

TECHNOLOGIES
XML
EXTENSIBLE MARKUP LANGUAGE
PARTIE III

2

XPATH

XPATH

Langage de requêtes XPATH

4

- ❓ XPATH est un langage simple de requête qui permet de localiser des parties d'un document XML (identifier un ensemble de noeuds dans le document)
 - modélise un document XML comme un arbre de noeuds
 - utilisé par **XSLT** pour faire de la transformation de documents XML
- ❓ Adopte une syntaxe non-XML.

Versions

5

- XPath 1.0 (recommandation W3C depuis 1999 et utilisée dans XSLT 1.0)
- XPath 1.0 est inclus dans XPath 2.0,
- XPath 2.0 est inclus dans Xquery
- XPath 3.0 (18 December 2014)
- XPath 3.1 (21 Mars 2017)

Expression XPATH

6

Une expression XPath est un chemin de localisation, constitué d'étapes (séparés par /):

- Une étape est composée de trois parties:

Axe::Test_de_Noëud [predicat]

❓ axe de déplacement (fils, frères, ancêtres...)

❓ test de de nœud: pour designer le(s) nœud(s) concerné(s) selon l'axe.

❓ Prédicats: des conditions à respecter (expression booléenne à vérifier sur chaque nœud trouvé). [optionnel]

- Chaque étape renvoie un ensemble de nœuds et pour chacun d'entre eux on y applique le même processus.

Expression XPATH

7

- Un chemin peut être:
 - absolu
 - commence par la racine
 - /étape1/.../étapeN
 - relatif
 - commence à un nœud courant
 - étape1/.../étapeN
- Les types de nœuds utilisés par Xpath sont: racine, élément, texte, attribut, espace de noms, instruction de traitement et commentaire

