

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

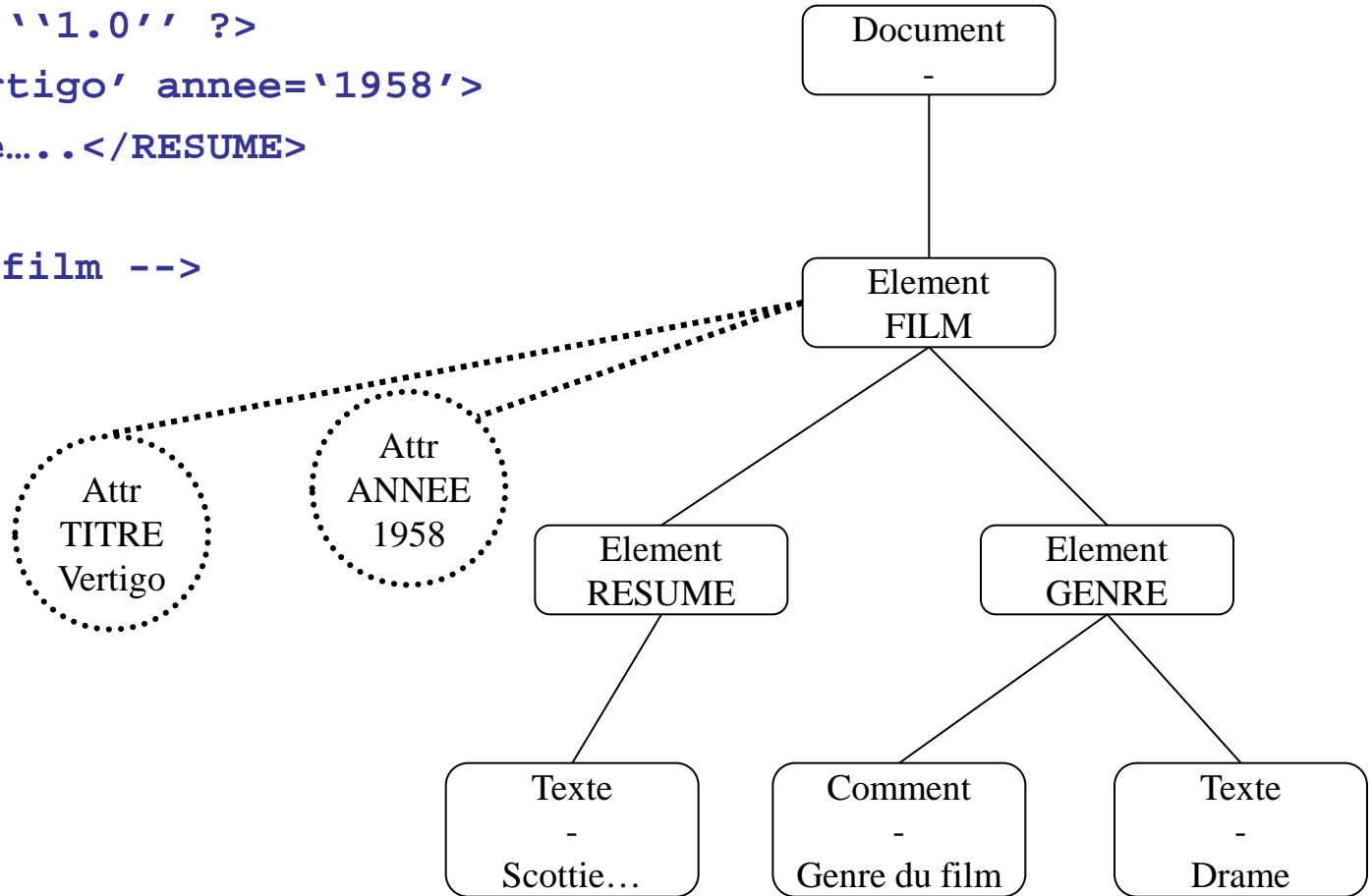
- Arbres XML
  - Définition
  - Ordre du document
  - Sérialisation
- Langage Xpath
  - Expression de chemin
  - Evaluation
  - Prédicats
  - Attributs
  - Axes

# 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  $T_1, T_2, \dots, T_n$
  - retourne un ou plusieurs noeuds  $T_n$ , tels qu'il existe des arcs  $T_1 \rightarrow T_2, \dots, T_{n-1} \rightarrow T_n$
- 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 '/' :

[ / ]étape<sub>1</sub>/étape<sub>2</sub>/.../étape<sub>n</sub>

- Une étape a la forme suivante :

**Axe::filtre[prédictat<sub>1</sub>][prédictat<sub>2</sub>]**

- 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) *prédictat(s)* expriment des propriétés que doivent satisfaire les nœuds retenus.



