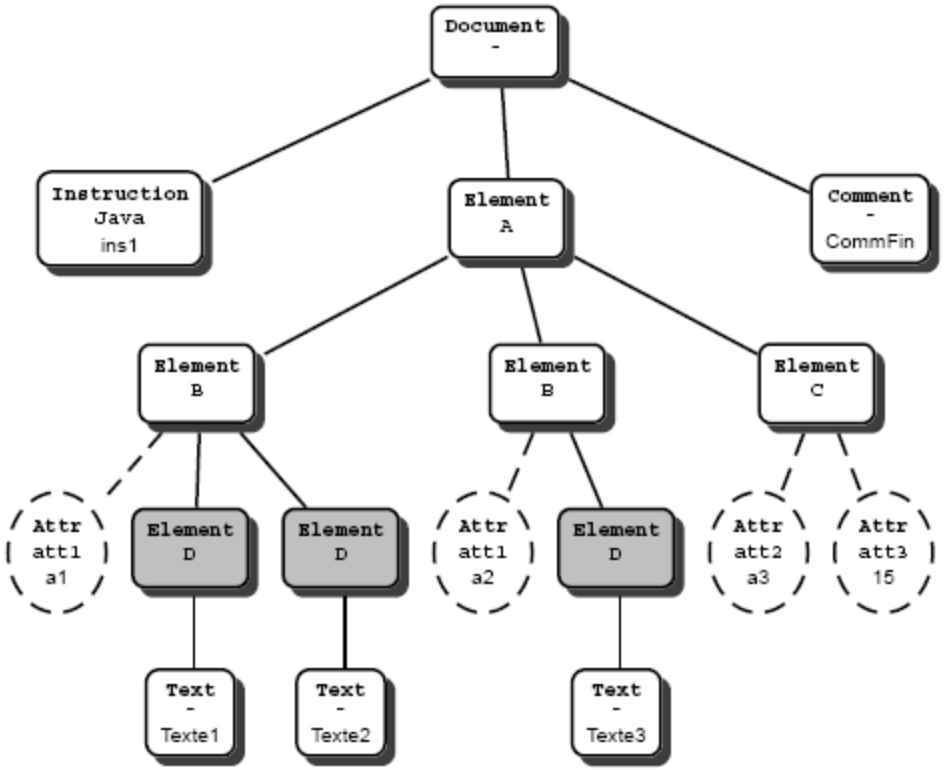
- noeuds enfants et descendants : child (axe par défaut), descendant, descendant-or-self ("//")
- noeud parent et ancêtres : parent (".."), ancestor, ancestor-or-self
- noeuds précédents et suivants : preceding, following
- frères : preceding-sibling, following-sibling
- noeud lui-même : self (".")
- noeuds attributs du noeud contexte : attribute (@)