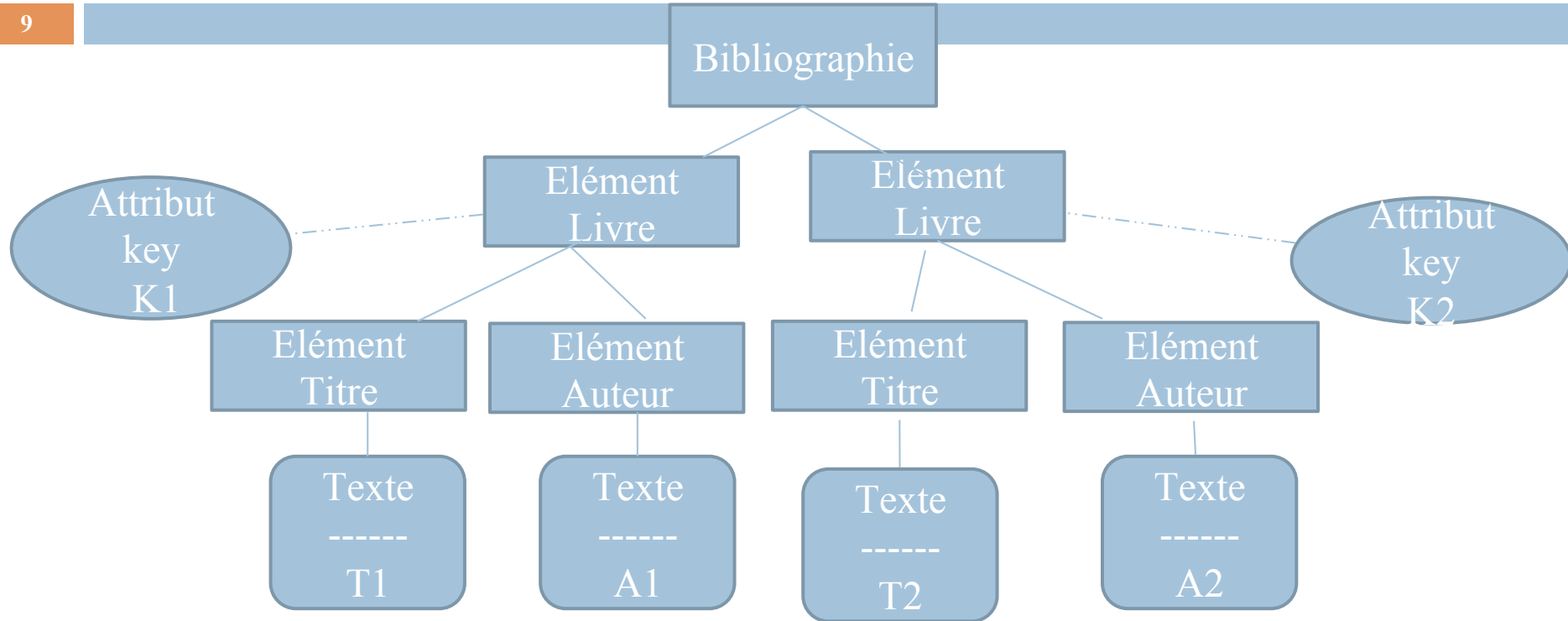
Exemple

8

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Ceci est un exemple -->
<bibliographie >
  <livre key= " K1" >
    <titre>T1</titre>
    <auteur> A1</auteur>
  </livre>
  <livre key= "K2" >
    <titre>D2</titre>
    <auteur>A2</auteur>
  </livre>
</bibliographie>
```


Exemple: /Bibliographie/Livre[@key= "K2"]

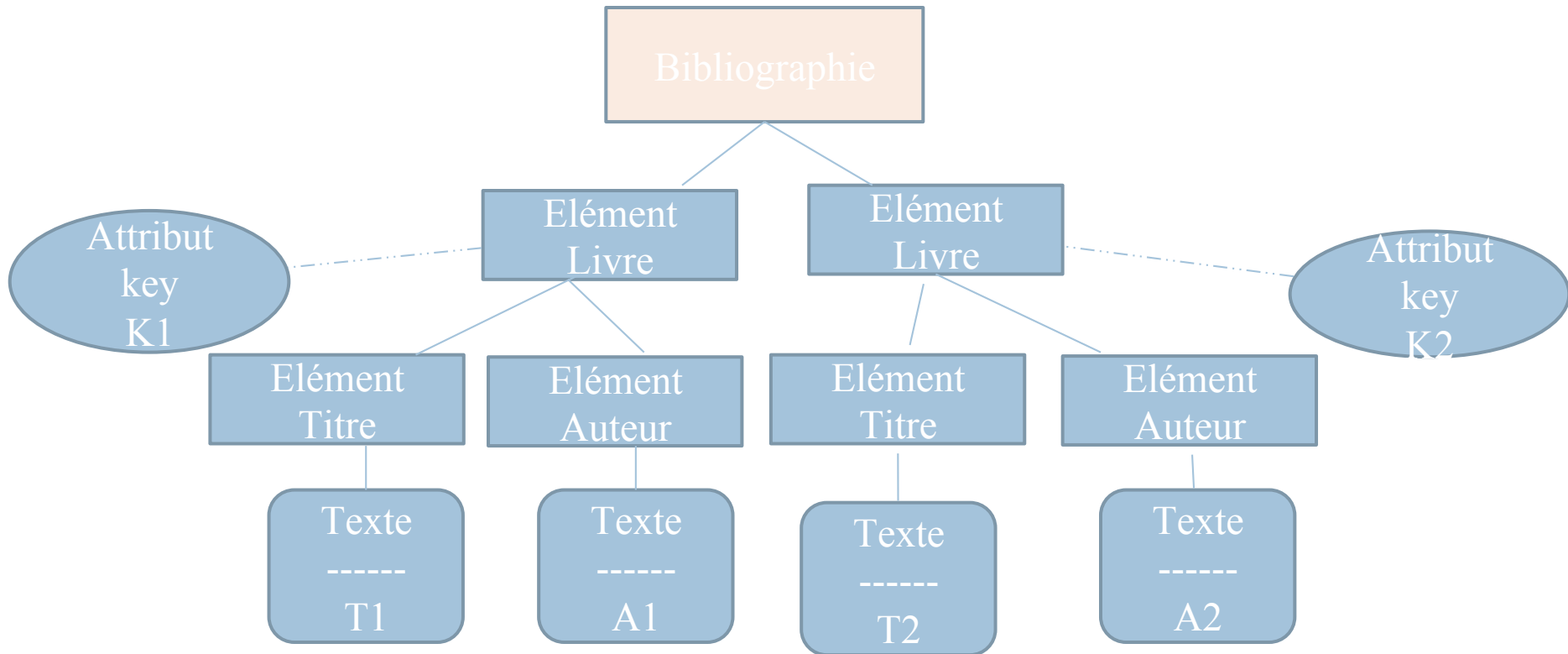
9



- Identifier tout élément Livre enfant du nœud Bibliographie possédant un attribut = K2

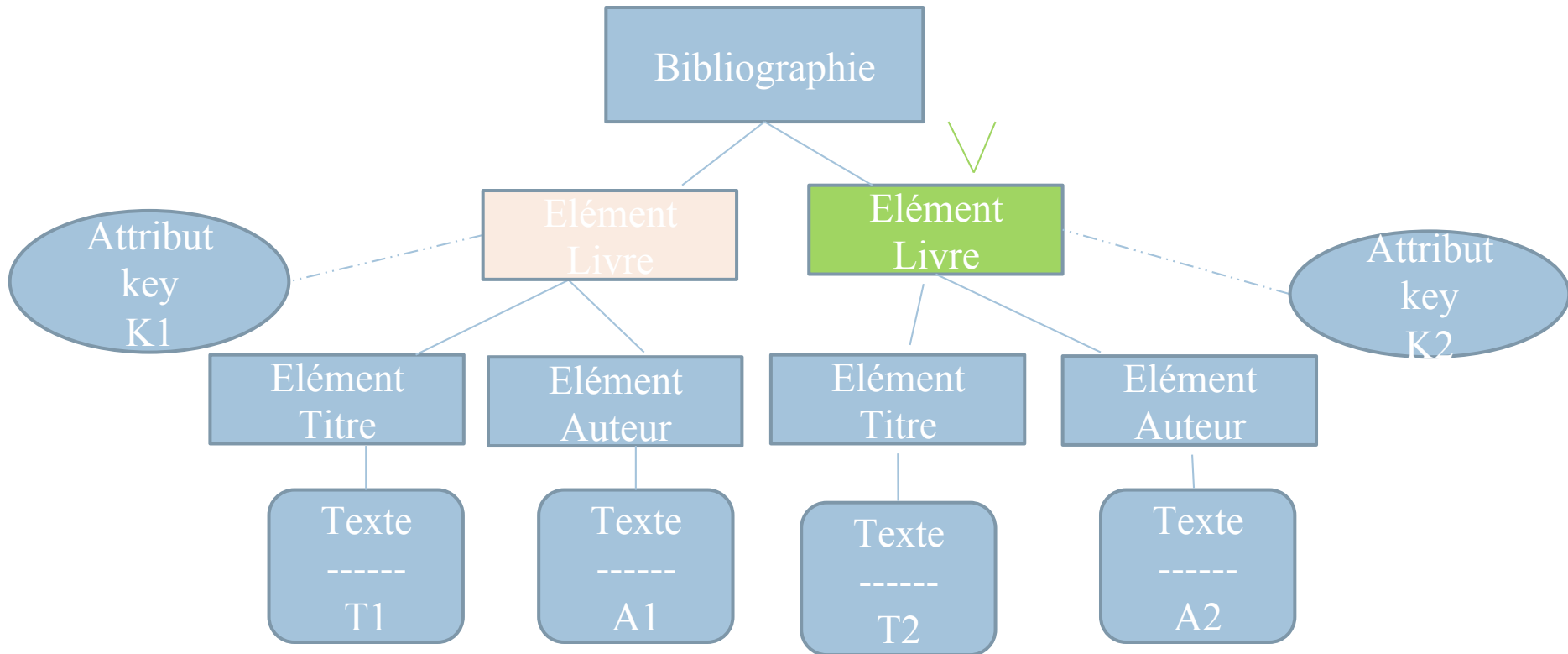
/Bibliographie/Livre[@key= "K2"] : 1ère étape

10



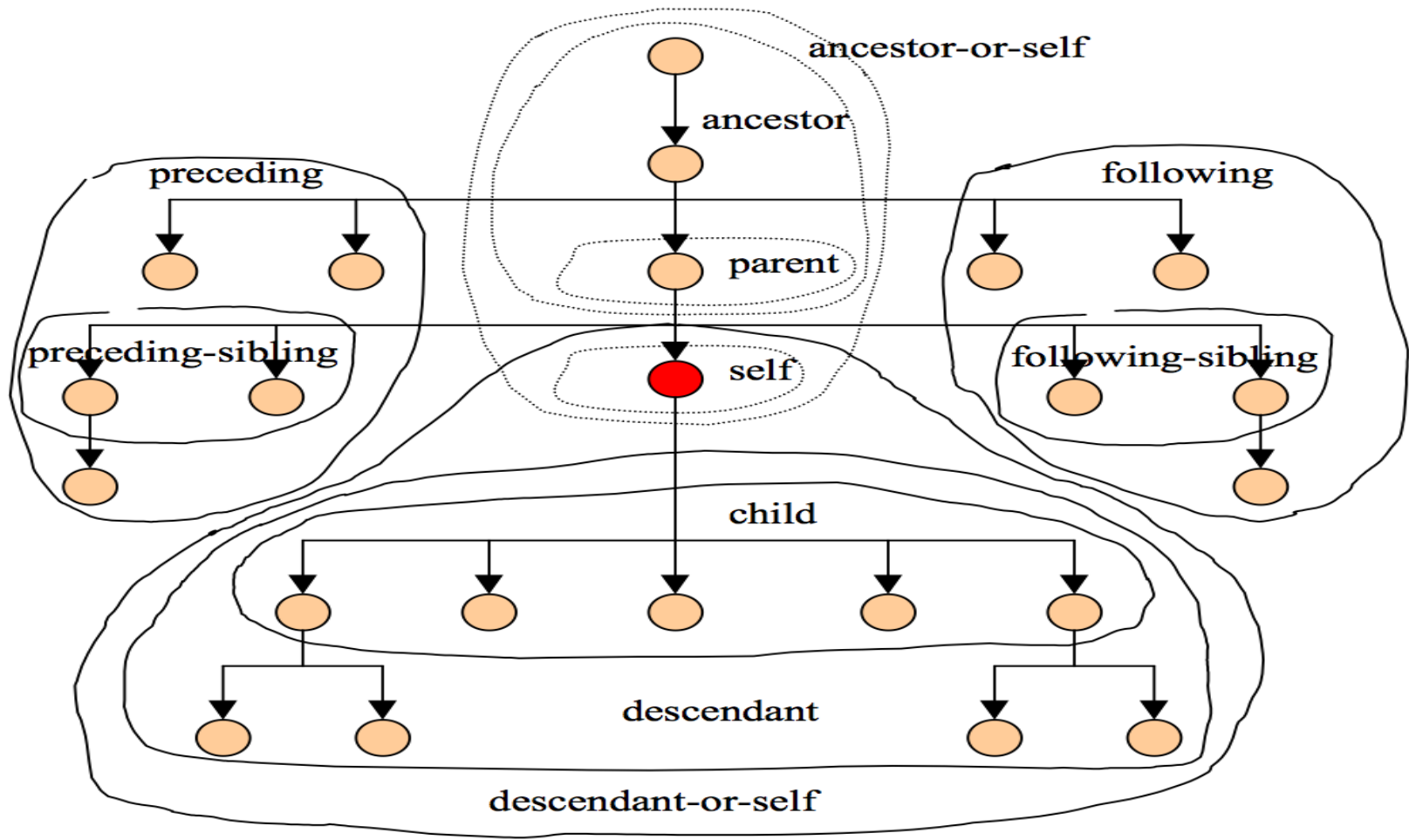
/Bibliographie/Livre[@key= "K2"] : 2ème étape

11



Axe de déplacement

12



C'est la direction dans laquelle on se dirige à partir du nœud courant (vers le père, vers les fils..). Pour plus d'information voir : <http://www.w3.org/TR/xpath/#axes>

Axe de déplacement (Suite)

13

Nom de l'axe	Description
ancestor	ancêtres du nœud courant
ancestor-or-self	le nœud courant et ses ancêtres
attribute	attributs du nœud courant
child	enfants du nœud courant
descendant	descendants du nœud courant
descendant-or-self	nœud courant et ses descendants
following	nœuds suivant le nœud courant
following-sibling	frères suivants du nœud courant
parent	père du nœud courant
preceding	nœuds précédant le nœud courant
preceding-sibling	frères précédents du nœud courant
self	nœud courant

- Filtrer les nœuds
 - ❑ nom: les nœuds de l'axe qui portent ce nom
 - ❑ *: les nœuds de type Element
- Filtrer les nœuds textuels
 - ❑ text(): tous les nœuds de type Text de l'axe
- Filtrer les commentaires
 - ❑ comment(): tous les nœuds de type Comment de l'axe
- Filtrer les instructions de traitement
 - ❑ processing-instruction(): tous les nœuds de type instruction de traitement de l'axe
- Filtrer les nœuds
 - ❑ node(): tous les types de nœud (éléments, commentaires..) de l'axe

Abréviations

15

- `child::nom<=> nom`
 - `child::div/child::para -> div/para` (para fils du div)
- `attribute:: <=> @`
 - `child::para[attribute::type="warning"] -> para[@type="warning"]`
- `/descendant-or-self::node() <=> //`
 - `/descendant-or-self::node()/child::para -> //para`
(select tous les para descendant du root)
- `self::node() <=> .`
 - `self::node()/descendant-or-self::node()/child::para -> ./para`
(select para descendant du noeud contextuel)
- `parent::node() <=> ..`
 - `parent::node()/child::title -> ../title`
(select titre fils du parent du neoud contextuel)

Pour plus d'information voir :

<https://www.w3.org/TR/1999/REC-xpath-19991116/>