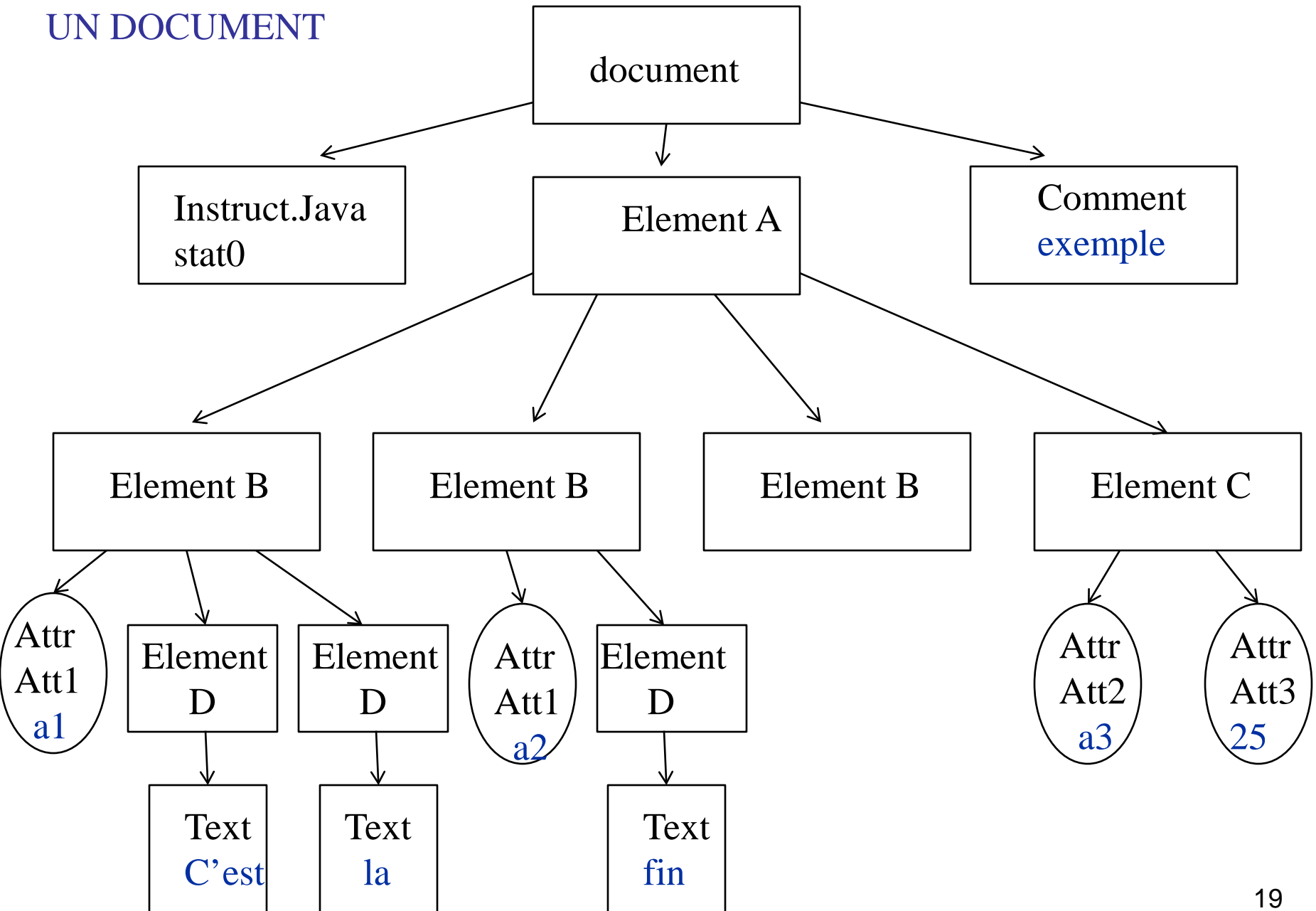
# Evaluation d'une expression

- 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



# Le même sous forme sérielle

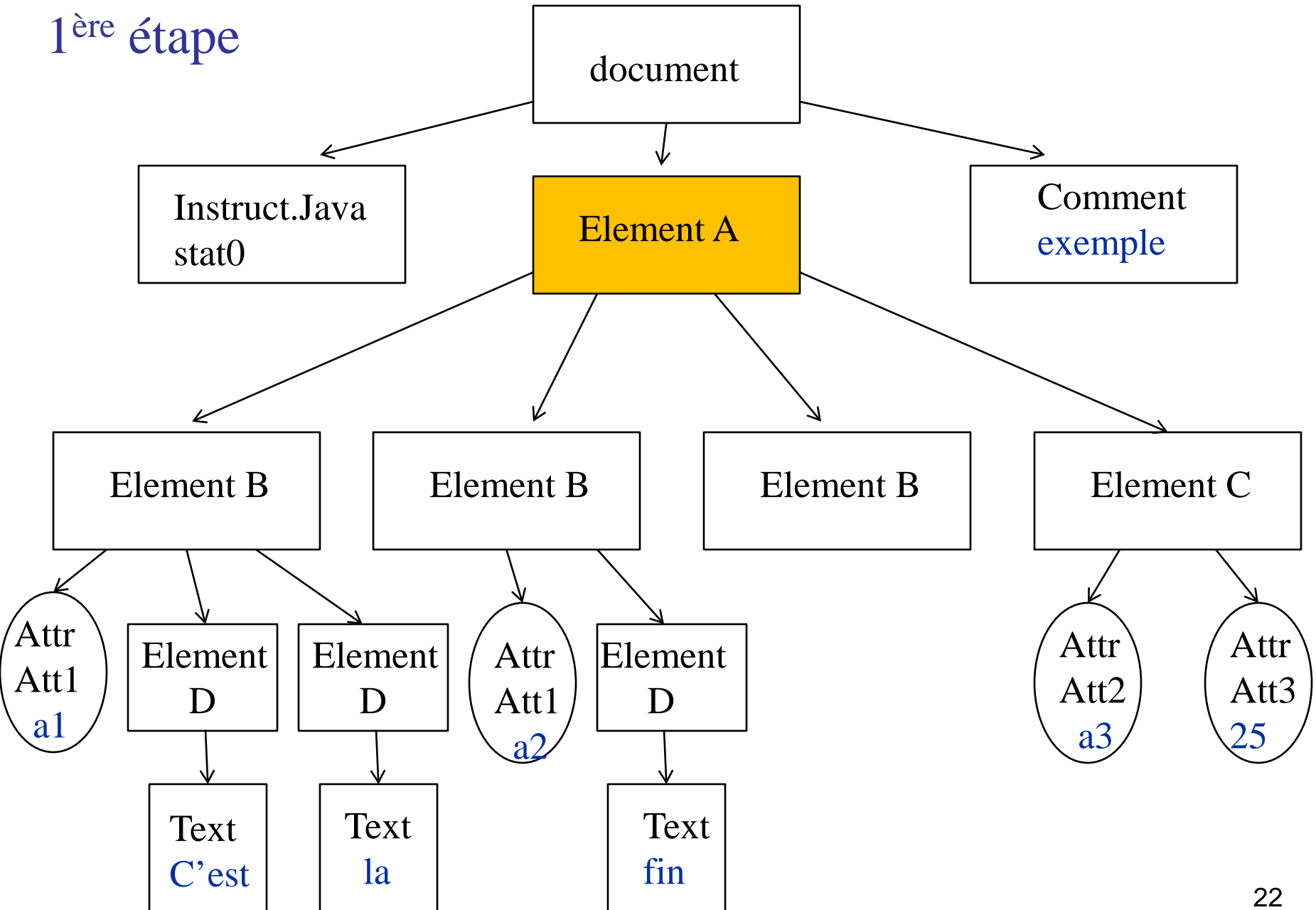
```
<?xml version= '1.0' encoding='ISO-8859-1'?>
<?java stat0?>
<A>
  <B att1='a1'>
    <D>C'est</D>
    <D>la</D>
  </B>
  <B att1='a2'>
    <D>fin</D>
  </B>
  <B/>
  <C att2='a3' att3='25' />
</A>
<!-- exemple-->
```