Filtres

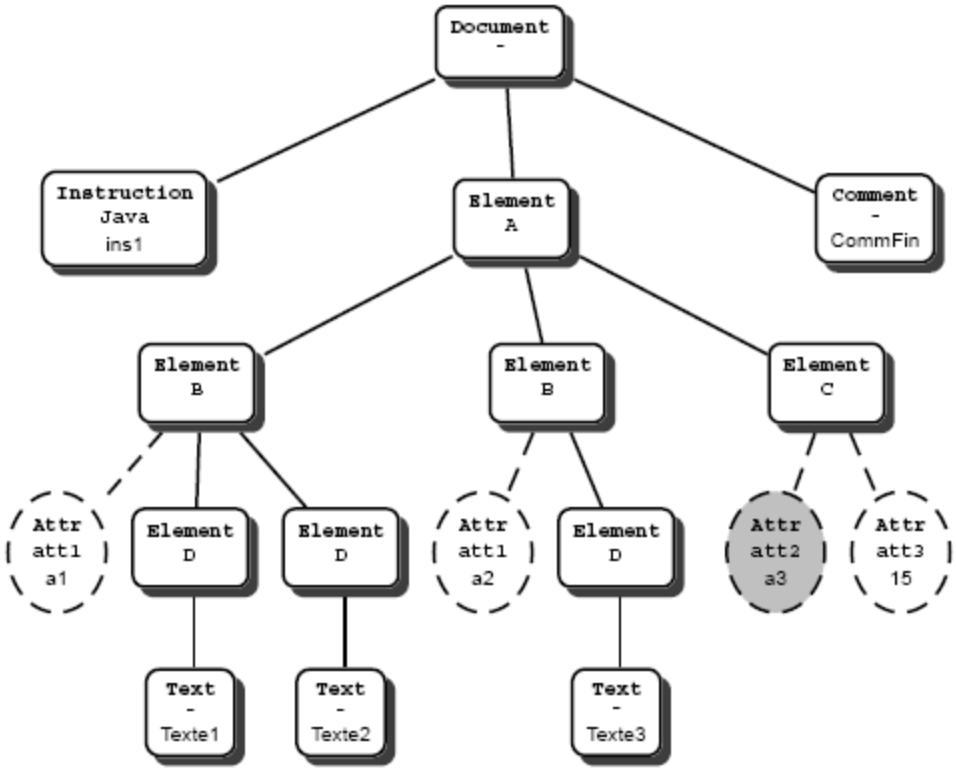
Deux manières de filtrer les noeuds :

- par leur nom :
  - `possible pour les types de noeuds qui ont un nom :`
    - Element,
    - ProcessingInstruction,
    - Attr
- par leur type DOM
  - `text(),`
  - `comment(),`
  - `processing-instruction(),`
  - `*,`
  - `node()`

