Module MLBDA
Master Informatique
Spécialité DAC

Cours 6 –XPath

XPath

- Langage permettant de sélectionner des nœuds dans un arbre XML
- La construction de base est l'expression de chemin.
  - prédicats pour spécifier les valeurs d'éléments ou d'attributs
  - l'expression de chemin s'applique à un nœud de l'arbre et sélectionne les nœuds qu'on peut atteindre en suivant tous les chemins définis par cette expression.
- XPath est une spécification W3C (version 1.0 en 1999).

XPath

- http://www.w3.org/TR/xpath.html

- Brique de base pour d'autres standards XML
  - XSchema
  - XLink : spécification des hyperliens
  - XPointer : pointer des éléments de documents avec des expressions XPath dans les URL
  - XSLT : langage de transformation de documents
  - XQuery : langage de requêtes
  - …

Contenu

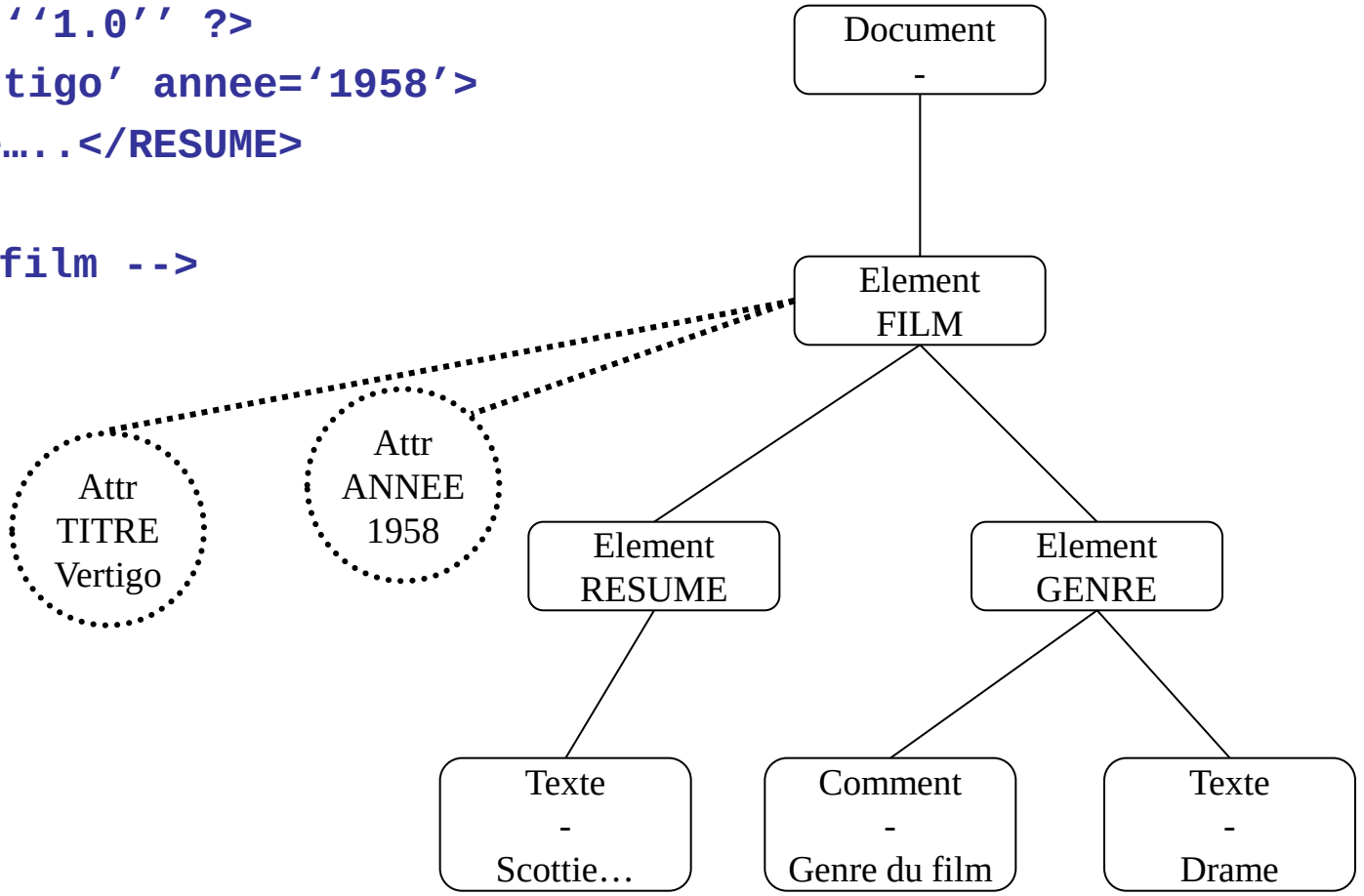
- Arbres XML
  - Définition
  - Ordre du document
  - Sérialisation
- Langage Xpath
  - Expression de chemin
  - Evaluation
  - Prédicats
  - Attributs
  - Axes

Arbres XML

- L’arbre d’un document comporte 7 types de nœuds associés à chaque constituant d’un document :
  - **Document** (document XML)
  - **Element** (élément XML)
  - **Attr** (attribut XML)
  - **Text** (valeurs textuelles)
  - **Comment** (commentaire)
  - **ProcessingInstruction** (instruction de traitement)
    - Instructions destinées à un processeur;
    - Commencent par <?
  - **Namespace** (espace de noms)
    - Permet d’éviter les conflits de noms

Exemple

```
<? xml version= ‘‘1.0’’ ?>
<FILM titre=‘vertigo’ annee=‘1958’>
<RESUME> Scottie…..</RESUME>
<GENRE>
   <-- genre du film -->
Drame
</GENRE>
</FILM>
```



Enfants et descendants

- Seuls les nœuds racine ou élément ont des **enfants** : nœuds fils de type élément, texte, commentaire ou instruction de traitement.

- Les **descendants** du nœud racine ou d'un nœud élément sont ses enfants ou les enfants de ses enfants.

⇒ Les enfants et les descendants d'un nœud ne sont donc pas des nœuds de type attribut ou espace de noms.

Ordre du document

- L’ensemble des nœuds de l’arbre d’un document est muni d’un ordre : l’**ordre du document** qui est l’ordre de lecture, dans le document XML, des constituants représentés par chaque nœud.

Elément, contenu, valeur

- Un document est un arbre d'éléments.
- Ne pas confondre
  - le contenu de l'élément (ce qui est délimité par ses balises, càd un fragment XML (ou un sous-arbre))
  - Avec sa valeur

Valeur textuelle

- Chaque nœud a une **valeur textuelle** :
  - la **valeur textuelle** du nœud racine/d'un nœud élément est la concaténation des valeurs textuelles de ses descendants de type texte, dans l'ordre du document,
  - la **valeur textuelle** d'un nœud texte est la chaîne de caractères constituant ce texte,
  - la **valeur textuelle** d'un nœud attribut est la chaîne de caractères constituant la valeur de cet attribut.

Sérialisation d’un arbre XML

Passage de la forme arborescente à la forme sérialisée :

Parcours de l’arbre par la racine, puis de gauche à droite et en profondeur d’abord :

Le nœud Document génère `<? xml version= ‘‘1.0’’ ?>`

A la première visite d’un nœud Element, on crée une balise ouvrante du nom de l’élément, et on ajoute un attribut pour chaque nœud attribut visité :

```
<FILM titre=‘vertigo’ annee=‘1958’>
```

Lorsqu’on sort d’un élément, on crée la balise fermante de l’élément.

XPath, langage de navigation

- XPath permet de naviguer
  - Dans l'arbre du document
  - A partir d'un nœud origine
  - Vers un ou plusieurs nœuds destinations
  - En sélectionnant des chemins dans l'arbre

Xpath, langage de requêtes ?

- XPath n’est pas un langage de requêtes : le contenu d’un élément (fragment correspondant au sous-arbre de l’élément) destination n’est pas extrait.
- Cependant XPath comporte un système riche de fonctions et d’opérations : calcul sur le contenu d’un élément destination, p.ex valeur textuelle
- XPath est utilisé pour faire des requêtes sur les documents dans des langages de requêtes (Xquery) ou de transformation de documents comme XSLT.

Navigation

- Modèle de données XML : graphe dont les nœuds sont des éléments.
- Expression de chemins
  - séquence de noeuds T1, T2, ..., Tn
  - retourne un ou plusieurs noeuds Tn, tels qu'il existe des arcs T1→ T2, ... Tn-1→Tn
- Une expression de chemins XPath permet de sélectionner des chemins à parcourir dans l'arbre du document en partant d'un nœud origine jusqu'à un ou plusieurs nœuds destination.

Expression de chemin

- Expression de chemin simple :
  - nom de racine suivi par une séquence de balises

- Expression de chemin généralisée :
  - utilisation de patterns (sous-chaîne de caractère, disjonction, optionalité, nombre quelconque de balises).

Expression de chemin

- Le principe de la syntaxe XPATH est semblable à celui de l'adressage du système de fichier. C'est une suite d'étapes, séparées par des '/' :

$$[/]\text{étape}_1/\text{étape}_2/\ldots/\text{étape}_n$$

- Une étape a la forme suivante :

$$\text{Axe::filtre}[\text{prédicat}_1][\text{prédicat}_2]$$

  - L'***axe*** définit le sens du parcours des nœuds
  - Le ***filtre*** indique le type des nœuds qui seront retenus dans l'évaluation de l'expression
  - Le (ou les) ***prédicat(s)*** expriment des propriétés que doivent satisfaire les nœuds retenus.