# Exemple d'évaluation

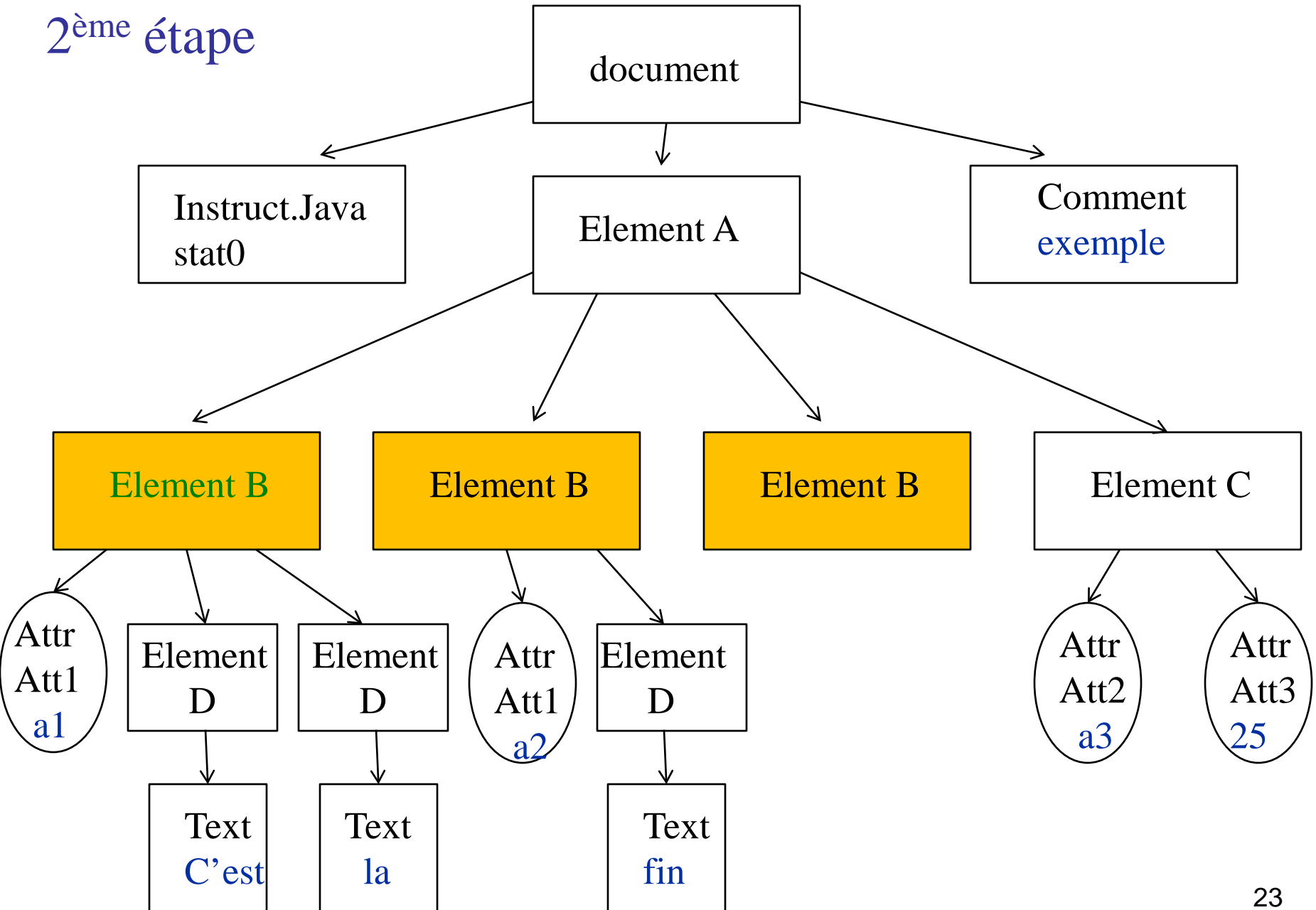
## /A/B/@att1

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

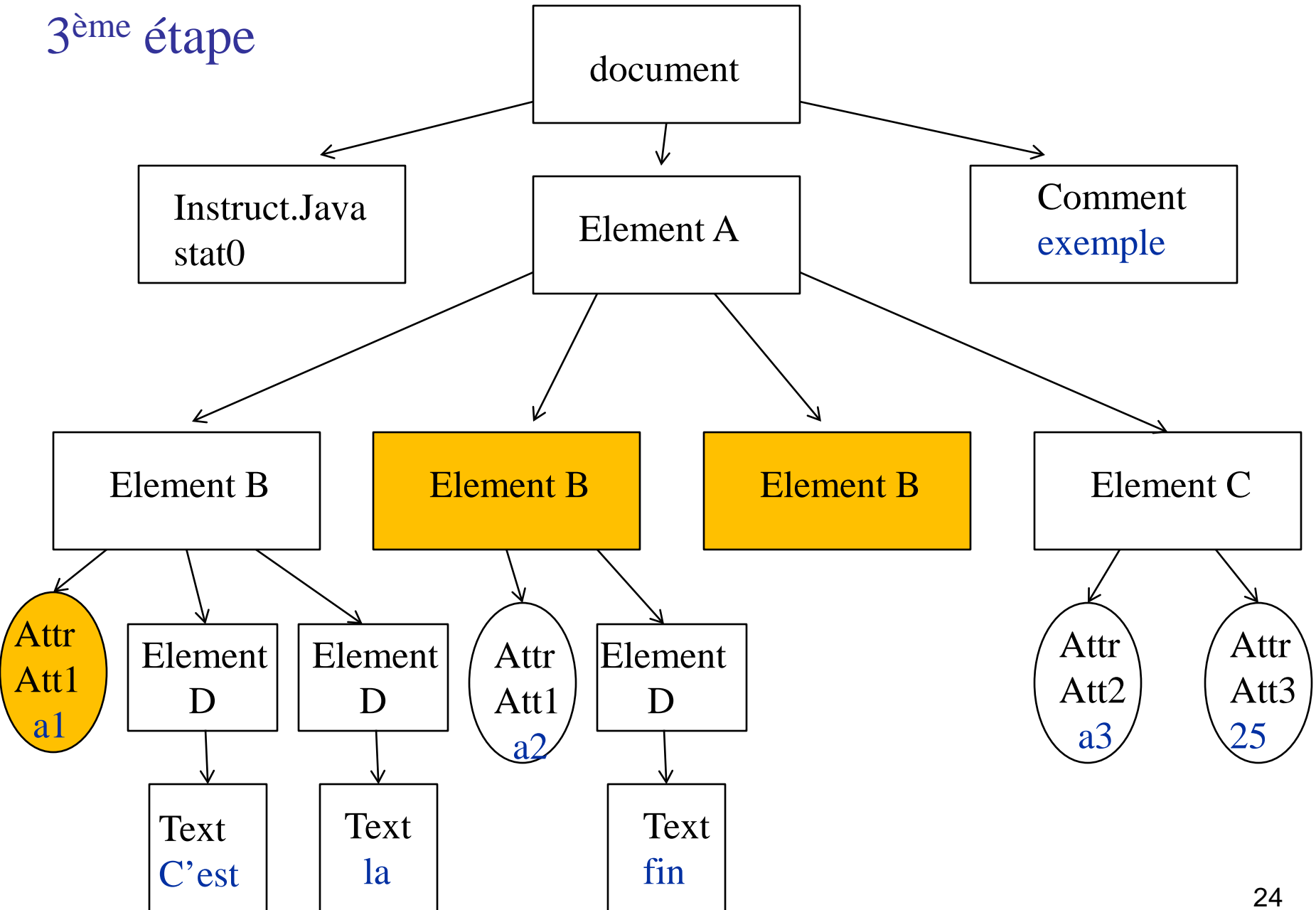
# 1<sup>ère</sup> étape



## 2<sup>ème</sup> étape

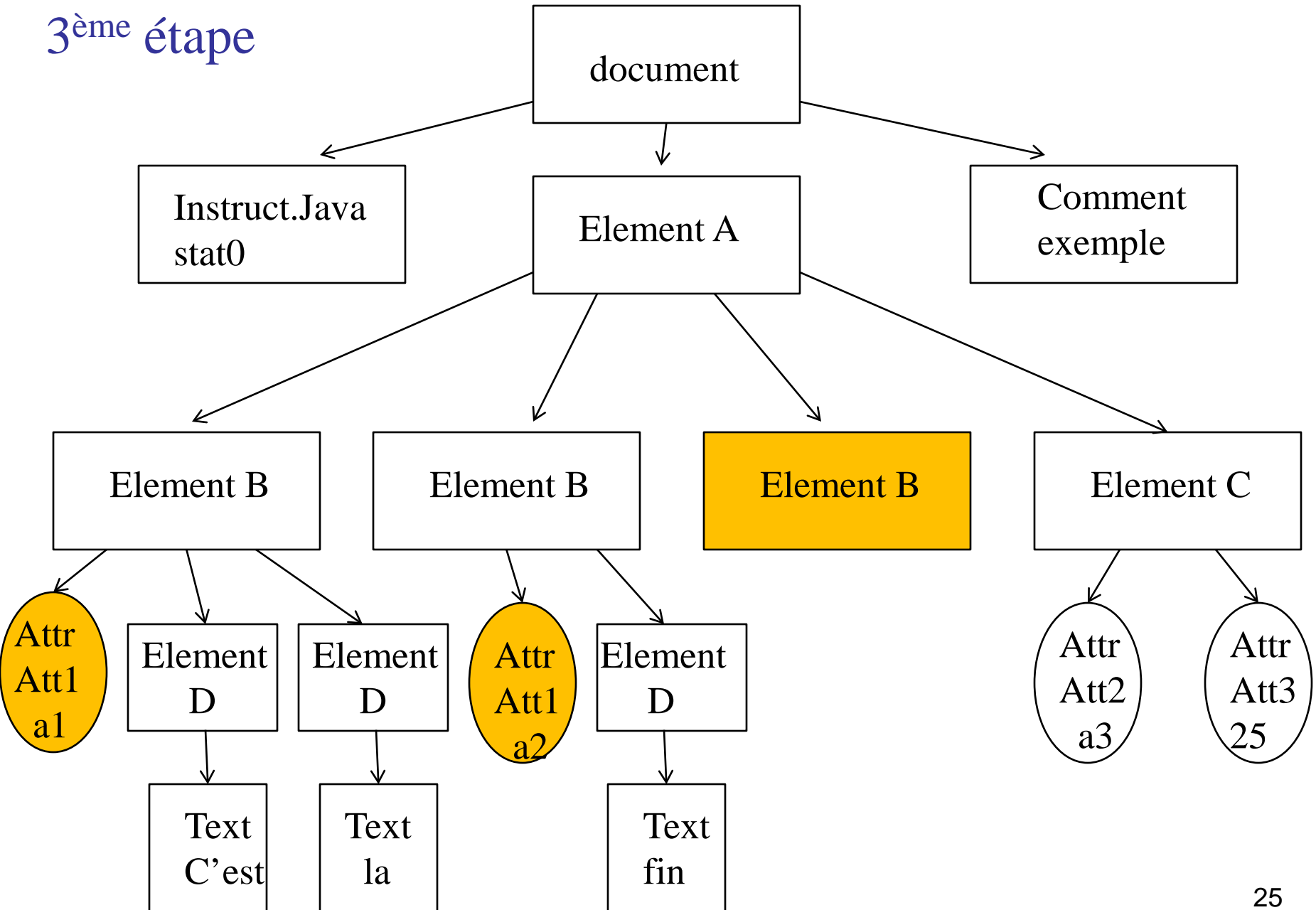