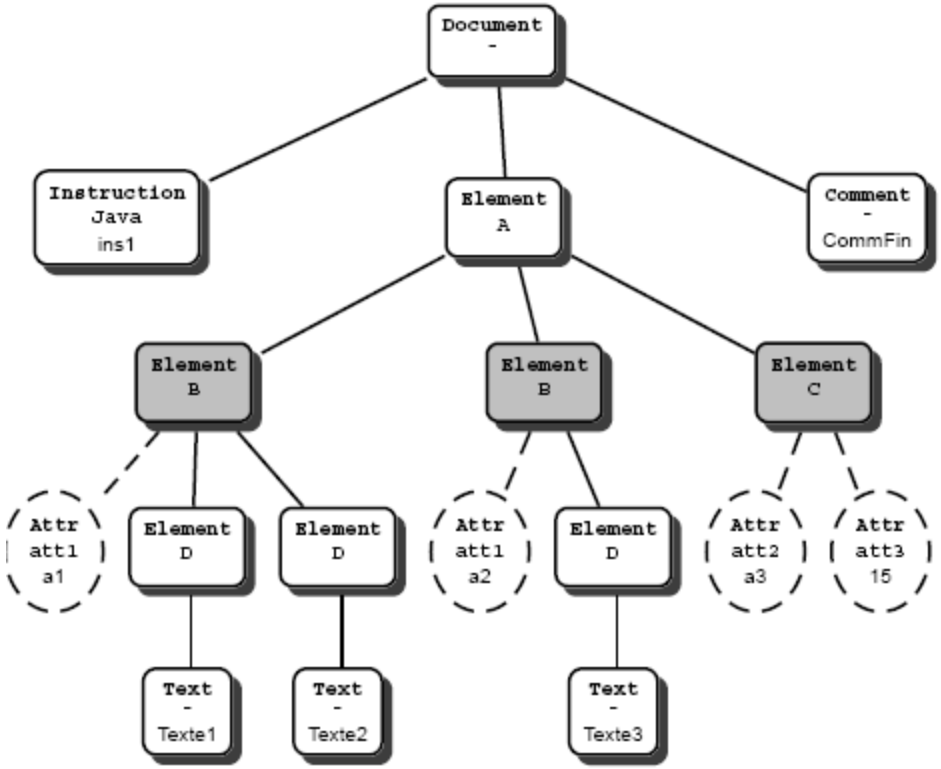
Chemin absolu : /A/B/D



/descendant::node()/@att2



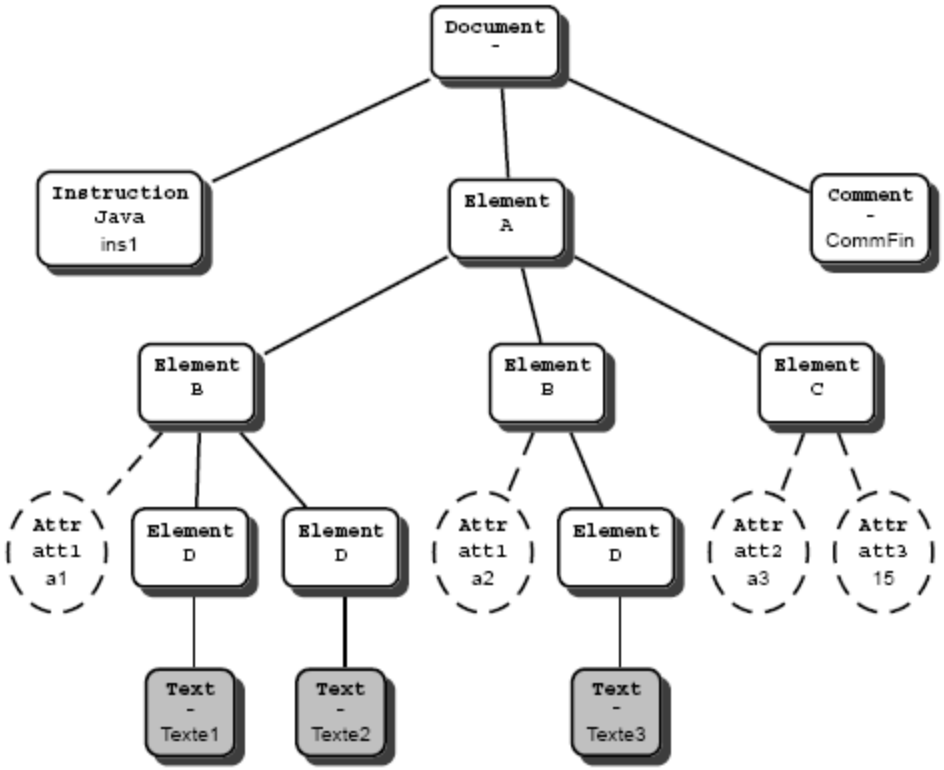
Nom générique : /A/*



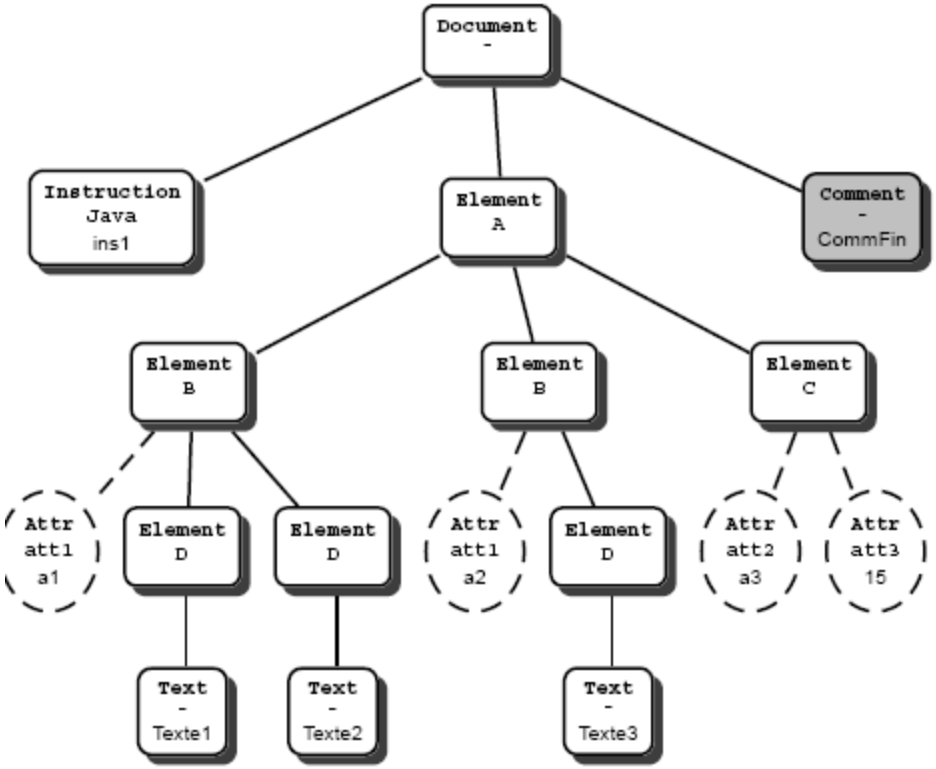
Filtrage sur le type de noeud

- text() : noeuds de type Text
- comment() : noeuds de type Comment
  - `exemple : /comment()`
- processing-instruction() : noeuds de type ProcessingInstruction
  - `exemple : /processing-instruction(), ou /processing-instruction('java')`
- node() : tous les types de noeud

/A/B//text()


Document
-
Instruction
Java
ins1
Element
A
Comment
-
CommFin
Element
B
Element
B
Element
C
Attr
att1
a1
Element
D
Element
D
Attr
att1
a2
Element
D
Attr
att2
a3
Attr
att3
15
Text
-
Texte1
Text
-
Texte2
Text
-
Texte3

/comment()


Document
-
Instruction
Java
ins1
Element
A
Comment
-
CommFin
Element
B
Element
B
Element
C
Attr
att1
a1
Element
D
Element
D
Attr
att1
a2
Element
D
Attr
att2
a3
Attr
att3
15
Text
-
Texte1
Text
-
Texte2
Text
-
Texte3

Notation abrégée de parent::A

- La notation abrégée ".." désigne le père du noeud contexte, quel que soit son type.
- L'expression ".." est équivalent à parent::node()
  - le filtre node() désigne tous les types de noeud sauf les attributs
- Attention : ".." est différent de parent::*