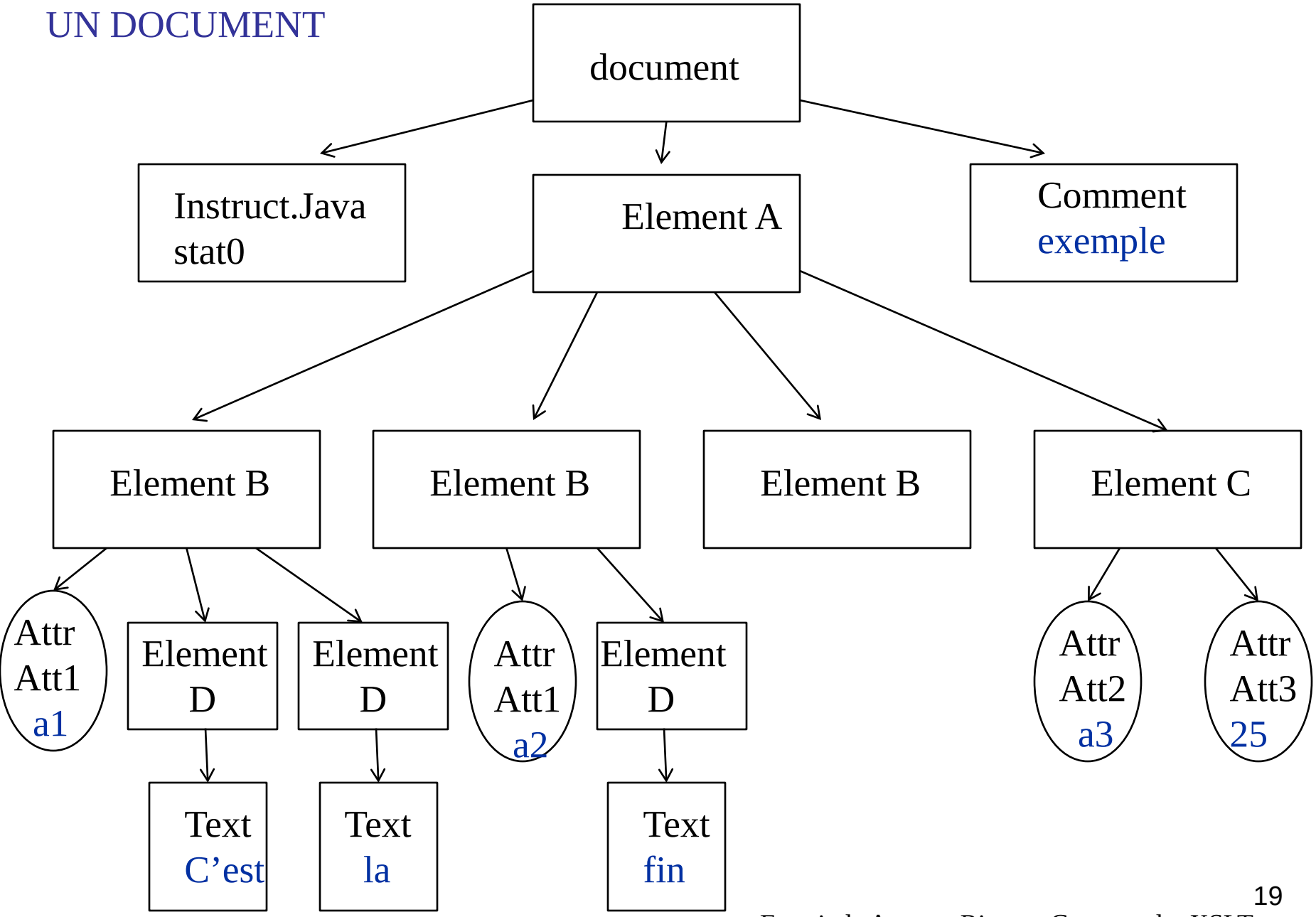
Evaluation d'une expression

- L'évaluation d'une étape peut être une valeur, ou un ensemble de nœuds (node-set).
- Le nœud origine du chemin est le premier nœud contexte.
- Les étapes sont évaluées les unes après les autres :
  - À partir du nœud contexte, on évalue l'étape 1, qui renvoie un ensemble de nœuds.
  - Chaque nœud de cet ensemble est considéré comme nœud contexte pour l'évaluation de l'étape 2.
  - On procède de la même manière pour les autres étapes.
- Remarques
  - Les nœuds ne sont pas extraits du document
  - Un nœud ne peut être référencé qu'une seule fois dans un même ensemble.

XPath (types)

- Types
  - Boolean, Number, String, Node-sets (sous-arbre)
- Types de noeuds :
  - processing instruction nodes (instructions)
  - comment nodes (commentaires)
  - root nodes (racine)
  - element nodes (élément)
  - attribute nodes (attribut)
  - namespace nodes (attributs d’un noeud)
  - text nodes (contenu)

UN DOCUMENT



Extrait de Amann, Rigaux, Comprendre XSLT

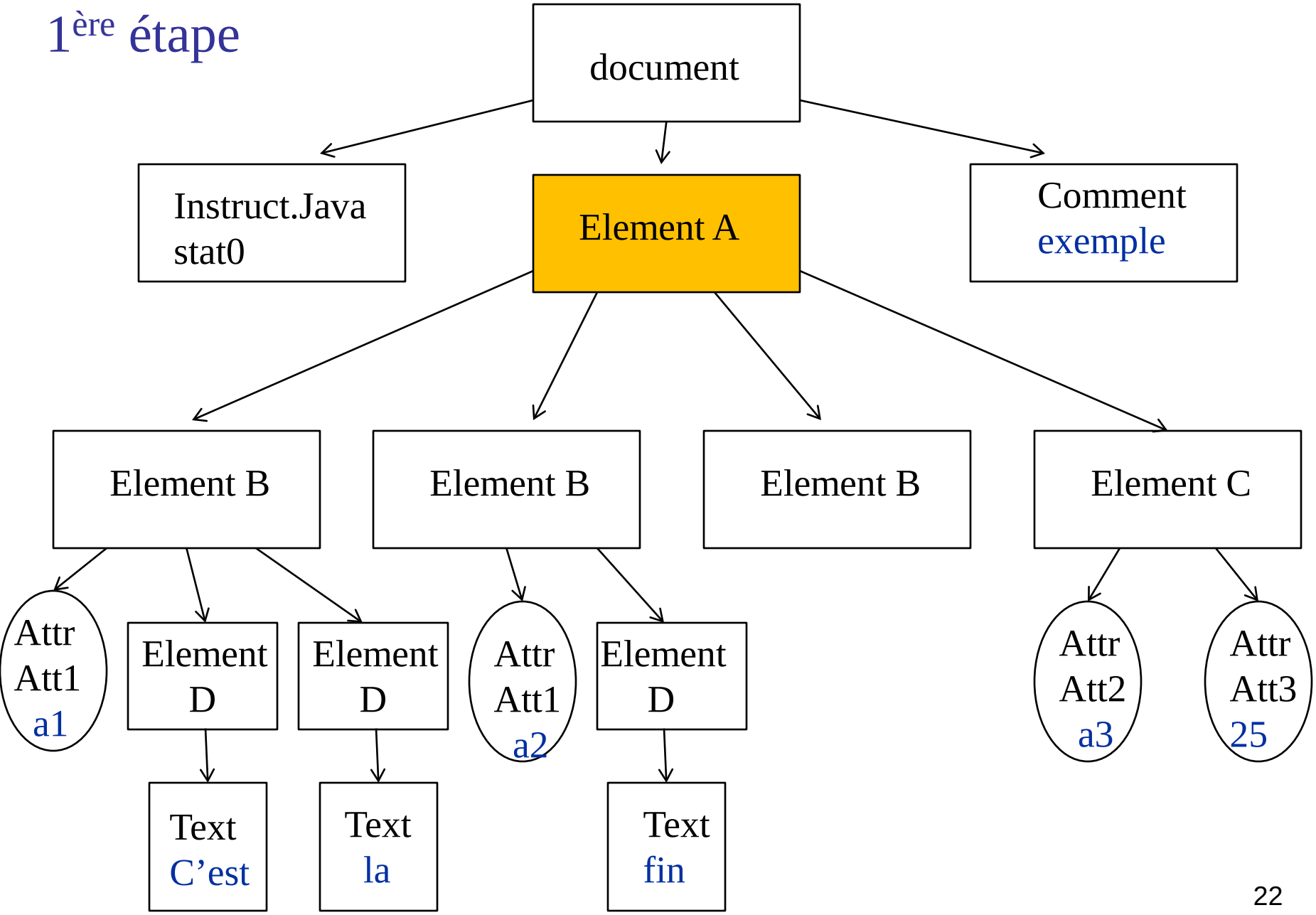
Le même sous forme sérielle

```
<?xml version= ’1.0’ encoding=‘ISO-8859-1’?>
<?java stat0?>
<A>
   <B att1=‘a1’>
       <D>C’est</D>
       <D>la</D>
   </B>
   <B att1=‘a2’>
       <D>fin</D>
   </B>
   <B/>
   <C att2=‘a3’ att3=‘25’/>
</A>
<!-- exemple-->
```