XPath: Expression

16

Expression	Description	Exemple
/	Sélectionne la racine	
/A/B	Sélectionne tous les éléments B fils de l'élément A	/Bibliographie/livre
//A	Sélectionne tous les éléments A	//livre
	Permet de combiner les requêtes	A B
*	Sélectionne tous les enfants du nœud contextuel	/Bibliographie/*
A/@*	Sélectionne tous les attributs de l'élément A	Livre/@*

Prédicat

17

- Un prédicat est une condition qui permet de raffiner la sélection sur des éléments obtenus en effectuant un test sur le nom d'un élément, son contenu ses attributs et sa position dans l'arbre.

Prédicat

18

Expression	Description	Exemple
[i]	i ième élément de la sélection	/livre[2]
[last()]	Identifie le dernier élément de la sélection	//livre[last()]
A[B]	Localise tout élément A enfant du noeud contextuel et ayant un élément enfant B	auteur[annee]
A[@att]	Identifie tout élément A enfant du noeud contextuel et possédant un attribut att	annee[@edition]
A[B='val']	Localise tout élément A enfant du noeud contextuel et ayant un élément enfant B possédant comme valeur ' val '	auteur[annee='1956']
A[.='val']	Localise tout élément A enfant du noeud contextuel et ayant comme valeur ' val '	livre[.='XML']
A[@att='val']	Condition sur les attributs	livre[@ISBN="xxx x"]

Chemin de localisation: Syntaxe

19

Axe::Test_de_Noeud [predicat]

- Exemples:
 - `child::personne[child::nom= 'xxx ']` →
éléments fils personne (selon le nœud de départ) dont le fils nom est xxx
 - `child::personne[attribute::numero= '123 ']` →
éléments fils personne dont l'attribut numero a 123 comme valeur
 - `child::test[contains(text(),'coucou') and position()=2]` →
le deuxième fils test si celui-ci contient le texte *coucou*

Quelques fonctions Xpath

20

- `last()`: Identifie le dernier élément de la sélection
`? //livre[last()]`
- `position()`: retourne l'indice de la position contextuelle (context position) d'un élément par rapport à son parent.
`? /livre[position()=1]`
- `contains(chaine1, chaine2)` teste si chaine1 contient chaine2
`? //*[contains(name(), 'x')]: éléments dont le nom contient 'x'`
- `name()` renvoie le nom du nœud contexte
`? //*[name()='livre']: éléments dont le nom est 'livre'`

Quelques fonctions Xpath


21

 `not(expression)` permet d'exprimer la négation

- `//XX[not(@*)]>` : éléments XX qui n'ont pas d'attribut

 `string-length(string)`: retourne la longueur d'un string

- `//*[string-length(name()) = 5]` : les éléments dont le nombre de caractère du nom = 5

 `starts-with(string1,string2)`: vrai si la première chaîne commence par la deuxième