### 3<sup>ème</sup> étape

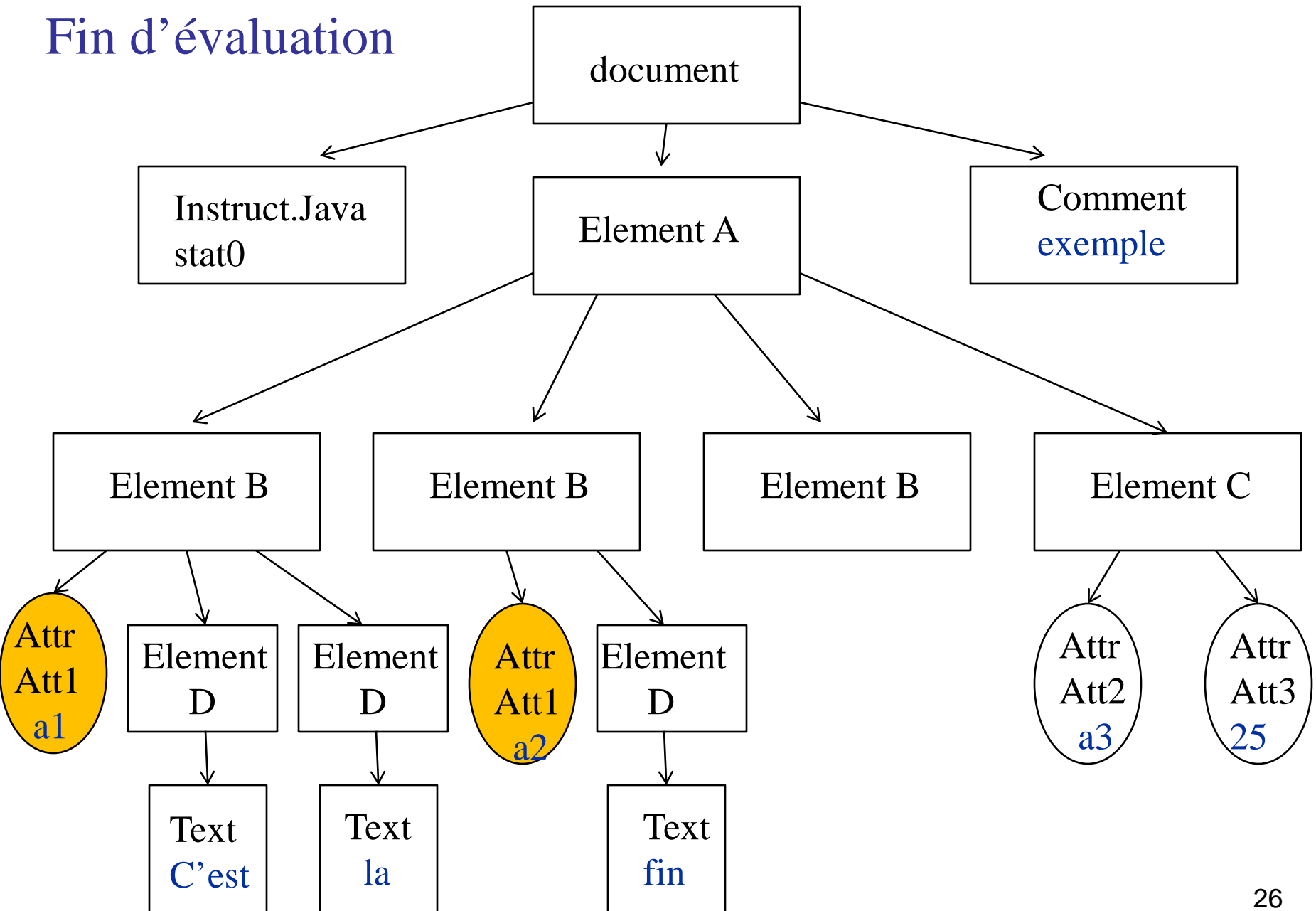




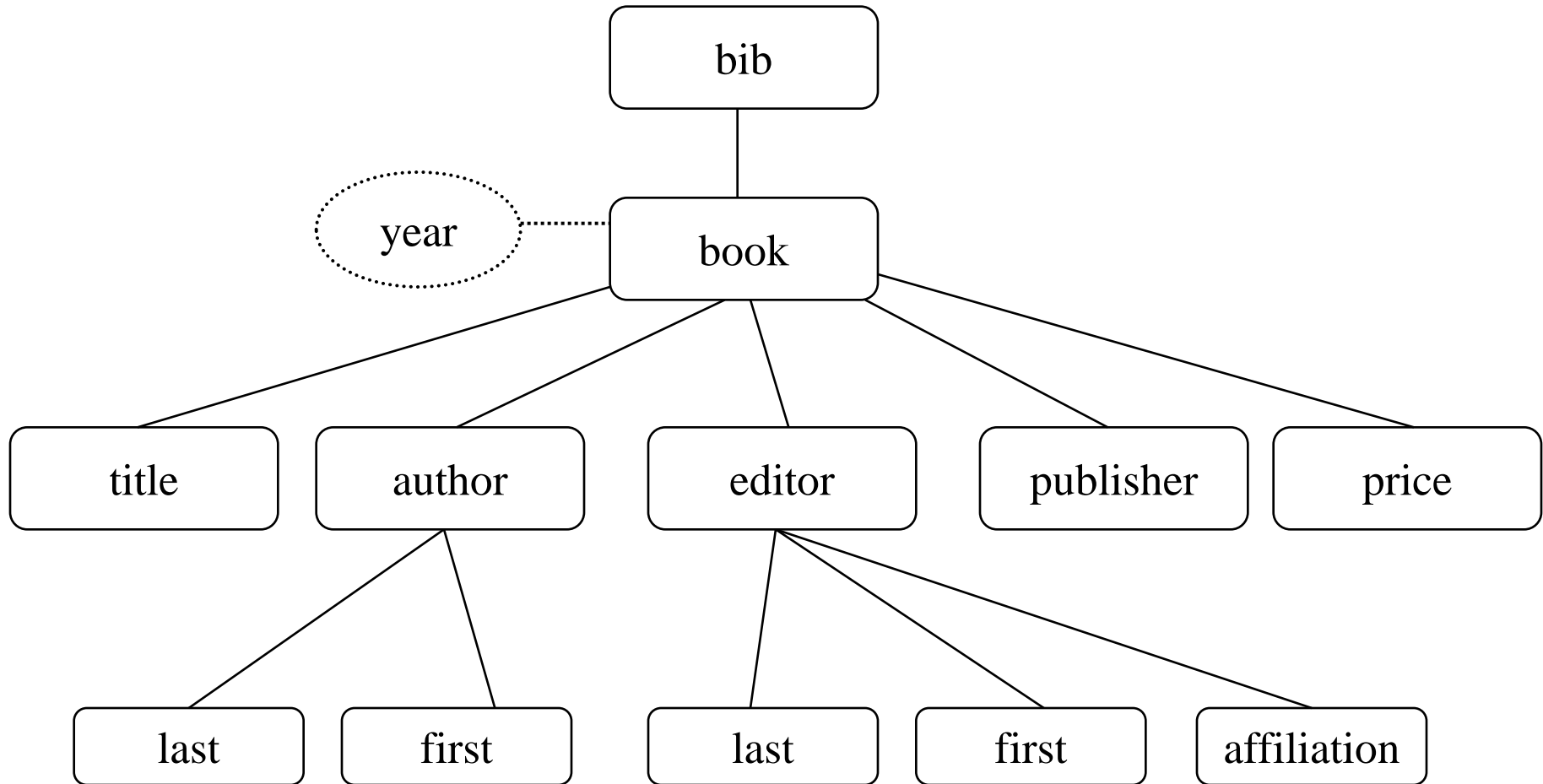
### 3<sup>ème</sup> étape



## Fin d'évaluation



# Un autre exemple



# DTD Example

[www.bn.com/bib.xml](http://