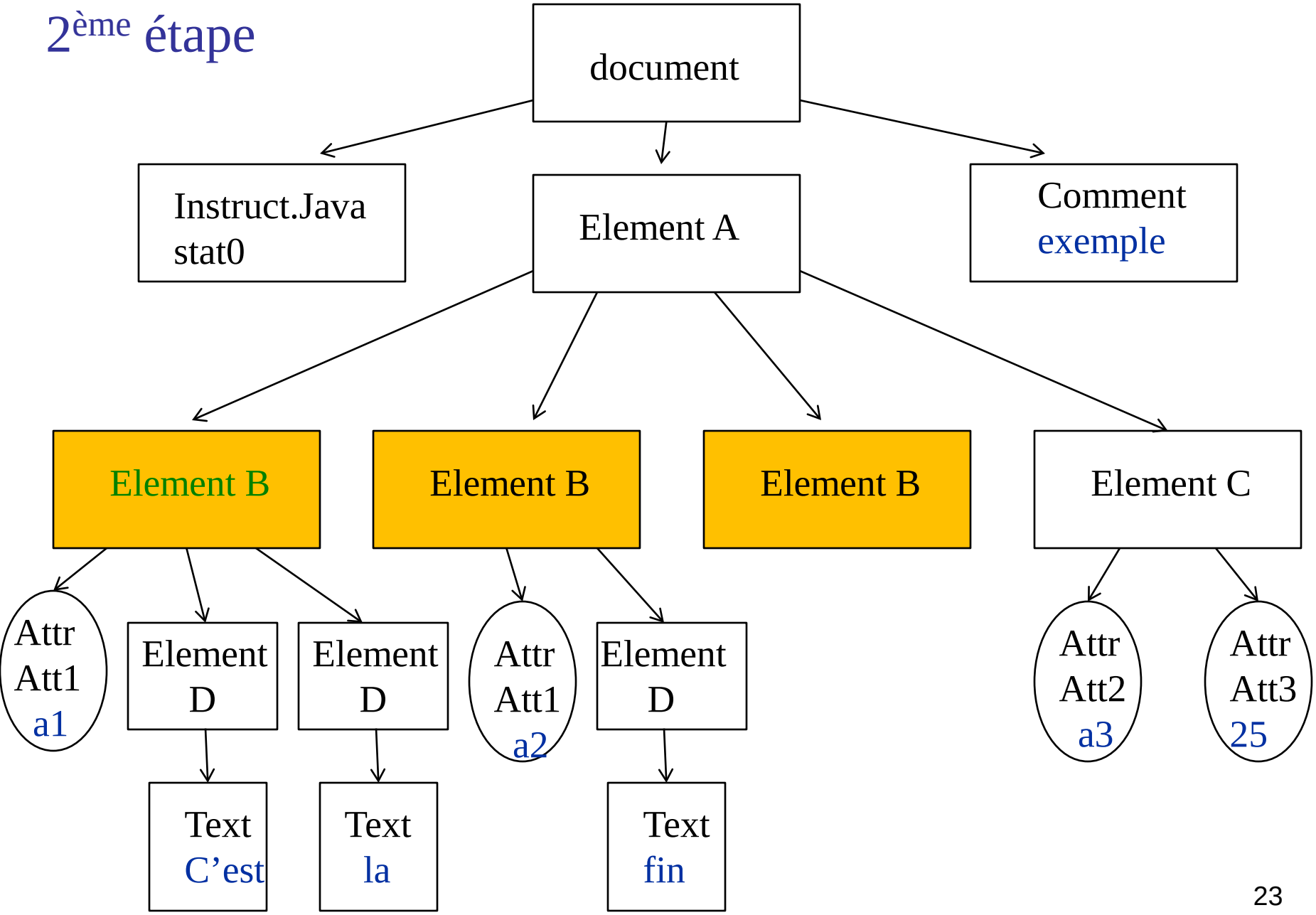
Exemple d'évaluation
/A/B/@att1

- 1ère étape : /A/B/@att1
  - A partir de la racine, on cherche les nœuds de type élément de balise A
- 2ème étape : B/@att1
  - À partir d'un nœud contexte A, on cherche les nœuds de type élément de balise B
- 3ème étape : @att1
  - À partir d'un nœud contexte B on cherche les nœuds de type attribut att1.

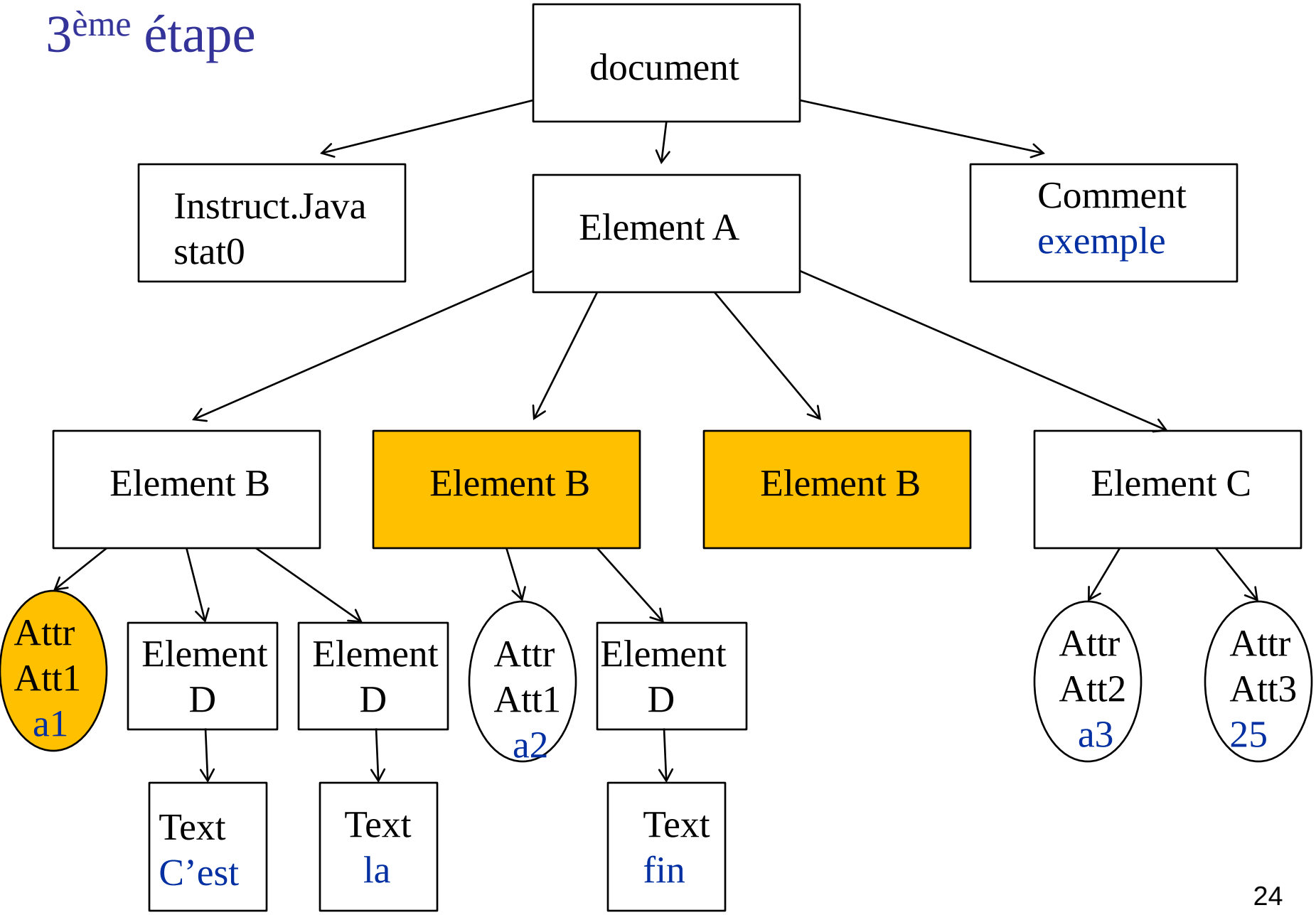
1ère étape


document
Instruct.Java
stat0
Element A
Comment
exemple
Element B
Element B
Element B
Element C
Attr
Att1
a1
Element
D
Element
D
Attr
Att1
a2
Element
D
Attr
Att2
a3
Attr
Att3
25
Text
C'est
Text
la
Text
fin