- `//*[starts-with(name(), 'x')]` : éléments dont le nom commence par 'x'

 `count(ensemble de noeuds)`: Permet de tester le nombre de nœuds

- `//livre[count(chapitre)=4]`

Pour plus d'information voir: <https://www.w3.org/TR/1999/REC-xpath-19991116/>

Exemples

Recherche par Axe ::Test_de_Noead

22

- `/descendant::*` :

tout les éléments descendants de la racine

- `/xx/yy/descendant::*` :

Sélectionne tous les descendants de yy (fils de xx)

- `//xx/descendant::*` :

Sélectionne tous les éléments qui ont xx comme ancêtre

- `/*/*/*/*xx` :

Sélectionne tous les éléments xx qui ont trois ancêtres

EXTENSIBLE STYLESHEET LANGUAGE XSL

- C'est une recommandation du W3C qui a vu le jour en novembre 1999.
- conçu pour transformer des documents XML en d'autres formats comme PDF ou des pages HTML
- un langage de réécriture pour XML subdivisé en:
 - ❑ XSLT (Transformation) : permet de produire un document XML ou texte à partir d'un autre document par application de règles de transformation
 - Utilise XPath pour la localisation des noeuds
 - ❑ % XSL-FO (Formatting Object): permet de transformer un document XML en un fichier directement affichable ou imprimable (p.ex. PDF)

XSLT

25

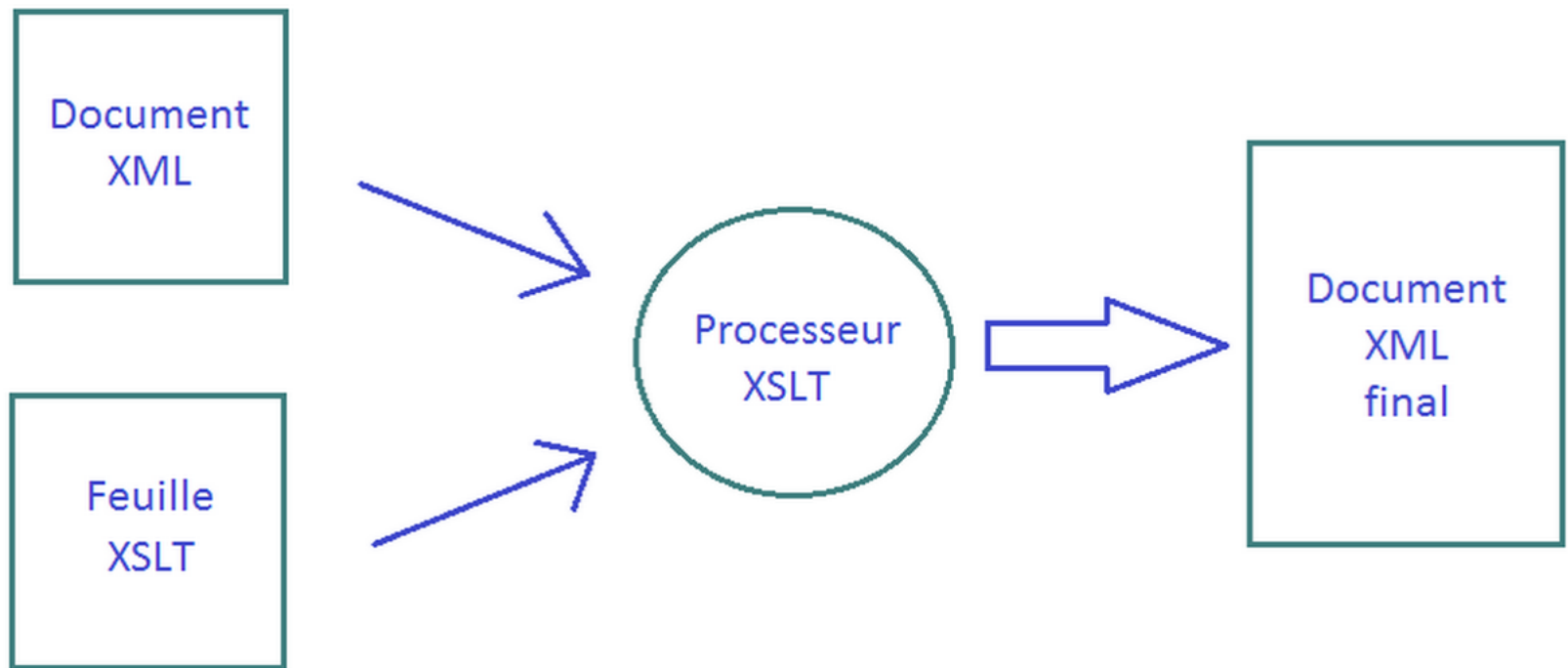
- Un langage de description de transformation à opérer sur un arbre XML avec une syntaxe XML.
- Modèle de calcul
 - ❑ Utilise une technique de *filtrage* à base de *motifs* (patterns) *et de modèles* (template) décrits dans des *règles* (template rules) pour transformer des arbres

Représentent les informations à repérer dans l'arbre qui correspond à XML

Correspondent aux traitements à appliquer aux patterns

Principe de base

26



Processeur XSLT

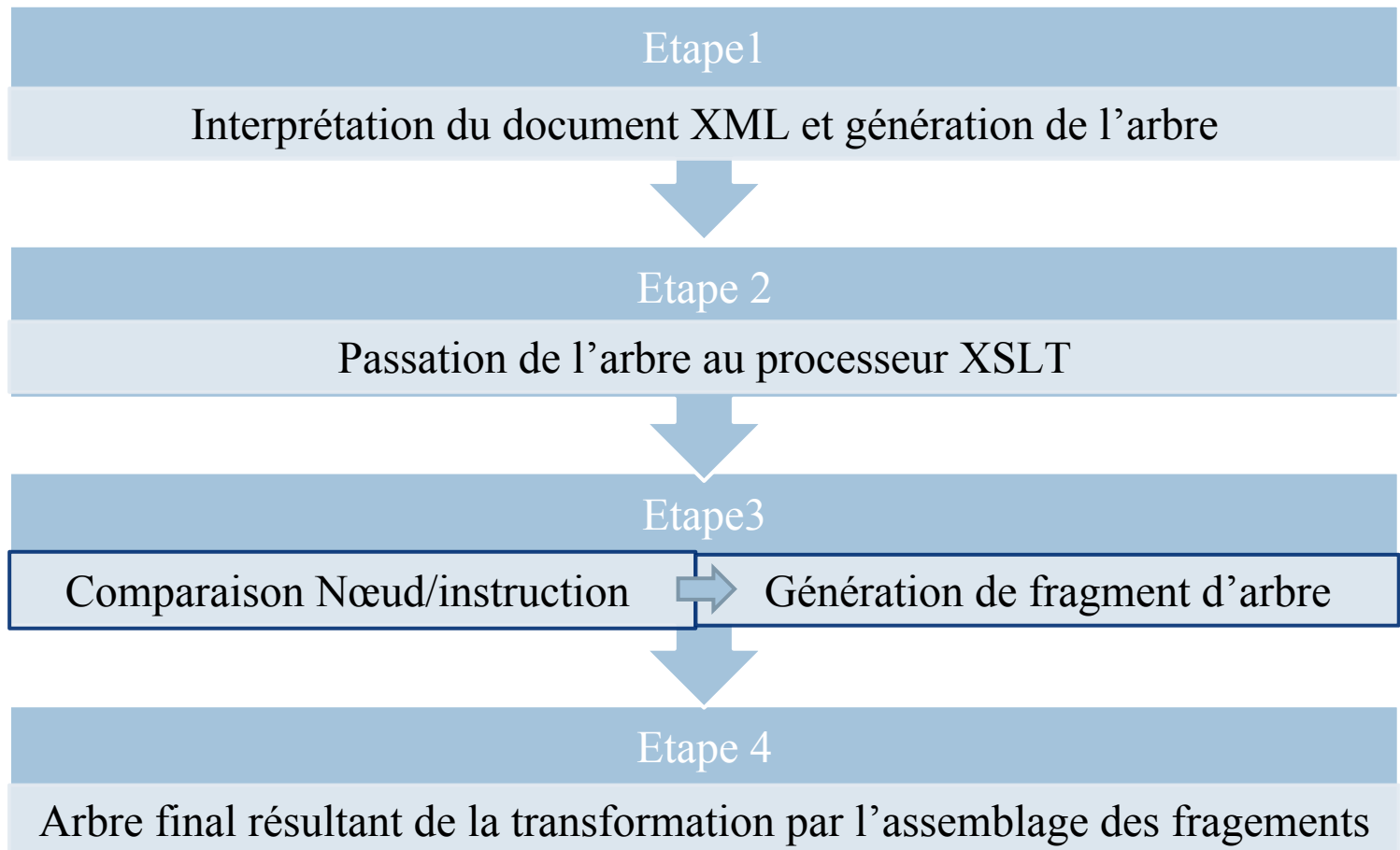
27

- Le processeur XSLT exécute les instructions de la feuille de style
- Il y a de nombreux processeurs XSLT tels que :
 - [Xalan](#) du projet Apache/XML, [libxslt](#), MSXML de Microsoft (intégré dans IE), XML Parser d'Oracle
- La plupart des éditeurs XML ont un processeur XSLT intégré

Processus de transformation

28

? Le processus de transformation se résume comme suit:



Processus de transformation

29

- Pour XSLT, un document XML est vu et analysé comme un ensemble:
 - ❑ Nœud racine
 - ❑ Nœuds éléments
 - ❑ Nœuds attributs
 - ❑ Nœud texte
 - ❑ Nœuds de traitement
 - ❑ Nœuds commentaire

Structure d'un document XSLT

30

Prologue

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" encoding="UTF-8" doctype-public="-//W3C//DTD HTML 4.01//EN"
    doctype-system="http://www.w3.org/TR/html4/strict.dtd"
    indent="yes"/>
```