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`

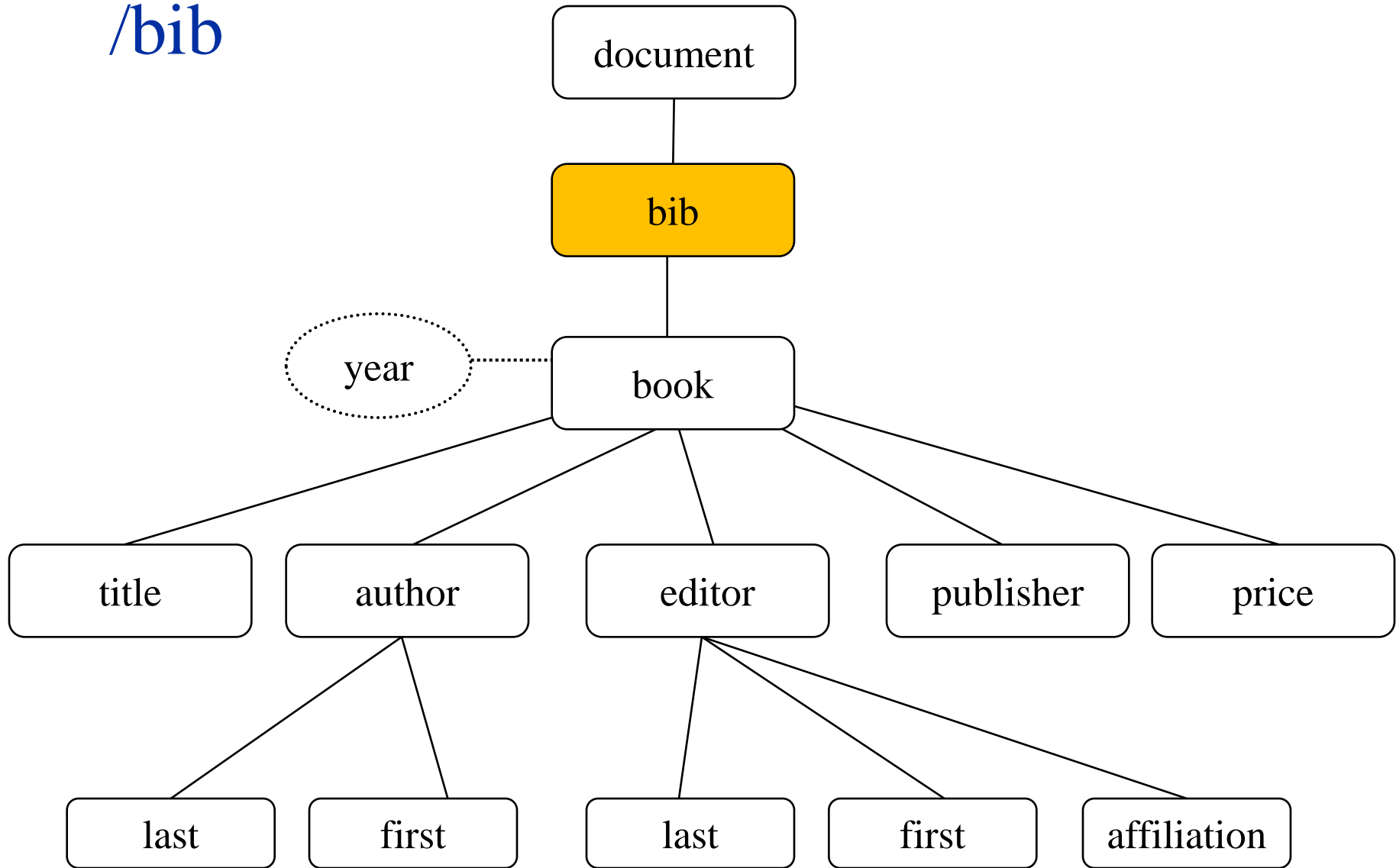
Un chemin qui commence par //A sélectionne tous les chemins qui ont pour origine le nœud contexte et destination un nœud A quels que soient les nœuds intermédiaires éventuels..