  - rappel : le filtre * désigne tous les *éléments* parents quel que soit leur nom

self::node() ou “.”

- L’expression “.” désigne le noeud contexte lui-même.
- Généralement elle est complétée par un filtre.
- Dans notre document exemple :
  - self::node() et self::B retournent le noeud contexte.
  - self::A renvoie un ensemble vide. Pourquoi?

Prédicats

- Prédicat : expression booléenne constituée d'un ou plusieurs tests, composés avec les connecteurs logiques habituels and et or
- Test : toute expression XPath, dont le résultat est convertie en booléen;
  - `une comparaison, un appel de fonction.`

→ il faut connaître les règles de conversion

Pour bien comprendre

Dans l'expression `/A/B[@att1]` :

- on s'intéresse aux noeuds de type B fils de l'élément racine A.
- parmi ces noeuds on ne prend que ceux pour lesquels le prédicat `[@att1]` s'évalue à true
- cette expression s'évalue avec pour noeud contexte un élément B
- `[@att1]` vaut true si et seulement si @att1 renvoie un ensemble de noeuds non vide

Quelques exemples

- les noeuds /A/B qui ont un attribut @att1 :

```
/A/B[@att1]
```

- les noeuds /A/B qui ont un attribut @att1 dont la valeur est 'a1' :

```
/A/B[@att1='a1']
```

- le premier noeud de type Text descendant d'un /A/B :

```
/A/B/descendant::text()[position()=1]
```