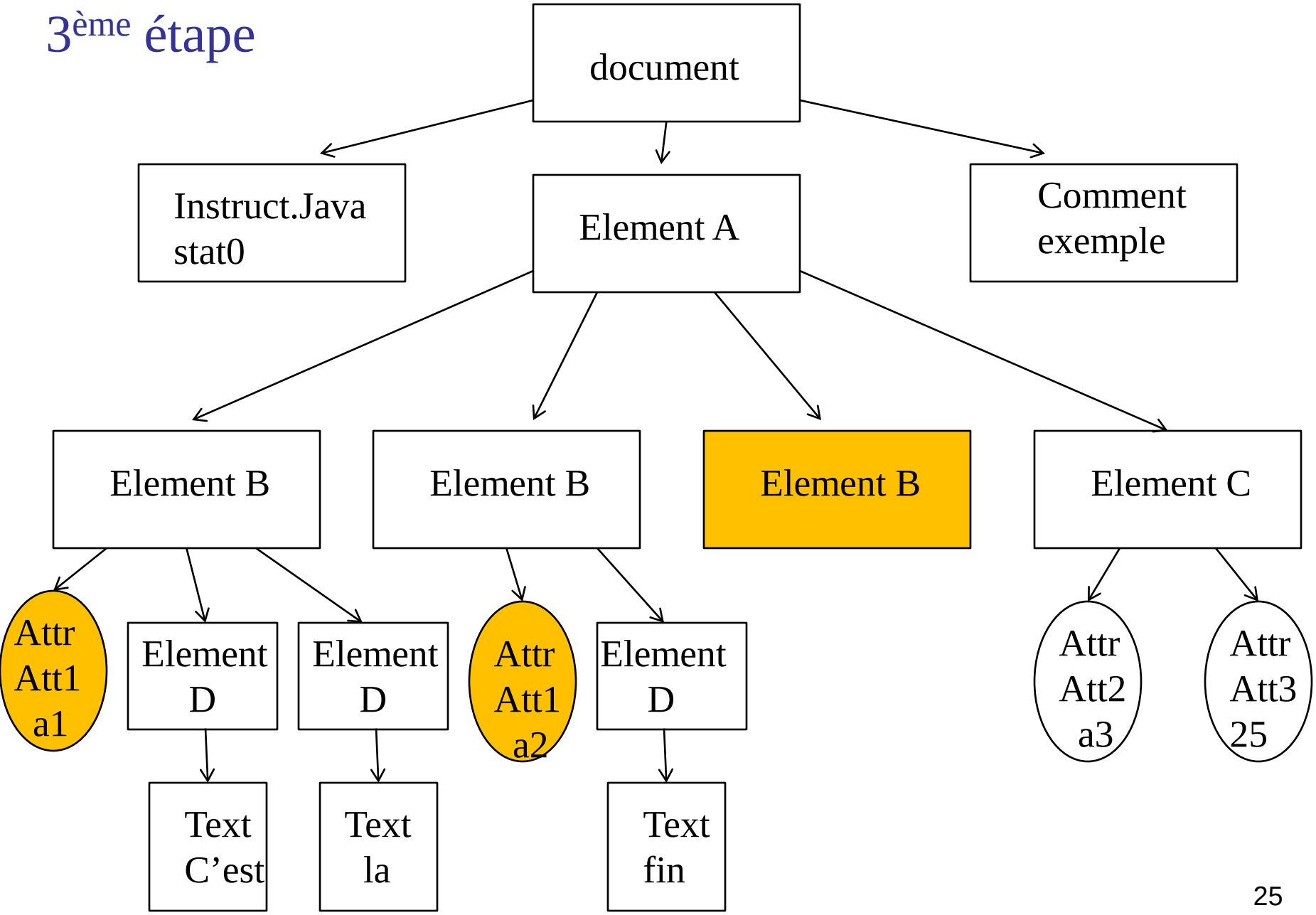
2ème étape


document
Instruct.Java
stat0
Element A
Comment
exemple
Element B
Element B
Element B
Element C
Attr
Att1
a1
Element
D
Element
D
Attr
Att1
a2
Element
D
Attr
Att2
a3
Attr
Att3
25
Text
C'est
Text
la
Text
fin

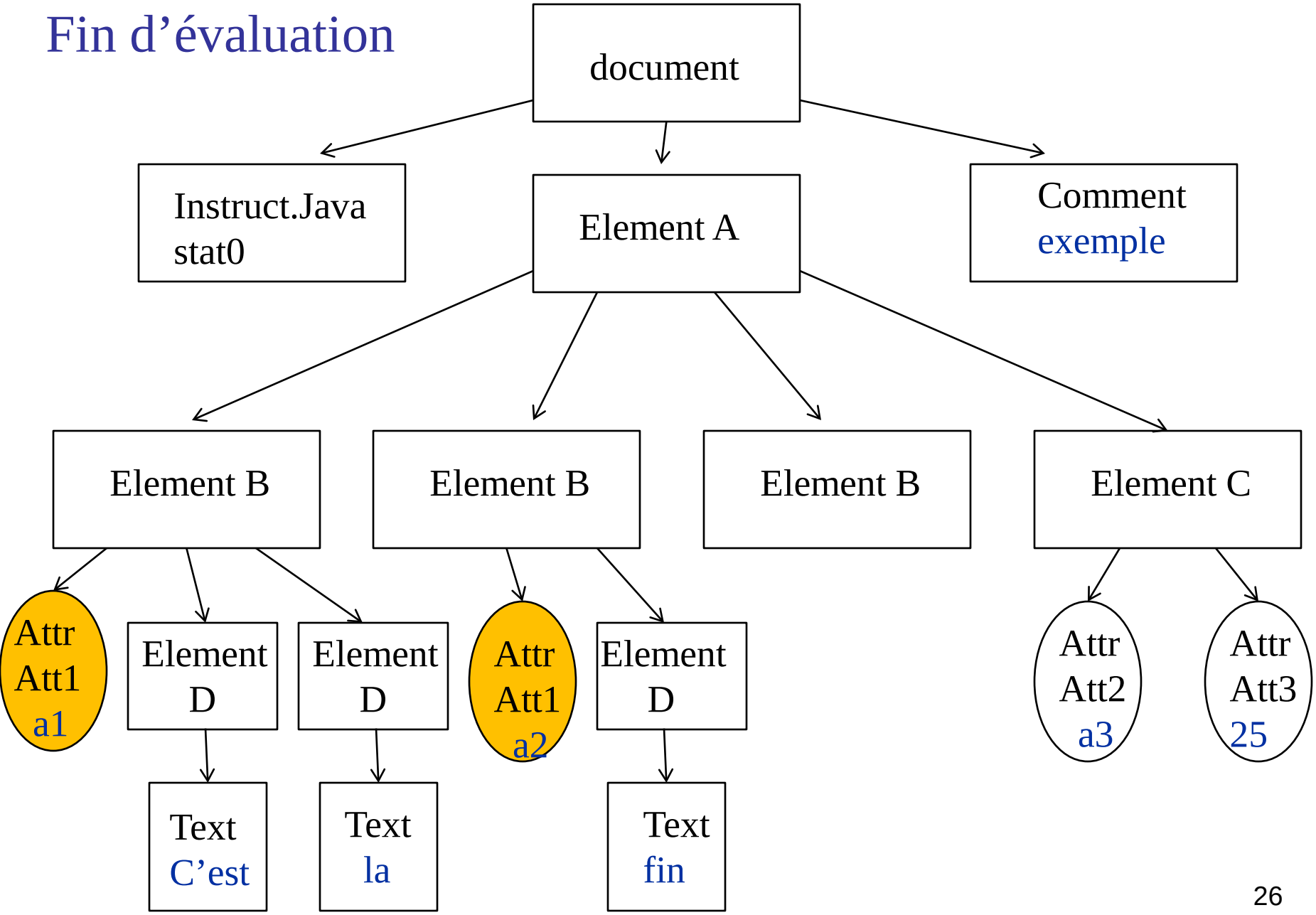
3ème étape


document
Instruct.Java
stat0
Element A
Comment
exemple
Element B
Element B
Element B
Element C
Attr
Att1
a1
Element
D
Element
D
Attr
Att1
a2
Element
D
Attr
Att2
a3
Attr
Att3
25
Text
C'est
Text
la
Text
fin