Associer une DTD au
résultat de la
transformation (URI/FPI)

précise si le résultat
doit être indenté
(ajout de tabulations
ou d'espaces pour
plus de visibilité au
niveau du output ex:
html)

Corps {ensemble de règles
(templates)}

```
  <xsl:template match="/">
    <html>
      <HEAD>
        <TITLE>Titre</TITLE>
      </HEAD>
      <BODY>
        <h1>Hello </h1>
      </BODY>
    </html>
  </xsl:template>
```

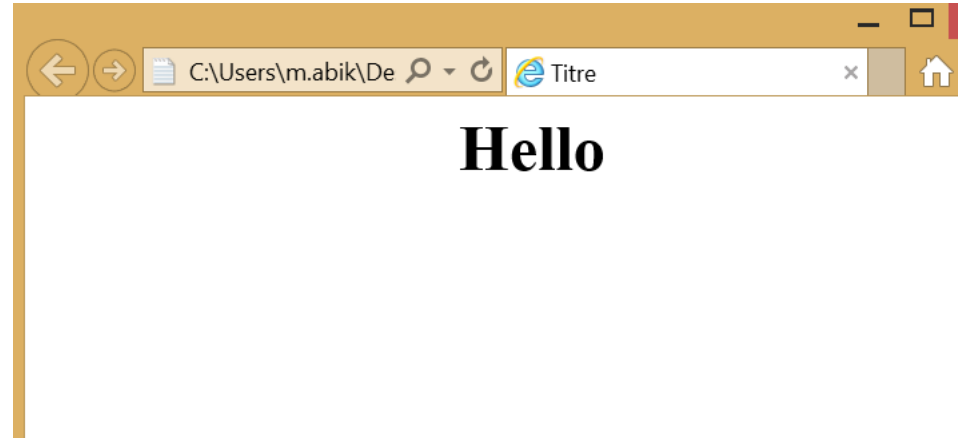
```
</xsl:stylesheet>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" encoding="UTF-8" indent="yes"/>
  <xsl:template match="/">
    <html>
      <HEAD>
        <TITLE>Titre</TITLE>
      </HEAD>
      <BODY>
        <h1 align="center"> Hello </h1>
      </BODY>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

```
<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="Test.xsl"?>

<bibliographie>
  <livre key="Michard01" lang="fr">
  <livre key="Zeldman03" lang="en">
</bibliographie>
```



Règles de transformation

32

□ Format d'une règle XSLT

`<xsl:template match="un motif" name="nom du
template " priority = "number" mode "qname" >`

[action]

`</xsl:template>`

Expression XPath : identifie
le/les noeud(s) XML du
document qui est/sont
concerné(s) par la règle et
sur le(s) quel(s) il faut
appliquer une action

Permet à un nœud d'être
traité à plusieurs reprises de
manière différente

effectue la transformation et/ou spécifie les
caractéristiques de la présentation (Composé de
plusieurs instructions)


```

<Livres>
  <livre>
    <titre> Graphisme des sites Web</titre>
    <auteur>
      <nom>Florian Schaffer</nom>
    </auteur>
    <annee edition="2">1995</annee>
  </livre>
  <livre>
    <titre> XML</titre>
    <auteur>
      <nom>E.NIEDERMAIR</nom>
    </auteur>
    <annee edition="1">2001</annee>
  </livre>
  <livre>
    <titre> ROUTEUR CISCO</titre>
    <auteur>
      <nom>Joe Habraken </nom>
    </auteur>
    <annee edition="2">2001</annee>
  </livre>
</Livres>

```

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0" xmlns="http://www.w3.org/1999/xhtml">
  <xsl:output method="html"/>
  <xsl:template match="/">
    <html>
      <HEAD>
        <TITLE>Titre</TITLE>
      </HEAD>
      <BODY>
        <h1>Liste des ouvrages </h1>
        <table border="1" cellspacing="0" cellpadding="3">
          <tr bgcolor="#6699FF">
            <td>Titre</td>
            <td>Auteur</td>
          </tr>
          <xsl:apply-templates/>
        </table>
      </BODY>
    </html>
  </xsl:template>
  <xsl:template match="livre">
    <tr>
      <td><xsl:value-of select="titre"/></td>
      <td><xsl:value-of select="auteur"/></td>
    </tr>
  </xsl:template>
</xsl:stylesheet>