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

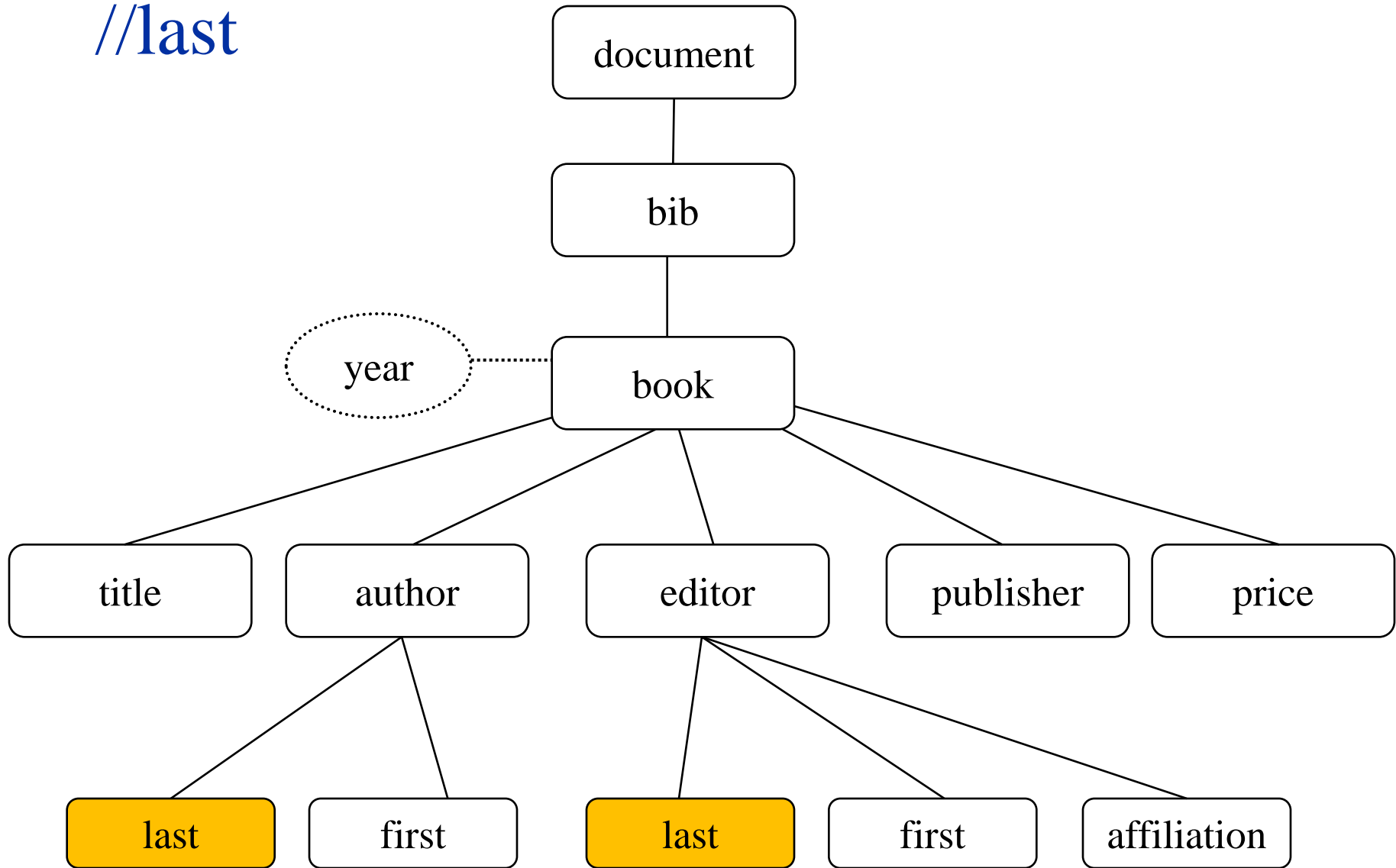
L'étoile \* sélectionne tous les nœuds fils du nœud qui la précède.