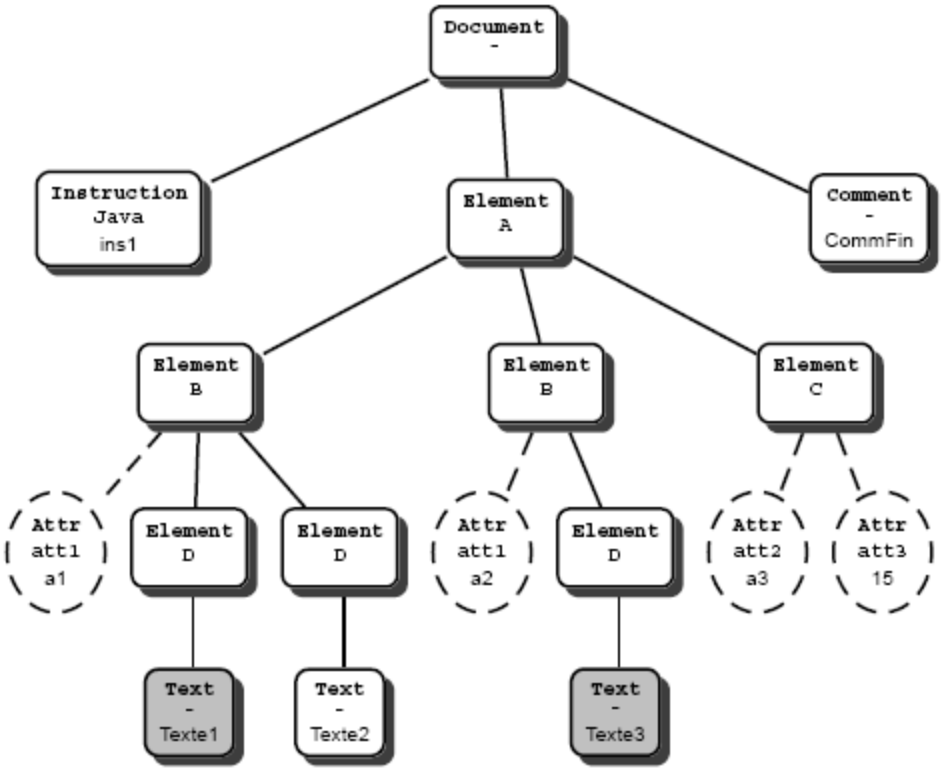
- Autre version :

```
/A/B/descendant::text()[1]
```

Contexte d'évaluation

- Une étape s'évalue en tenant compte d'un contexte constitué d'un noeud, position initiale du chemin ;
- Ce noeud fait lui-même partie d'un ensemble obtenu par évaluation de l'étape précédente
  - on connaît la taille de cet ensemble (fonction *last*())
  - on connaît la position du noeud contexte dans cet ensemble (fonction *position*())
  - cet ensemble peut être un singleton

/A/B/descendant::text()[1]



Typage avec XPath

On peut effectuer des comparaisons, des opérations. Cela implique un typage et des conversions de type.

Types XPath :

- *numériques*
- *chaînes de caractères*
- *booléens* (true et false)
- *les ensembles de noeuds*

Numériques

- Notation décimale classique
- Opérateurs logiques (<, >, !=)
- Opérations arithmétique : +, -, *, div, mod
- La fonction *number*() permet de tenter une conversion
- Si la conversion échoue on obtient NaN (*Not a Number*). À éviter...

```
//node()[number(@att1) mod 2=1]
```

Conversions

Deux conversions sont toujours possibles :

- vers une chaîne de caractères.
  - `utile pour la production de texte en XSLT (balise xsl:value-of)`
- vers un booléen
  - `utile pour les tests effectués dans XSLT (xsl:if, xsl:when)`

Fonctions

Quelques fonctions utiles dans les prédicats :

- concat(chaîne1, chaîne2, …) pour concaténer des chaînes
- contains(chaîne1, chaîne2) teste si chaîne1 contient chaîne2
- count (expression) renvoie le nombre de noeuds désignés par expression
- name() renvoie le nom du noeud contexte
- not(expression) : négation

Exemples, prédicats

Axes « d'avancement » :

- child::*[3] : le 3e enfant
- child::*[position()=3] : idem
- child::*[last()] : le dernier enfant
- descendant::*[last()] : le dernier descendant

La position d'un noeud dépend de l'axe choisi.