```

"/" s'applique au document XML tout entier

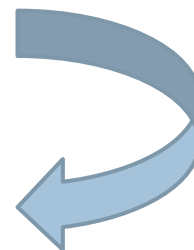
Liste des ouvrages

Titre	Auteur
Graphisme des sites Web	Florian Schaffer
XML	E.NIEDERMAIR
Routeur Cisco	Joe Habraken

Application des règles

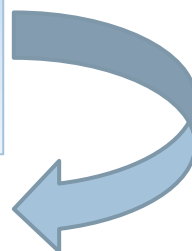
34

```
<xsl:template match="/">  
...  
</xsl:template>
```



- Se déclenche pour l'élément racine. Elle ne s'exécute qu'une seule fois et c'est souvent par une règle sur la racine qui lance le processus de transformation

```
<xsl:template match="livre">  
...  
</xsl:template>
```



- Se déclenche chaque fois que les chemins sur lesquels elle est lancée aboutissent à un élément **livre**. Elle pourra être exécutée plusieurs fois en fonction de la structure du document XML à transformer.

Modèle de traitement

35

- `xsl:apply-templates` permet d'invoquer l'application des règles de templates à un document
- Chaque application de règle (action) à un nœud produit un fragment de document XML (suite de nœuds représentant des éléments, des attributs, des instructions de traitement et des commentaires)
- Tous les fragments sont assemblés pour former le document résultat.

Déclenchement de règle

36

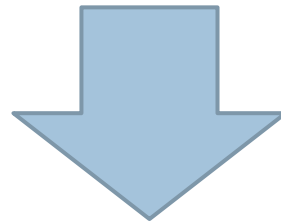
- L'élément `xsl:apply-templates` déclenche toutes les règles possibles sur les noeuds spécifiés par l'expression Xpath.
- Si on a pas précisé l'expression Xpath :les règles seront lancées pour tous les noeuds fils du nœud courant:
 - pour les nœuds auxquels est associée une règle, la règle s'appliquera,
 - pour les autres, ce sont les *règles par défaut* qui s'appliqueront (cf règles prédéfinies)

Processus de transformation

37

- Le processus de transformation consiste à appliquer des règles sur des nœuds dits *actifs* du documents source.
- Au départ, seule la racine est active et la première règle est donc appliquée à cette racine
- xsl:apply-templates active d'autres nœuds via select qui contient une expression Xpath.
- Le processus s'arrête lorsqu'il n'y a plus de nœuds actifs.

- Il est possible de vouloir traiter un même élément du document XML source plusieurs fois de plusieurs façons différentes dans une même transformation



Mode

EX: Modalité d'application des règles

```
<xsl:template match="/">
  <html>
    <HEAD>
      <TITLE>Titre</TITLE>
    </HEAD>
    <BODY>
      <h1>Liste des ouvrages </h1>
      <table border="1" cellspacing="0" cellpadding="3">
        <tr bgcolor="#6699FF">
          <td>Titre</td>
          <td>Auteur</td>
        </tr>
        <xsl:apply-templates select="//livre" mode="auteur"/>
      </table>
      <h1>Liste des ouvrages Français </h1>
      <table border="1" cellspacing="0" cellpadding="3">
        <tr bgcolor="#66FFFF" align="center">
          <td>Titre</td>
          <td>Prix</td>
        </tr>
        <xsl:apply-templates select="//livre" mode="lang"/>
      </table>
    </BODY>
  </html>
</xsl:template>
<xsl:template match="livre" mode="auteur">
  <tr>
    <td><xsl:value-of select="titre"/></td>
    <td><xsl:value-of select="auteur"/></td>
  </tr>
</xsl:template>
<xsl:template match="//livre" mode="lang">
  <xsl:if test="@lang='fr'">
    <tr>
      <td><xsl:value-of select="titre"/></td>
      <td><xsl:value-of select="auteur"/></td>
    </tr>
  </xsl:if>
</xsl:template>
</xsl:stylesheet>
```

Liste des ouvrages

Titre	Auteur
XML langage et applications	Alain Michard
Designing with web standards	Jeffrey Zeldman
Réseaux et systèmes informatiques mobiles	Samuel Pierre

Liste des ouvrages Français

	Auteur
XML langage et applications	Alain Michard
Réseaux et systèmes informatiques mobiles	Samuel Pierre

Traitement du même nœud (livre) et c'est le mod qui tranche sur la règle à appliquer

Vous remplacez // par / par livre si on remplace le match de / par /livres

Priorités des règles

40

- Lorsque plusieurs règles peuvent s'appliquer à un même nœud, le choix de la règle est d'abord dicté par les priorités d'import entre les feuilles de style puis par les priorités entre les règles

1: Priorité d'import:

- Une feuille de style importée a une priorité d'import inférieure à la feuille de style réalisant l'import
- les feuilles importées en premier ont une priorité inférieure à celles importées ensuite

Priorités des règles

41

2: Priorité entre les règles

- Usage de la propriété renseignée au niveau de la règle ou celle définit par défaut

Priorité par défaut	Motifs concernés
-0.5	pour les motifs très généraux de la forme *, @*, node(), text(), comment() ainsi que le motif /
-0.25	pour les motifs de la forme *: name ou prefix:*
0	- Tous les patterns constitués d'une seule étape XPath, avec un nom d'élément ou d'attribut, et sans prédicat (<i>name</i> ou <i>@name</i>) le pattern processing-instruction('nom')
0,25	- Tous les patterns constitués d'une seule étape XPath, avec un filtre sur tous les éléments ou attributs d'un espace de nom
0.5	tous les patterns différents des précédents cas pour patterns avec prédicats, ou constitués de plusieurs étapes

Règles prédéfinies

42

- Il existe des règles prédéfinies pour traiter les différents nœuds d'un document:
- Ces règles ont la priorité la plus faible
- Ils ne s'appliquent à un nœud que si la feuille de style ne contient aucune règle qui le concerne
- Ils provoquent un parcours récursif de l'arbre en profondeur envoyant en sortie le contenu textuel de l'arbre tout entier

Règles prédéfinies(2)

43

Si le nœud contexte est le nœud document ou un nœud element, continuer sur les fils de ce nœud

Traverser la racine et tous les nœuds

```
<xsl:template match="*|/">
  <xsl:apply-templates/>
</xsl:template>
```

Si on trouve un nœud de texte ou un attribut on retourne le contenu textuel du nœud.