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

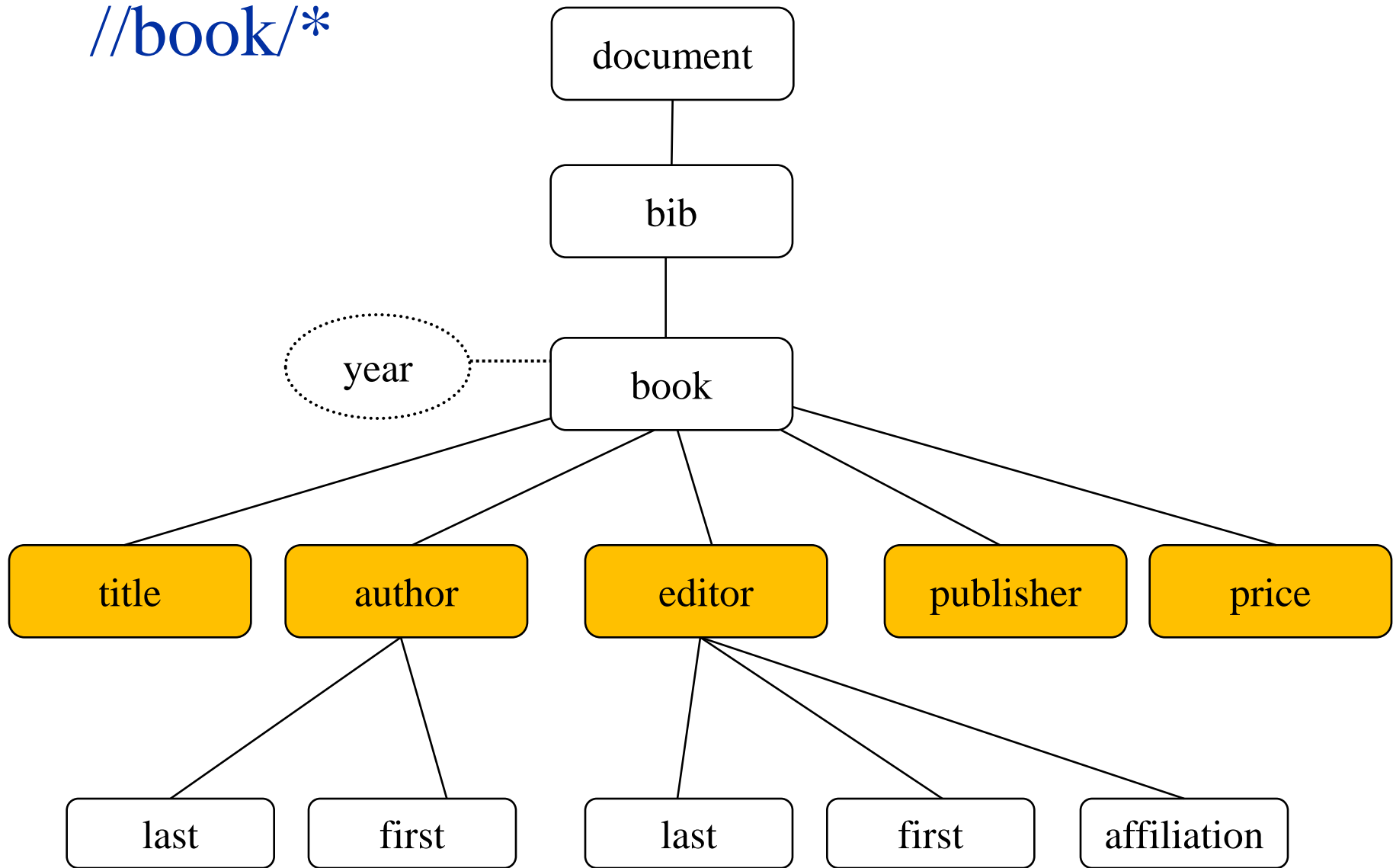
/bib



//last



//book/\*





# Example

```
<bib>
  <book>
    <title/>
    <editor>
      <last/>
      <first/>
      <affiliation/>
    </editor>
    <publisher/>
    <price/>
  </book>
</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 : **attribute::a1** raccourci pour **@a1**
  - Le filtre : le type de nœud était un élément (par défaut)
  - Il n'y avait pas de prédicat

# 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 //
- Ancêtres
  - direct parent ..
  - indirect ancestor
- Frères
  - même niveau following- /preceding-sibling
  - Successeurs/prédécesseurs following/preceding
- Autres
  - ancestor-or-self, descendant-or-self, 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 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

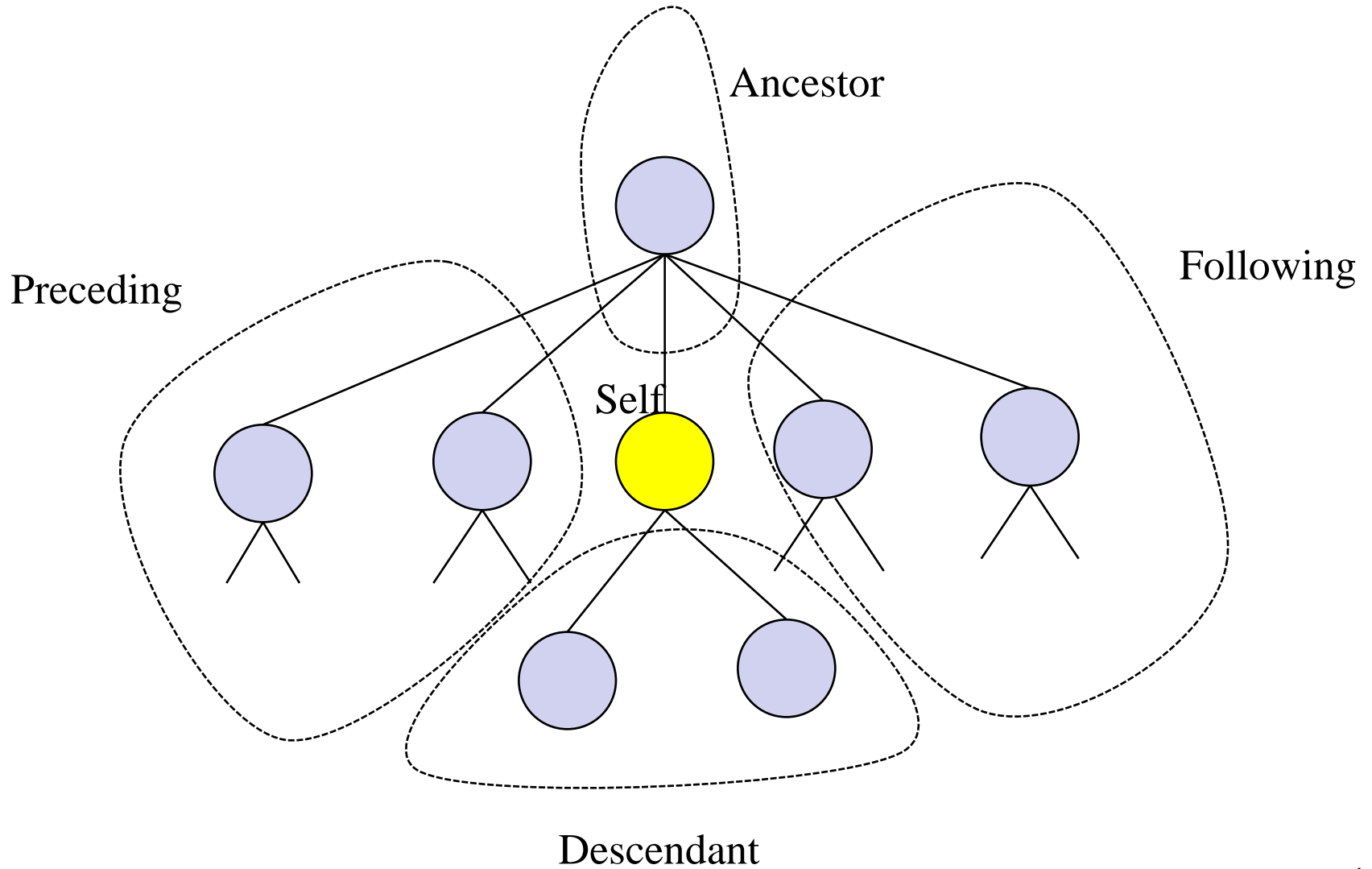
- L'axe **descendant-or-self** contient le nœud contextuel et ses descendants
- L'axe **ancestor-or-self** contient le nœud contextuel et ses ancêtres; ainsi l'axe **ancestor-or-self** contient toujours le nœud racine
- Les axes **ancestors**, **descendants**, **following**, **preceding** et **self** 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.