Prédicats : Axes inverses

Axes « inverses » :

- `ancestor::*[1]` : le premier ancêtre du noeud contexte (dernier dans l'ordre du document).
- `preceding-sibling::*[last()]` : le dernier frère qui précède le noeud contexte (premier dans l'ordre du document).

Prédicats : nombres d'occurences

Test du nombre d'occurrences :

- CINEMA[count(SEANCE) > 1] : noeuds cinémas avec au moins 2 séances
- FILM[count(ACTEUR) = 0] : noeuds films sans acteur
- FILM[not(ACTEUR)] : noeuds films sans acteur

Prédicats : sélection par valeur

Sélection par valeur :

- FILM[not(ACTEUR[NOM='Willis'])] : noeuds films sans Bruce Willis
- FILM[ACTEUR/NOM='Willis'] : noeuds films avec Bruce Willis
- FILM[ACTEUR[NOM='Willis']] : idem

Exemples, prédicats avec tests

Test sur la structure : chemins imbriqués avec connecteurs logiques (qualifiers)

- les noeuds acteurs avec un nom et une date de naissance :

  ACTEUR[NOM and DATENAISSANCE] ou ACTEUR[NOM][DATENAISSANCE]

- le noeud du film Brazil avec l'acteur De Niro :

  FILM[@TITRE = 'Brazil' and ACTEUR/NOM = 'De Niro']

- autre version :

  FILM[@TITRE = 'Brazil'][ACTEUR/NOM = 'De Niro'] :

Remarque sur les prédicats

Les deux premières expressions sont équivalentes, mais pas la troisième !

- FILM[position()=1][ACTEUR]
- FILM[ACTEUR and position()=1]
- FILM[ACTEUR][position()=1]

Pourquoi ?