- par défaut, les nœuds attributs ne sont pas atteints (il faut ajouter des règles pour les atteindre)

Sortir les feuilles « texte » et les « attributs »

```
<xsl:template match="text()|@*">
  <xsl:value-of select="."/>
</xsl:template>
```

Règles prédéfinies(3)

44

□ Commentaires et instructions de traitement

```
<xsl:template match="processing-  
instruction() | comment()" />
```

❓ Ne rien faire

EX: Règles prédéfinies

45

```
<Livres>
  <livre>
    <titre> Graphisme des sites Web </titre>
    <auteur>
      <nom> Florian Schaffer </nom>
    </auteur>
    <annee edition="2"> 1995 </annee>
    <prix> 20 </prix>
  </livre>
  <livre>
    <titre> XML </titre>
    <auteur>
      <nom> E.NIEDERMAIR </nom>
    </auteur>
    <annee edition="1"> 2001 </annee>
    <prix> 52 </prix>
  </livre>
  <livre>
    <titre> Routeur Cisco </titre>
    <auteur>
      <nom> Joe Habraken </nom>
    </auteur>
    <annee edition="2"> 2001 </annee>
    <prix> 32 </prix>
  </livre>
</Livres>
```

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl= "http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="text"/>
</xsl:stylesheet>
```



Graphisme des sites Web Florian Schaffer 1995 20 XML E.NIEDERMAIR 2001 52 Routeur Cisco Joe Habraken 2001 32

Appel Template Explicite

46

- Call-template permet l'appel d'un template indépendamment des nœuds courants

```
<xsl:call-template name="templatename">
```

...

```
</xsl:call-template>
```

— L'élément call-template peut passer une nouvelles valeur de param à un template

Ex: Passage de paramètre

47

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/livres">
    <html>
      <HEAD> <TITLE>Titre</TITLE> </HEAD>
      <BODY>
        <h1>Liste des ouvrages </h1>
        <table border="1" cellspacing="0" cellpadding="3">
          <tr bgcolor="#6699FF">
            <td>Titre</td>
            <td>Auteur</td>
          </tr>
          <xsl:for-each select="livre">
            <xsl:call-template name="liste_livre">
              <xsl:with-param name="p" select="."/>
            </xsl:call-template>
          </xsl:for-each>
        </table>
      </BODY>
    </html>
  </xsl:template>
  <xsl:template name="liste_livre">
    <xsl:param name="p"/>
    <tr>
      <td><xsl:value-of select="$p/titre"/></td>
      <td><xsl:value-of select="$p/auteur"/></td>
    </tr>
  </xsl:template>
</xsl:stylesheet>
```

Liste des ouvrages

Titre	Auteur
XML langage et applications	Alain Michard
Réseaux et systèmes informatiques mobiles	Samuel Pierre
Designing with web standards	Jeffrey Zeldman

Instruction xsl:for-each

48

- C'est un parcours itératif qui joue le même rôle que apply-templates (au niveau local).
 - ❑ Pas de déplacement vers un autre template
 - ❑ Même contexte d'entrée à la sortie de la boucle

`<xsl:for-each select="Expression XPATH">`

EX: <xsl:for-each>

49

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0" xmlns="http://www.w3.org/1999/xhtml">
  <xsl:output method="html" />
  <xsl:template match="livres">
    <html>
      <HEAD> <TITLE>Titre</TITLE> </HEAD>
      <BODY>
        <h1>Liste des ouvrages </h1>
        <table border="1" cellspacing="0" cellpadding="3">
          <tr bgcolor="#6699FF">
            <td>Titre</td>
            <td>Auteur</td>
          </tr>
          <xsl:for-each select="livre">
            <xsl:sort select="title" order="descending" />
            <tr>
              <td><xsl:value-of select="titre"/></td>
              <td><xsl:value-of select="auteur"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </BODY>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Liste des ouvrages

Titre	Auteur
XML langage et applications	Alain Michard
Réseaux et systèmes informatiques mobiles	Samuel Pierre
Designing with web standards	Jeffrey Zeldman

Instruction: <xsl:value-of>

50

- Sélectionne la valeur d'un nœud et l'insère dans la transformation

`<xsl:value-of select="expression XPath" />`

Ex:

`<xsl:value-of select="." />` : Insère le contenu textuel du nœud contextuel dans la sortie

Instruction `<xsl:sort>`

51

- Permet de trier un ensemble de nœuds en spécifiant le critère de sélection après un `xsl:apply-templates` ou un `xsl:for-each`.

```
<xsl:for-each select="livre">
  <xsl:sort select="titre" order="descending"/>
  <tr>
    <td><xsl:value-of select="titre"/></td>
    <td><xsl:value-of select="auteur"/></td>
  </tr>
</xsl:for-each>
```

Liste des ouvrages

Titre	Auteur
XML langage et applications	Alain Michard
Réseaux et systèmes informatiques mobiles	Samuel Pierre
Designing with web standards	Jeffrey Zeldman

Contrôle de flux

52

- XSLT propose deux instructions conditionnelles

- `<xsl:if test = boolean-expr>`

`</xsl:if>`

- `<xsl:choose>`

`<xsl:when test=boolean-expr> ... </xsl:when>`

`<xsl:when test=boolean-expr> ... </xsl:when>`

`<xsl:otherwise> ... </xsl:otherwise>`

`</xsl:choose>`

Contrôle de flux

53

- Plusieurs types de test disponibles.

Condition	Description
$a = b$	Cette condition vérifie que la valeur de l'élément a est égale à la valeur de l'élément b.
$\text{not}(a = b)$	Cette condition vérifie que la valeur de l'élément a n'est pas égale à la valeur de l'élément b.
$a \text{ \< } b$	$a < b$. Cette condition vérifie que la valeur de l'élément a est strictement inférieure à la valeur de l'élément b.
$a \text{ \<= } b$	$a \leq b$. Cette condition vérifie que la valeur de l'élément a est inférieure ou égale à la valeur de l'élément b.
$a \text{ \> } b$	$a > b$. Cette condition vérifie que la valeur de l'élément a est strictement supérieure à la valeur de l'élément b.