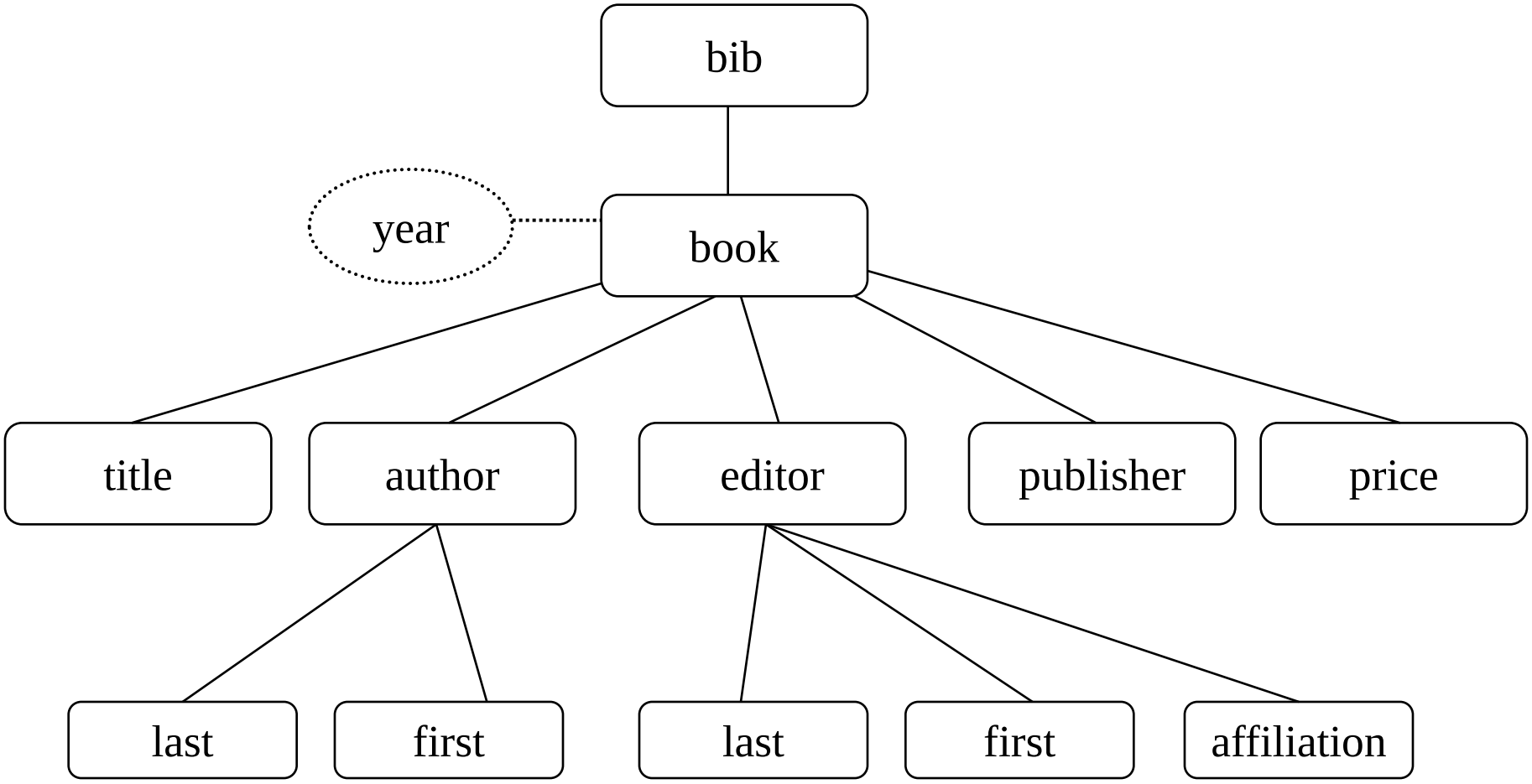
3ème étape


document
Instruct.Java
stat0
Element A
Comment
exemple
Element B
Element B
Element B
Element C
Attr
Att1
a1
Element
D
Element
D
Attr
Att1
a2
Element
D
Attr
Att2
a3
Attr
Att3
25
Text
C'est
Text
la
Text
fin

Fin d'évaluation



Un autre exemple



DTD Exemple

www.bn.com/bib.xml

```
<!ELEMENT bib (book*)>
<!ELEMENT book (title, (author+| editor+),
   publisher, price)>
<!ATTLIST book year CDATA #REQUIRED>
<!ELEMENT author (last, first)>
<!ELEMENT editor (last, first, affiliation )>
<!ELEMENT title (#PCDATA )>
<!ELEMENT last (#PCDATA )>
<!ELEMENT first (#PCDATA )>
<!ELEMENT affiliation (#PCDATA )>
<!ELEMENT publisher (#PCDATA )>
<!ELEMENT price (#PCDATA )>
```

Exemples d'expressions XPath

Un chemin qui commence par /A représente un chemin d'origine le nœud contexte vers le nœud fils A.

Ex: **`/bib`** sélectionne l'élément racine **`bib`**

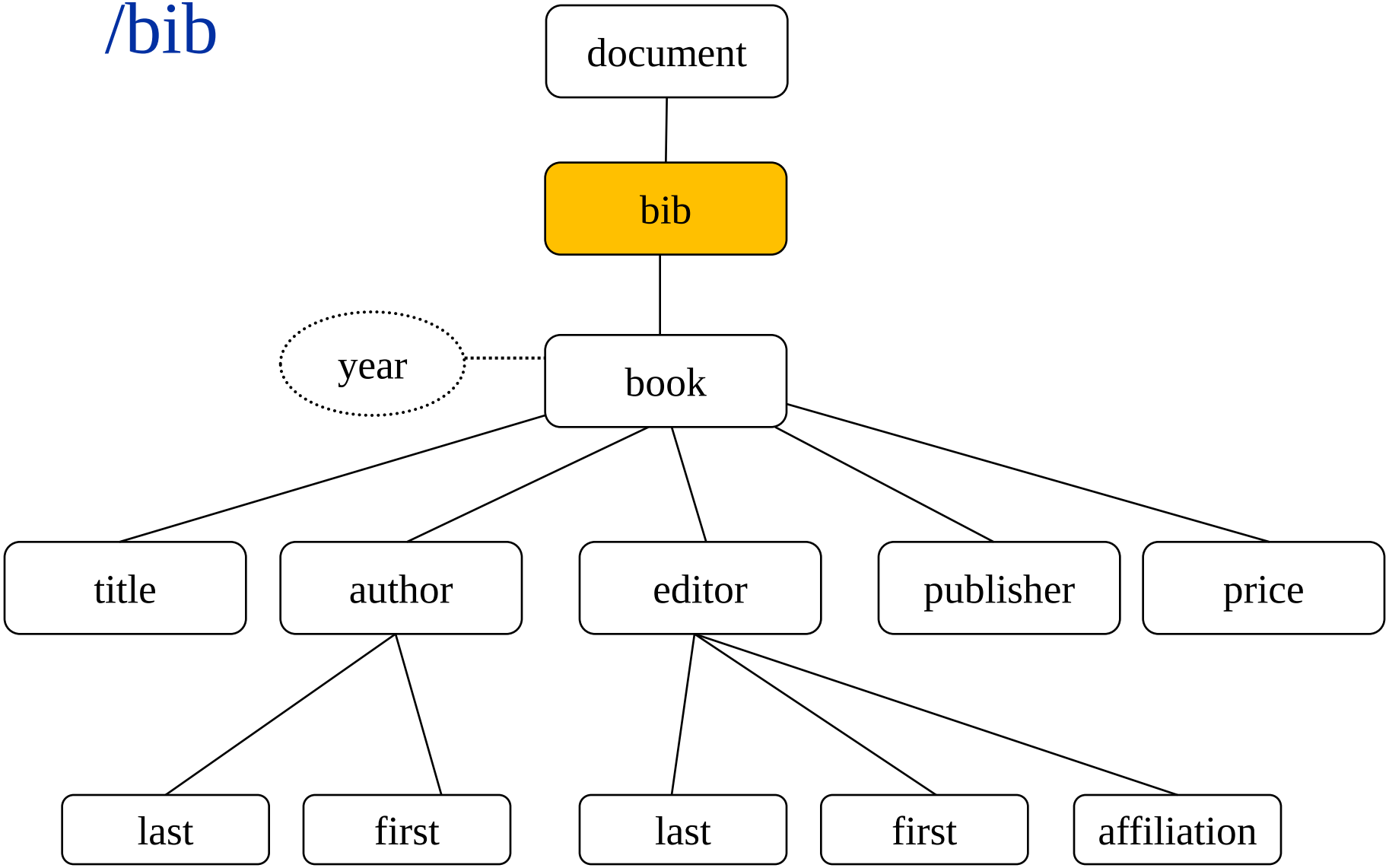
Un chemin qui commence par //A sélectionne tous les chemins qui ont pour origine le nœud contexte et destination un nœud A quels que soient les nœuds intermédiaires éventuels..

Ex: **`//editor`** sélectionne tous les éléments **`editor`** rencontrés.

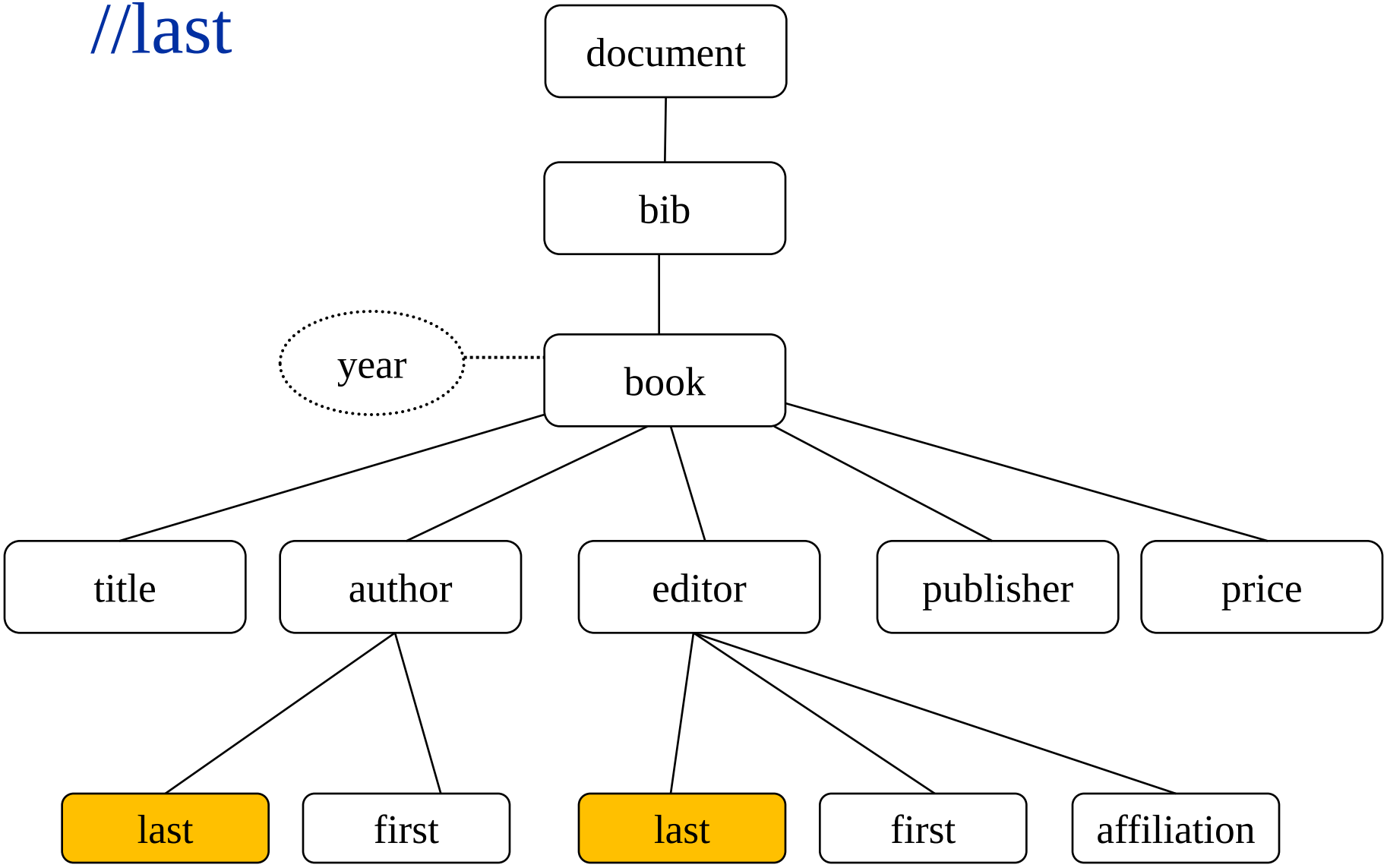
L'étoile * sélectionne tous les nœuds fils du nœud qui la précède.