Un cas épineux : les espaces

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet href="ex.xsl" type="text/xsl"?>
<!-- commentaire -->
<!DOCTYPE A SYSTEM "ex.dtd" >
<A>
<B at1=' val1 '
at2=' avec blanc '
/>
<C> </C>
<D> texte </D>
</A>
```

Où sont les espaces ?

Un peu partout !

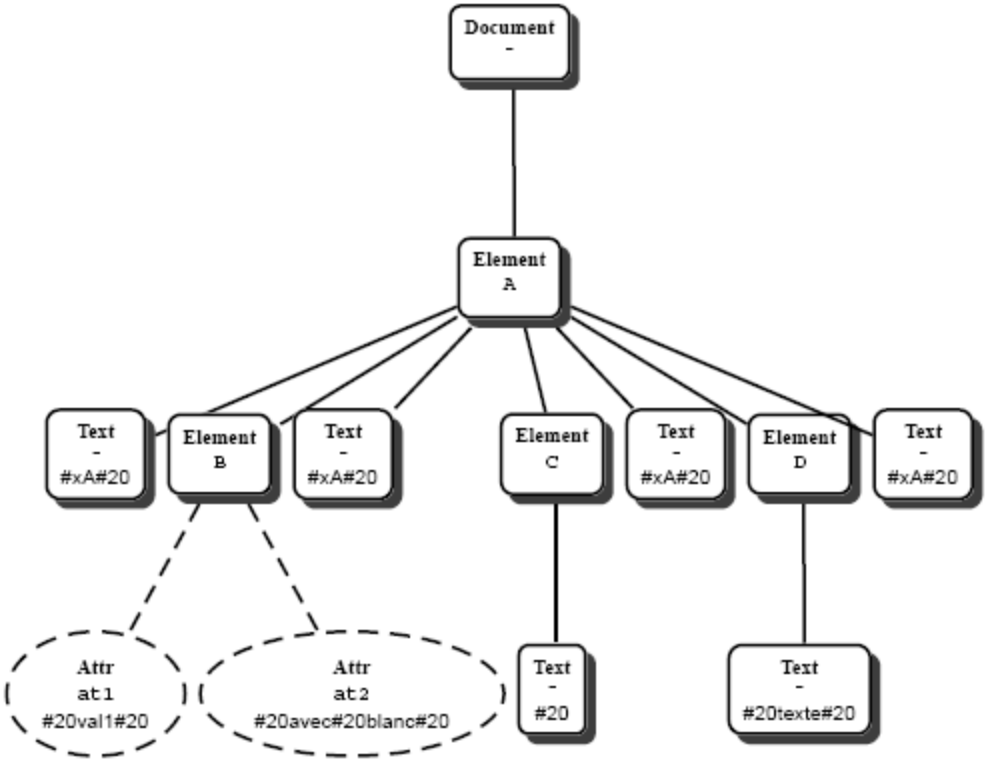
- avant la balise ouvrante de l'élément racine ;
- à l'intérieur d'une balise ;
- à l'intérieur d'un élément ;
- à l'intérieur de la valeur d'un attribut ;
- entre deux balises ;
- entre deux attributs.

Tous n'ont pas la même importance

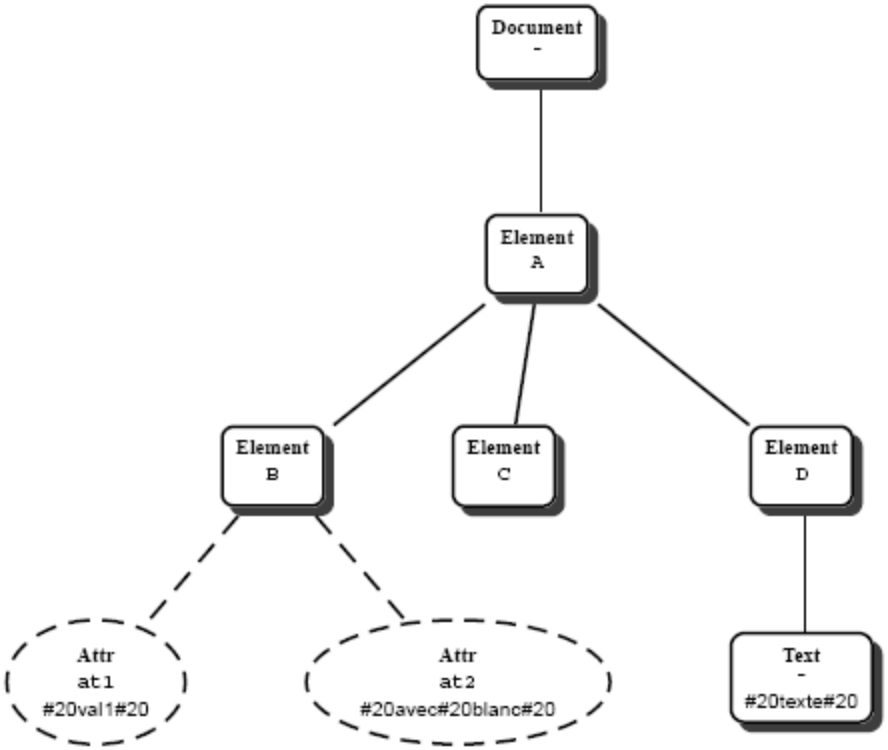
Les espaces dans XPath

- Les espaces du prologue ou de l'épilogue sont supprimés.
- Les espaces consécutifs sont réduits à un seul.
- Les fins de lignes sont représentées par le caractère #xA.
- En général, les espaces en début et fin d'attributs sont supprimées.

Arbre XPath du document



Instruction xsl:strip-space



Résumé

XPath est un langage pour **désigner des noeuds** dans un arbre XML:

- on navigue dans l'arbre grâce à des axes de navigation
- un chemin de navigation est une séquence d'étapes
- dans chaque étape on choisi un axe, un filtre et éventuellement des prédicats
- le résultat d'une étape (d'une séquence d'étapes) est un séquence de noeuds