- possible d'effectuer plusieurs tests à la fois (and, or)

Variables

54

- XSLT permet de définir des variables pour stocker un résultat intermédiaire

```
<xsl:for-each select="livre">
  <xsl:variable name="type" select="."/>
  <xsl:if test="contains($type, 'XML')">
    <tr>
      <td><xsl:value-of select="titre"/></td>
      <td><xsl:value-of select="auteur"/></td>
    </tr>
  </xsl:if>
</xsl:for-each>
```

Liste des ouvrages (XML)

Titre	Auteur
XML langage et applications	Alain Michard



- On ne peut pas modifier la valeur d'une variable (sauf dans le cas de for-each où la variable est considérée locale dans le for)
- la portée d'une variable : Dans l'élément qui contient sa déclaration
- Une variable est référencée avec la notation \$

Ex: Passage de paramètre

55

```
<html> <HEAD>
<TITLE>Bibliographies</TITLE> </HEAD>
<BODY>
  <h1>Liste des ouvrages</h1>
  <table border="1" cellspacing="0" cellpadding="3">
    <tr bgcolor="#6699FF">
      <td>Titre</td>
      <td>Auteur</td>
    </tr>
    <xsl:for-each select="livre">
      <xsl:call-template name="liste_livres">
        <xsl:with-param name="p" select="."/>
      </xsl:call-template>
    </xsl:for-each>
  </table>
</BODY>
</html>
</xsl:template>
<xsl:template name="liste_livres">
  <xsl:param name="p">
    <tr>
      <td><xsl:value-of select="$p/titre"/></td>
      <td><xsl:value-of select="$p/auteur"/></td>
    </tr>
  </xsl:template>
```

Liste des ouvrages

Titre	Auteur
XML langage et applications	Alain Michard
Réseaux et systèmes informatiques mobiles	Samuel Pierre
Designing with web standards	Jeffrey Zeldman

Paramètres des Templates

56

- Un Template peut avoir des paramètres

<xsl:param name="nom" select="valeur par défaut" />

- On charge la valeur d'un paramètre comme suit:

<xsl:with-param name="nom_paramètre" select="valeur_du_paramètre" />

- **<xsl:with-param>** est appelé par le call-template ou apply-templates

Ex2: Passage de paramètre

```
<xsl:template match="/livres">
```

```
  <html>
```

```
    <HEAD> <TITLE>Titre</TITLE> </HEAD>
```

```
    <BODY>
```

```
      <h1>Liste des ouvrages (Français) </h1>
```

```
      <table border="1" cellspacing="0" cellpadding="3">
```

```
        <tr bgcolor="#6699FF">
```

```
          <td>Titre</td>
```

```
          <td>Auteur</td>
```

```
        </tr>
```

```
        <xsl:apply-templates select="livre">
```

```
          <xsl:with-param name="p" select="'fr'" />
```

```
        </xsl:apply-templates>
```

```
      </table>
```

```
    </BODY>
```

```
  </html>
```

```
</xsl:template>
```

```
<xsl:template match="livre">
```

```
  <xsl:param name="p"/>
```

```
  <xsl:if test="@lang=$p">
```

```
    <tr>
```

```
      <td><xsl:value-of select="titre"/></td>
```

```
      <td><xsl:value-of select="auteur"/></td>
```

```
    </tr>
```

```
  </xsl:if>
```

```
</xsl:template>
```

Liste des ouvrages (Français)

Titre	Auteur
XML langage et applications	Alain Michard
Réseaux et systèmes informatiques mobiles	Samuel Pierre

Les instructions de création

58

□ `<xsl:element />`

❓ permet de créer un nouveau élément lors de la transformation du document XML

**`<xsl:element name="nom" namespace = "espace_de_nom">`
valeur de l'élément**

`</xsl:element>`

Les instructions de création(suite)

59

□ `<xsl:attribute />`

❓ Permet de créer des attributs à des éléments XML ou HTML

`<xsl:attribute name="nom">valeur </xsl:attribute>`

Ex:

```
<xsl:template match="/">
  <xsl:element name="couleur">
    <xsl:attribute name="primaire">oui</xsl:attribute>
    Rouge </xsl:element>
  </xsl:template>
```

Import et include

60

Pour composer une feuille de style à partir de plusieurs fichiers XSL

□ Importer

`<xsl:import href "uri"/>`

- Une feuille de style importée a une priorité d'import inférieure à la feuille de style réalisant l'import
- les feuilles importées en premier ont une priorité inférieure à celles importées ensuite

□ Inclure

`<xsl:include href "uri"/>`

❓ comportement équivalent à `xsl:import`

- mais pas de possibilité d'écraser une définition importée par une définition de plus haut-niveau
- erreur si deux définitions similaires

Webographie

61

- Livres

Livre XML cours et Exercice :

<https://campusbruxelles.files.wordpress.com/2014/02/2007-eyrolles-xml-cours-et-exercices.pdf>

- Espace de nom :

<http://deptinfo.unice.fr/twiki/pub/Minfo03/ServletEtXml/01c-xml-namespaces.pdf>

- DTD :

http://www2.amk.fi/digma.fi/www.amk.fi/material/attachments/vanhaamk/digma/5h5F5r4b3/DTD_0.7.swf

- XPATH:

<http://www.w3.org/TR/xpath/>

<http://www.loria.fr/~abelaid/Enseignement/miage-m1/XSL-Xpath.pdf>

- XSLT:

http://miage.univ-nantes.fr/miage/D2X1/chapitre_xslt/section_regles.htm#22

- DOM et SAX

<http://deptinfo.unice.fr/twiki/pub/Minfo03/ServletEtXml/06-xml-dom-sax.pdf>

<https://www.lri.fr/~benzaken/documents/sldomsax.pdf>

Webographie

62

- Livre XML cours et Exercice : <https://campusbruxelles.files.wordpress.com/2014/02/2007-eyrolles-xml-cours-et-exercices.pdf>
 - <http://www.telug.ca/inf6450/mod2/chapitre6.xml>
 - Espace de nom : <http://deptinfo.unice.fr/twiki/pub/Minfo03/ServletEtXml/01c-xml-namespaces.pdf>
 - Dtd : http://www2.amk.fi/digma.fi/www.amk.fi/material/attachments/vanhaamk/digma/5h5F5r4b3/DTD_0.7.swf
 - DOM et SAX <http://deptinfo.unice.fr/twiki/pub/Minfo03/ServletEtXml/06-xml-dom-sax.pdf>
 - ? <https://www.lri.fr/~benzaken/documents/sldomsax.pdf>
 - ? <http://www-lium.univ-lemans.fr/~lehuen/master1/xml/cours/xmldiapo3.pdf>
- XQUERY
 - ? Chapitre 6 de base de donn es avanc es XML-Requ te Xquery, Yannik Benezeth
<http://ilt.u-bourgogne.fr/benezeth/teaching/DUTIQ/BDDS3/CM6.pdf>
 - ? Xquery : Une extension de Xpath   la mode SQL Un concurrent d'XSLT ?, Yves Bekkers
https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=18&cad=rja&uact=8&ved=0ahUKEwjFw83x6NfQAhXGbRQKHVG2DQ04ChAWCE0wBw&url=http%3A%2F%2Fwww.irisa.fr%2FLIS%2FMembers%2Fbekkers%2Fxmldossier%2Fxquery%2Fattachment_download%2Ffile&usg=AFQjCNE1Ih2QwPJ21fNx68jdzdZtpuOdQw&sig2=63NIKIMEdlwb4dPffFxe5w
 - ? https://www.ibisc.univ-evry.fr/~serena/coursBDA0910_4.pdf
- XML et SGBD
 - ? http://miage.univ-nantes.fr/miage/D2X1/chapitre_bdxml/section_sgbd_xml.htm
 - ? <http://peccatte.karefil.com/software/rbouret/xmlbd.htm>
 - ? <http://code.ulb.ac.be/dbfiles/Ver2006amastersthesis.pdf>