Ex : **`//book/*`** sélectionne tous les fils des éléments **`book`** rencontrés

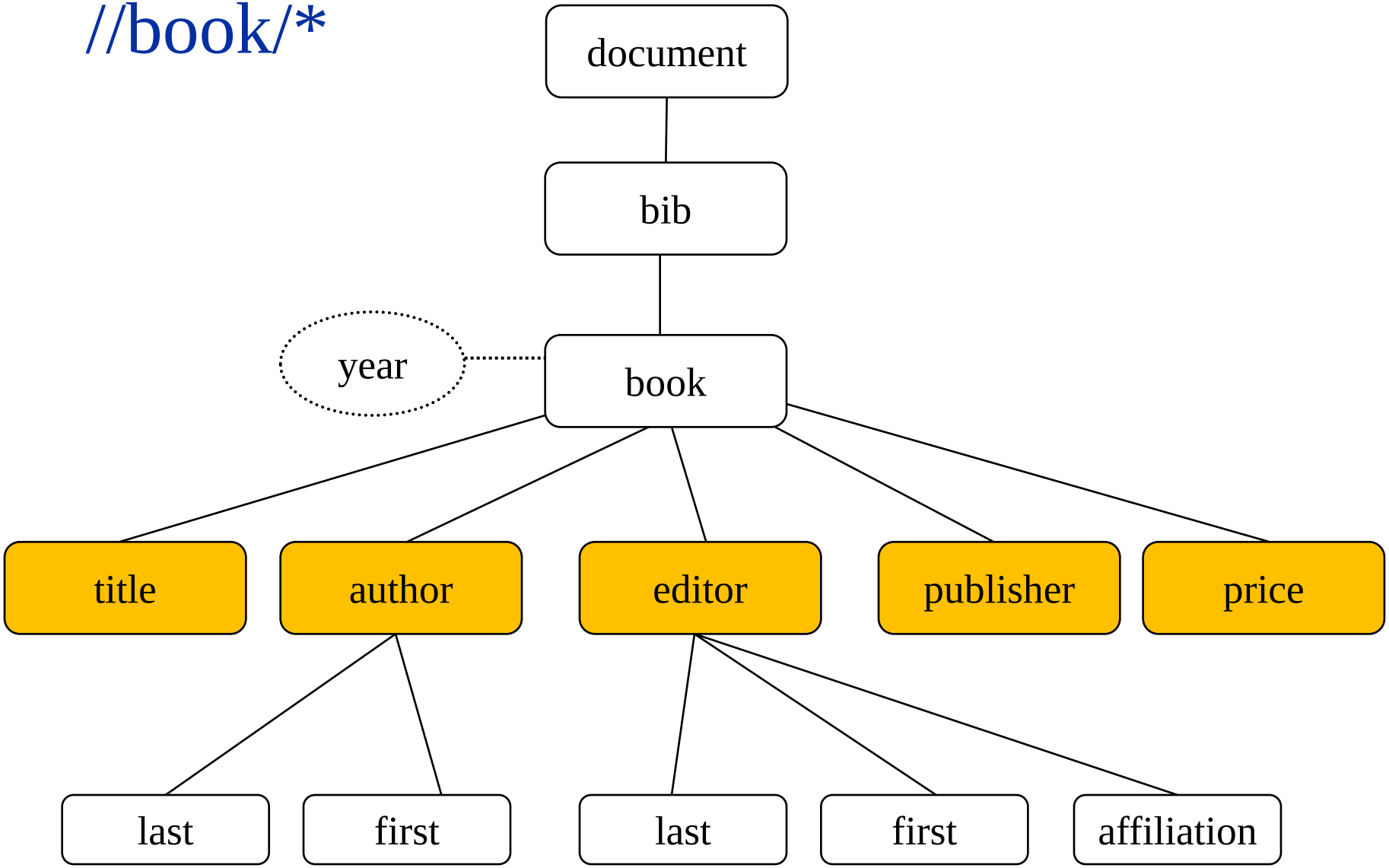
/bib


document
bib
year
book
title
author
editor
publisher
price
last
first
last
first
affiliation

//last


document
bib
year
book
title
author
editor
publisher
price
last
first
last
first
affiliation

//book/*


document
bib
year
book
title
author
editor
publisher
price
last
first
last
first
affiliation

Exemple

```
<bib>
   <book>
        <title/>
        <editor>
               <last/>
               <first/>
               <affiliation/>
        </editor>
        <publisher/>
        <price/>
   </book>
<book>
        <title/>
        <editor>
               <last/>
               <first/>
               <affiliation/>
        </editor>
        <publisher/>
        <price/>
   </book>
</bib>
```

/bib

//editor

//editor/*

Axe::filtre[prédicat]

- Dans les exemples précédents,
  - L'axe était
    - Soit child : **/A/B** raccourci pour **/child::A/child::B**
    - Soit attribut : **@a1** raccourci pour **attribute::a1**
    - **Soit descendants : // raccourci pour descendant-or-self::node()**
  - Le filtre : le type de nœud était un élément (par défaut)
  - Il n'y avait pas de prédicat

Prédicat

- Un prédicat est une expression booléenne (placée entre crochets), qui permet de spécifier un élément.
- Un prédicat est construit à partir
  - d'expressions de chemin et/ou
  - de fonctions prédéfinies (sur les nombres, les booléens, les nœuds et les chaînes.
- Les prédicats peuvent être connectés par les connecteurs **`or,and,not`**

Conditions

- Un prédicat peut être atomique :

  `/bib/book[1]` sélectionne le premier élément book de bib

  `/bib/book[author]` sélectionne les éléments **`book`** de l'élément s'ils possèdent un fils **`author`**

  `/bib/book[@year]` sélectionne les éléments **`book`** qui ont un attribut **`year.`**

- Un prédicat peut être une condition <att> <op de comparaison> <valeur>

  `/bib/book[@year= '2012']` sélectionne les éléments book ayant un attribut **`year`** de valeur 2012.

Fonctions

- Fonctions sur les nombres
  - Sum (nœuds) ?
  - Count (nœuds) : renvoie le nombre de noeuds
  - Round(nb) : renvoie l'arrondi le plus proche
- Fonctions sur les booléens
  - True()
  - False()
  - Not(booleen)

Fonctions

- Fonctions sur les nœuds :
  - Last() : vrai ssi le nœud est le dernier du contexte courant
    - `/bib/book[last()]` sélectionne le dernier élément book de bib.
  - Position() : retourne le numéro d'ordre du nœud dans le contexte courant. La première position vaut 1.
  - Name(nœuds) : renvoie l'étiquette d'un noeud
  - Id(nom) : renvoie le nœud identifié par l'étiquette nom.
- Fonctions sur les chaînes :
  - String(object) : convertit l'objet en argument en chaine de caractère
  - Contains (str1, str2) : teste si la chaîne contient la chaîne 2
  - Concat(str1, str2,..strn) : concaténation de chaînes

Prédicats

- Un prédicat (expression entre crochets) permet de spécifier plus précisément un élément.
- Un prédicat est un booléen
  - Prédicat atomique (existence d'un nœud, attribut comparateur valeur)
  - Expression de prédicat avec les connecteurs **`or,and,not`**
  - Expression XPath
- Exemples
  - Un nombre entre crochets donne la position d'un élément dans le jeu sélectionné.

    `/bib/book[1]` sélectionne le premier élément book de bib
  - La fonction **`last()`** sélectionne le dernier élément

    `/bib/book[last()]` sélectionne le dernier élément book de bib.