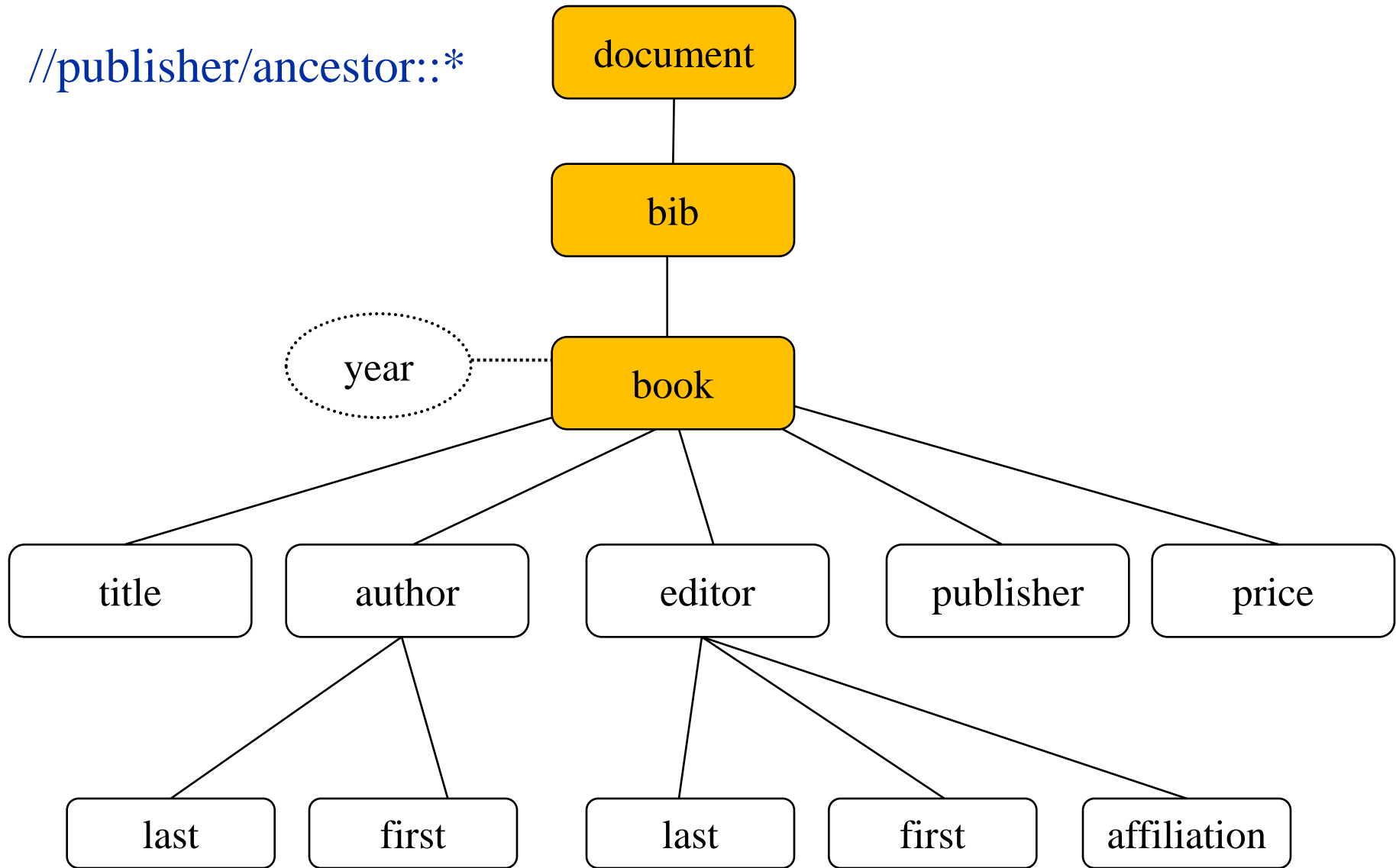
# Axes



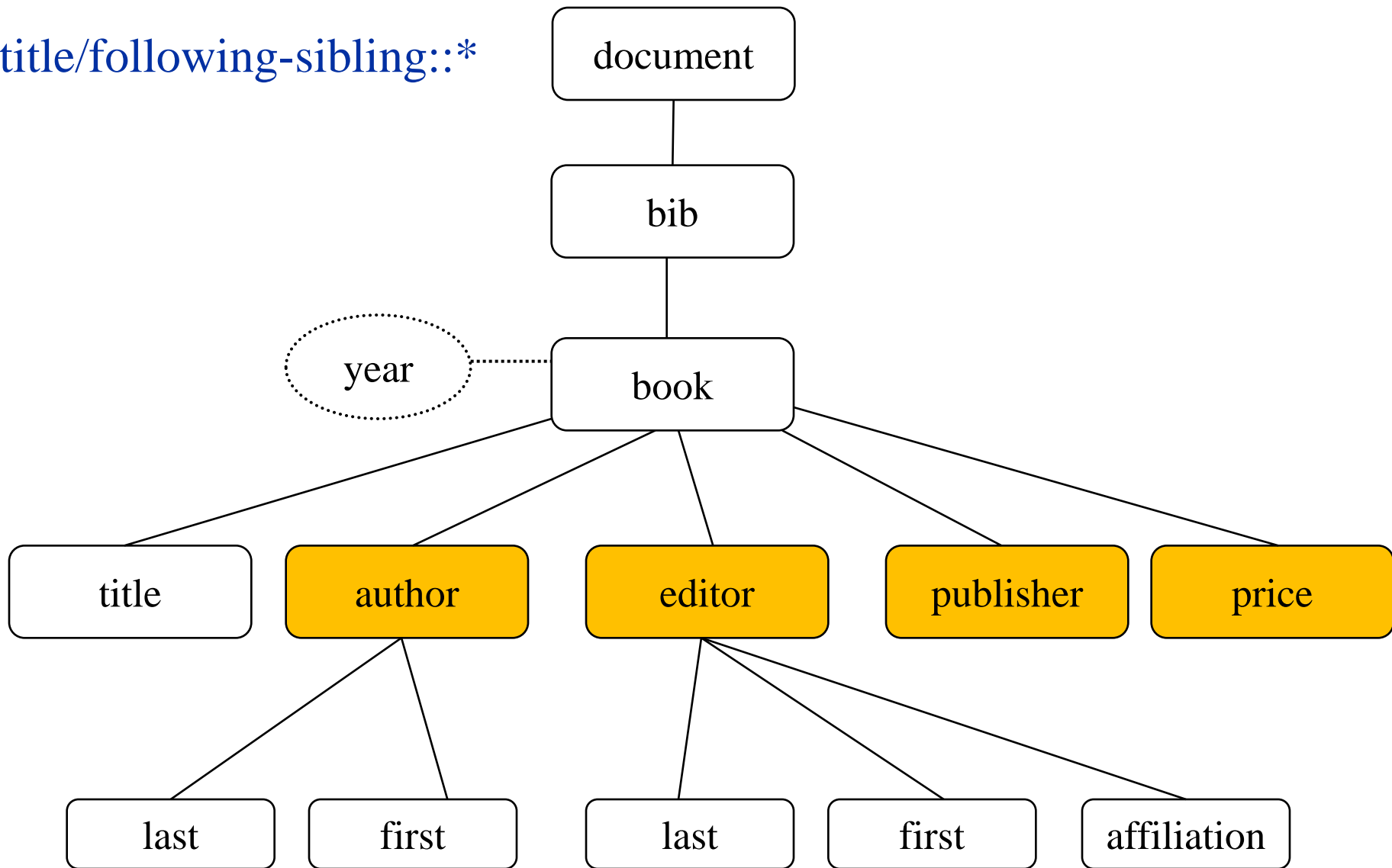
# Exemples

- `//editor/parent::*` renvoie les parents du nœud **editor**, càd **book** et **bib**
- `/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::\*



# 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*:

Éléments *book*:

Éléments *price* :

Attributs *lang* :

././.

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

/bookstore/book/text()



# Exercice (solution)

```
<?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 : / ou /.

Élément bookstore: /bookstore ou  
./bookstore

Éléments book: /bookstore/book ou  
//book

Éléments price : /bookstore/book/price

Attributs lang: // @lang

././ : racine

/bookstore// @lang/../../ éléments book

/bookstore/book/text() : ensemble vide

# 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()

# Exercice : Axes (solution)

```
<?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*

/descendant::book

Attributs *isbn*

//book/attribute::isbn

/child::book

ensemble vide

/bookstore/book/following-sibling::book

2eme élément book

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

tous les nœuds (sauf attributs)

descendants de bookstore

# 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()**