Exemple

```
<bib>
   <book>
        <title/>
        <editor>
               <last/>
               <first/>
               <affiliation/>
        </editor>
        <publisher/>
        <price/>
   </book>
<book>
        <title/>
        <editor>
               <last/>
               <first/>
               <affiliation/>
        </editor>
        <publisher/>
        <price/>
   </book>
</bib>
```

/bib/book[1]

/bib/book[last()]

Fonctions

- Positions relatives et information locale
  - **`position()`** : position dans le contexte
  - **`name()`** : retourne le nom de l'élément
  - **`count()`** : cardinalité d'un node-set (compte le nb d'éléments sélectionnés)
  - **`last()`** : indicateur de dernière position
  - Fonctions booléennes : **`and, or, not`**
  - opérateurs : **`mod, >, <=,`** etc.
  - fonctions de chaîne : **`contains, substring-before, string-length, ...`**
  - fonctions d'environnement : **`normalize-space`** (supprime les espaces de début et de fin, remplace les séquences d'espaces blancs par un seul espace.)
- Permettent des requêtes de base
  - analyse des contenus et noms de balises/attributs

Attributs

- Les attributs sont spécifiés par le préfixe **@**.
  - **`//@year`** sélectionne tous les attributs **`year`** du document
  - **`//book[@year]`** sélectionne tous les éléments **`book`** qui ont un attribut **`year`**.

- Les valeurs des attributs peuvent être utilisés comme critère de sélection
  - **`//book[@year="2004"]`** sélectionne tous les éléments **`book`** ayant un attribut **`year`** dont la valeur est 2004.

Exemples

- `//*[name()= "book"]`éléments **`book`**
- `//*[count(book)="2"]` éléments ayant 2 enfants **`book`**
- `//*[count(*)="2"]` éléments ayant deux enfants
- `//*[string-length(name())="3"]` retourne tous les éléments dont le nom a 3 caractères.

XPath (axes)

• Position (noeud courant)	self	.
• Descendants		
– direct	child	/
– indirect	descendant-or-self::node descendant	//
• Ancêtres		
– direct	parent	..
– indirect	ancestor , ancestor-or-self	
• Frères		
– même niveau	following- /preceding-sibling	
– Successeurs/prédecesseurs	following/preceding	
• Autres		
– namespace, attribute (@)		

Axes

- **`child`** donne les enfants du nœud contextuel. L'axe enfant est l'axe par défaut, et il peut être omis.
- **`self`** renvoie le nœud courant.
- **`parent`** contient le parent du nœud contextuel, s'il en a un.
- **`descendant`** contient tous les descendants du nœud contextuel (enfant, petit-enfant, etc.), à l'exception des nœuds attributs et espaces de nom.
- **`ancestor`** contient tous les éléments ancêtres du nœud contextuel (parent, parent du parent, etc.). Il contient forcément le nœud racine (sauf si le nœud contextuel est la racine).

Axes

- **`following-sibling`** contient tous les nœuds frères (nœuds qui ont le même parent) qui suivent le nœud contextuel.
- **`preceding-sibling`** contient tous les frères qui précèdent le nœud contextuel.
- **`following`** contient tous les nœuds du même document que le nœud contextuel qui sont après le nœud contextuel dans l'ordre du document, à l'exclusion de tout descendant, des attributs et des espaces de noms.
- **`Preceding`** contient tous les prédécesseurs du nœud contextuel, à l'exclusion des ancêtres; si le nœud contextuel est un attribut ou un espace de noms, la cible précédente est vide.

Axes

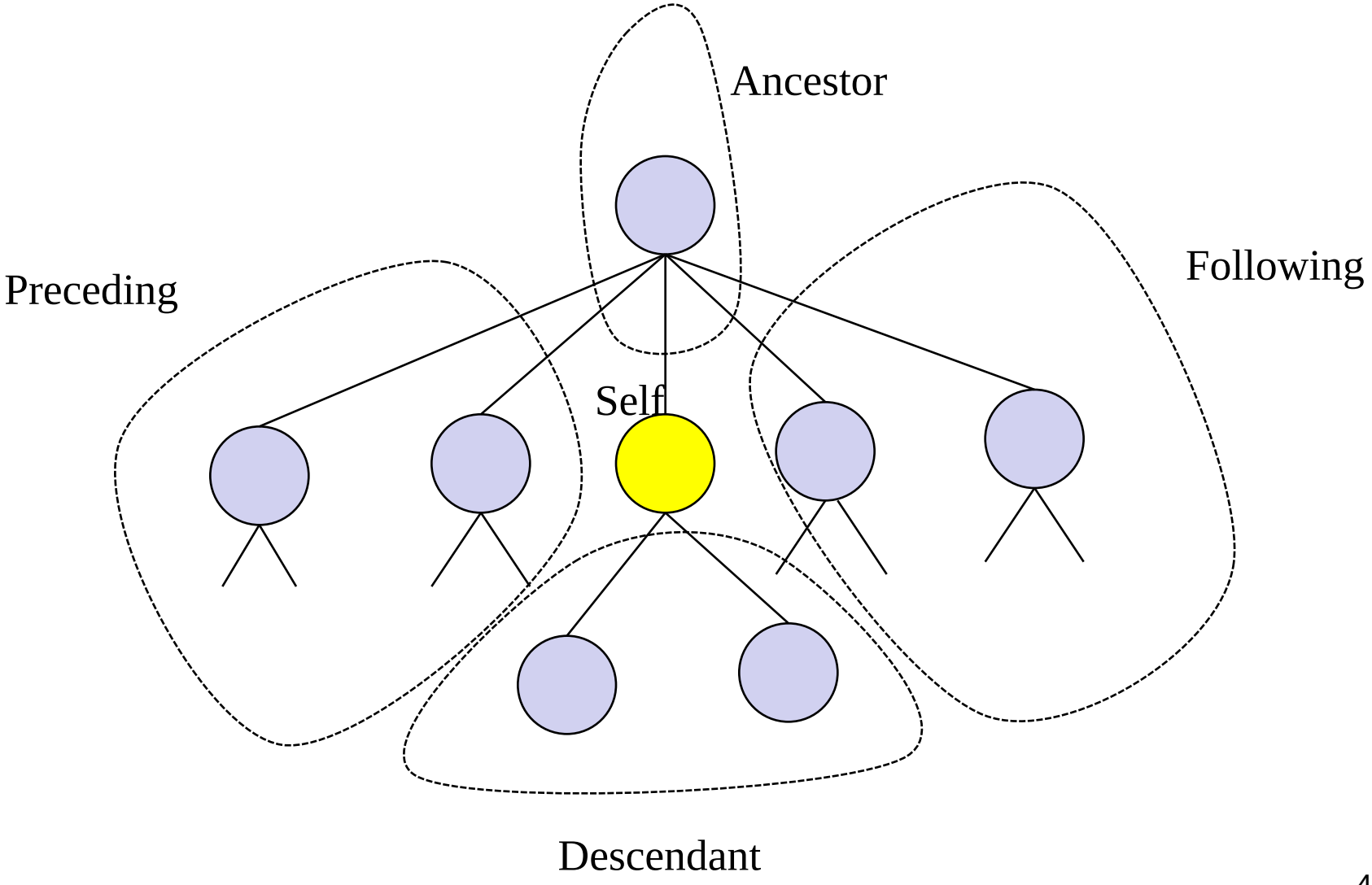
- L'axe **`descendant-or-self`** contient le nœud contextuel et ses descendants
- L'axe **`ancestor-or-self`** contient le nœud contextuel et ses ancêtres; ainsi l'axe **`ancestor-or-self`** contient toujours le nœud racine
- Les axes **`ancestor , descendant , following, preceding`** et **`self`** partitionnent un document (ignorant les attributs et les nœuds d'espace de nom) : il ne se chevauchent pas et ensemble ils contiennent tous les nœuds d'un document

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

renvoie tout le document.

Le symbole | permet de combiner des chemins.

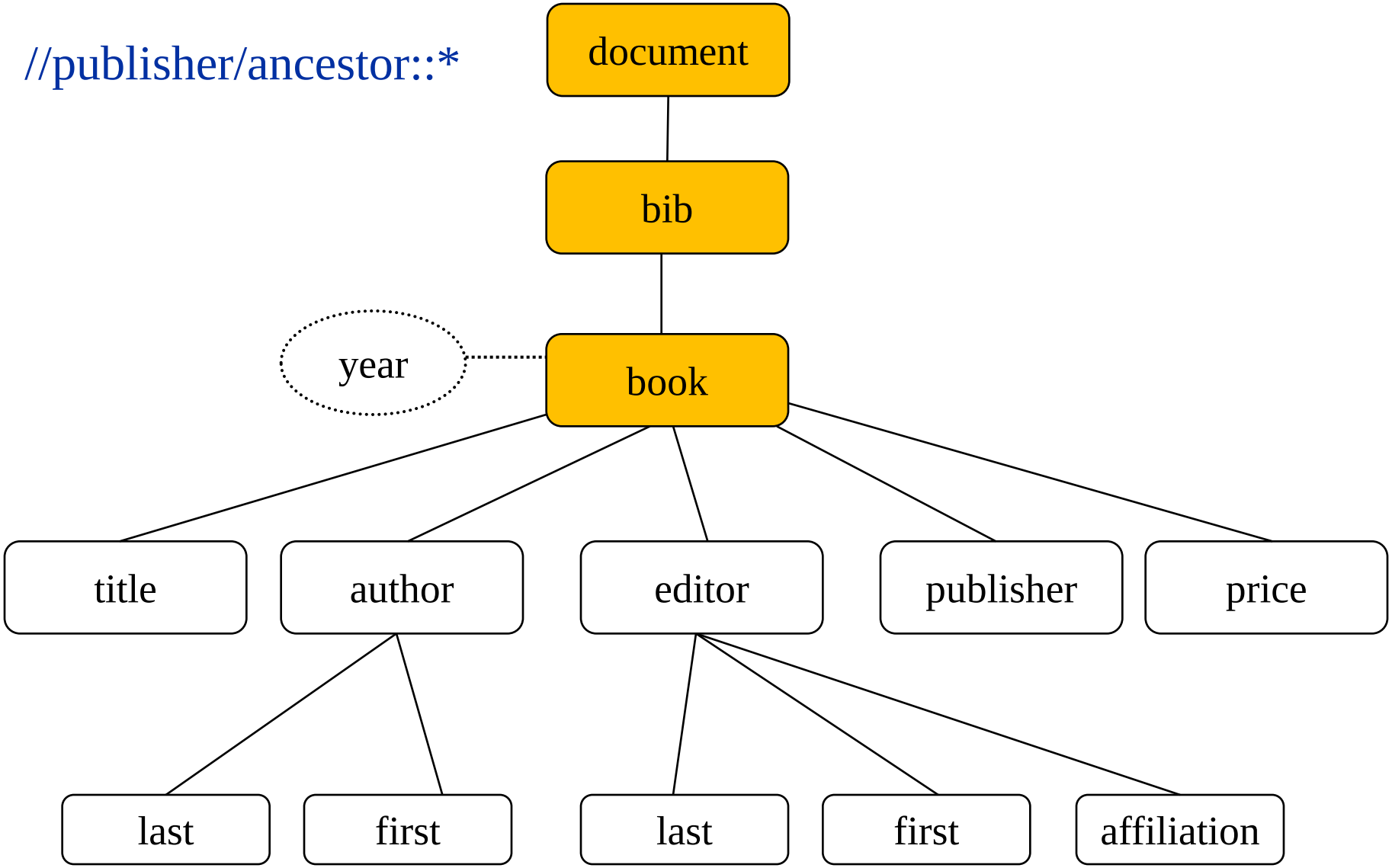
Axes



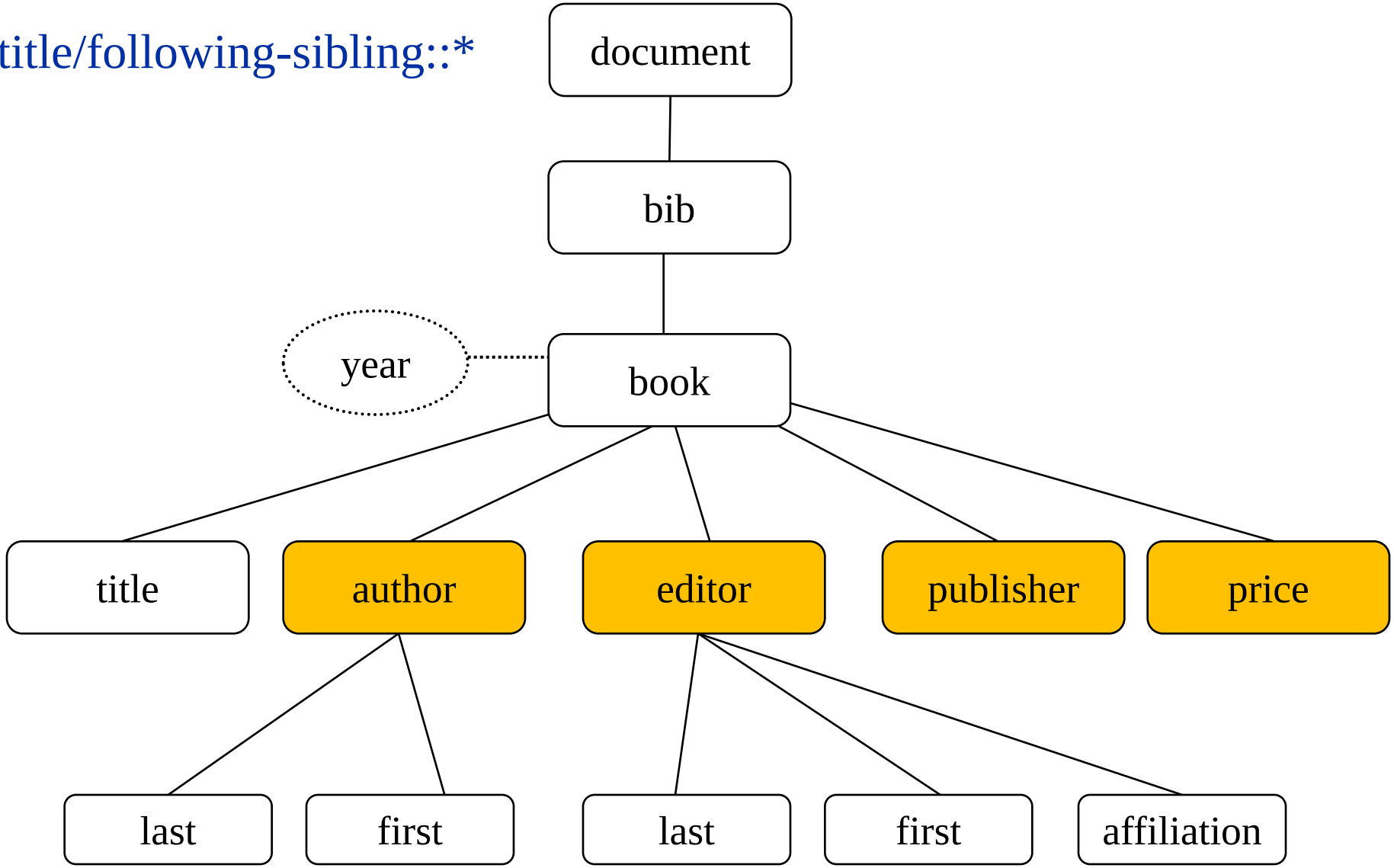
Exemples

- **`//editor/parent::*`** renvoie le parent du nœud **`editor`**, càd **`book`**
- **`/book/editor/descendant::*`** renvoie les descendants de **`editor`**, càd **`last, first, affiliation`**.
- **`//last/ancestor::*`** renvoie tous les ancêtres de **`last`**, càd **`author, editor, book, bib`**
- **`//editor/following-sibling::*`** renvoie les nœuds **`publisher, price`**
- **`//editor/preceding-sibling::*`** renvoie les nœuds **`author, title`**
- **`//author/following::*`** renvoie les nœuds **`editor, last, first, affiliation, publisher, price`**
- **`//author/preceding::*`** renvoie le nœud **`title`**

//publisher/ancestor::*



//title/following-sibling::*



Remarque

```
<X>
   <A>
      <B>
         <C/>
         <C/>
      </B>
      <D>
         <C/>
         <C/>
         <C/>
      </D>
   </A>
</X>
```

`//C[3]`

3eme élément C d'un élément fils de la racine.

Équivalent à

`/descendant-or-self::node()/child::C[3]`

`/descendant::C[3]`

Troisième élément C du document

Exercice

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
<!-- a bookstore database -->
<book isbn=“111111” cat=“fiction”>
<!-- a particular book -->
<title lang=“chn”>Harry Potter</title>
<price unit=“us”>79.99</price>
</book>
<book isbn=“222222” cat=“textbook”>
<title lang=“eng”>Learning XML</title>
<price unit=“us”>69.95</price>
</book>
<book isbn "333333" cat "textbook">
<title lang="eng">Intro. to Databases</title>
<price unit="usd">39.00</price>
</book>
</bookstore>
```

Racine du document :

Élément *bookstore*:

Eléments *book*:

Eléments *price* :

Attributs *lang* :

./../.

/bookstore//@lang/../..

/bookstore/book/text()

Exercice : Axes

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
<!-- a bookstore database -->
<book isbn=“111111” cat=“fiction”>
<!-- a particular book -->
<title lang=“chn”>Harry Potter</title>
<price unit=“us”>79.99</price>
</book>
<book isbn=“222222” cat=“textbook”>
<title lang=“eng”>Learning XML</title>
<price unit=“us”>69.95</price>
</book>
<book isbn "333333" cat "textbook">
<title lang="eng">Intro. to Databases</title>
<price unit="usd">39.00</price>
</book>
</bookstore>
```

Éléments *book*

Attributs *isbn*

/child::book

/bookstore/book/following-sibling::book

/bookstore/node()/descendant-or-self::node()

Exemple : valeur textuelle

```
<?xml version="1.0" encoding="utf-8"?>
<A>
   <B att1='1'>
        <D>Text 1</D>
        <D>Text 2</D>
   </B>
   <B att1='2'>
        <D>Text 3</D>
   </B>
   <C att2='a' att3='b'/>
</A>
```

Que renvoient :

//B[1]//text()

//B//text()[1]

//B[1]//text() [1]

//B/D/text()[1]

Remarque : la fonction text() a différentes implémentations selon les outils.

XPath (abréviations)

`child::` est l'axe par défaut, et peut être omis

`/child::book` est équivalent à **`/book`**

`child::book/child::title` peut s'écrire **`book/title`**

`attribute::` peut être remplacé par **`@`**

`child::book[attribute::year= "2002"]` peut s'écrire
`book[@year= "2002"]`

`//` est l'abréviation de **`/descendant-or-self::node()/`**

`.` est l'abréviation de **`self::node()`**

`..` est l'abréviation de **`parent::node()`**