www.mas-ict.ch

Hes·so
Haute Ecole Spécialisée de Suisse occidentale

Module: XML et les bases de données

Rappel sur le monde XML

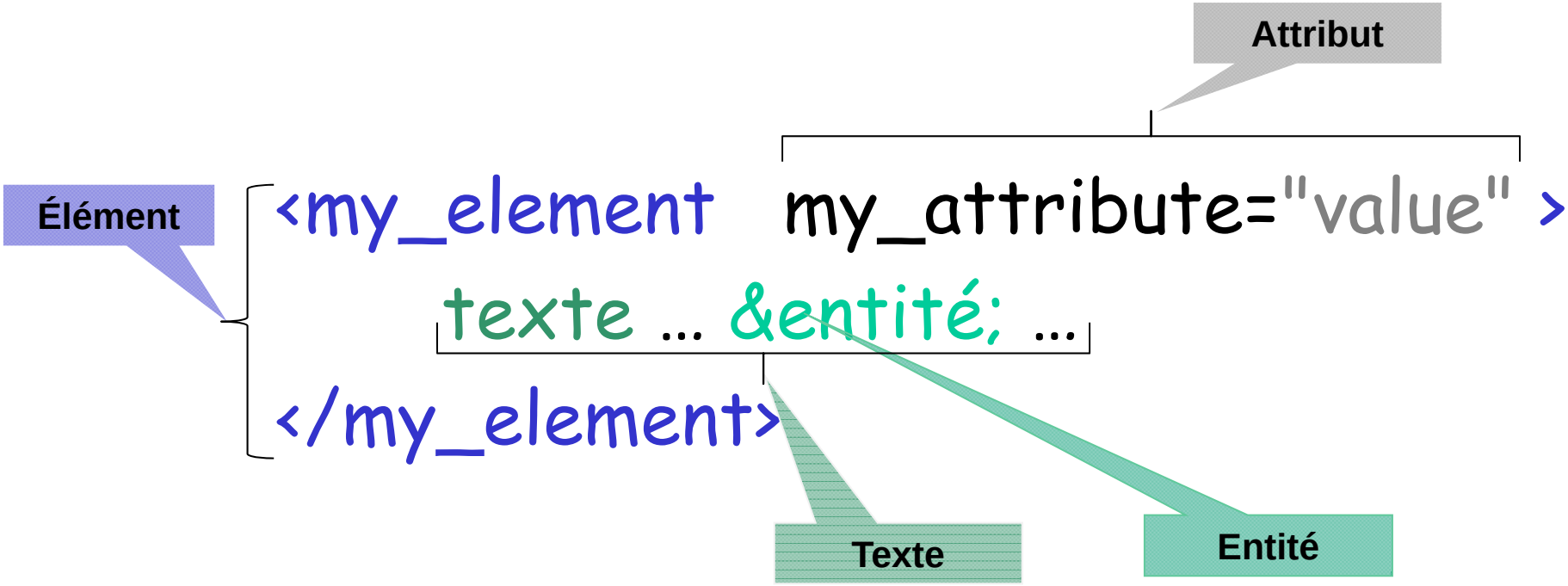
Houda Chabbi Drissi

houda.chabbi@hefr.ch

Source: G. Gardarin (livre et cours)

eXtended Markup Language

XML est un langage de **balisage extensible**, qui permet à chaque utilisateur de définir ses balises (tags) propres avec donc une valeur sémantique déterminée.

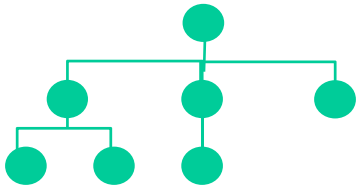


Qu'est ce que XML ?

- Séparation du **fond** (contenu) de la **forme** (présentation).
- Texte lisible et personnalisable

XML = arbre

```
<releveTemperature>
  <ville>Paris</ville>
  <pays>France</pays>
   <temp echelle="C">18</temp>
</releveTemperature>
```

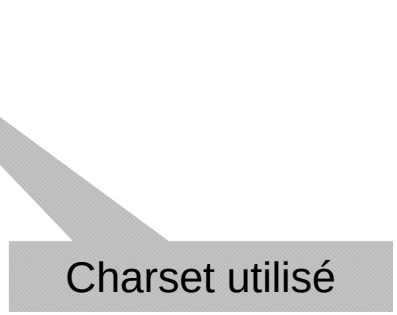


Entête XML

Fichier XML a toujours le même entête

```
<?xml version="1.0" ?>

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



Eléments, attributs et namespaces

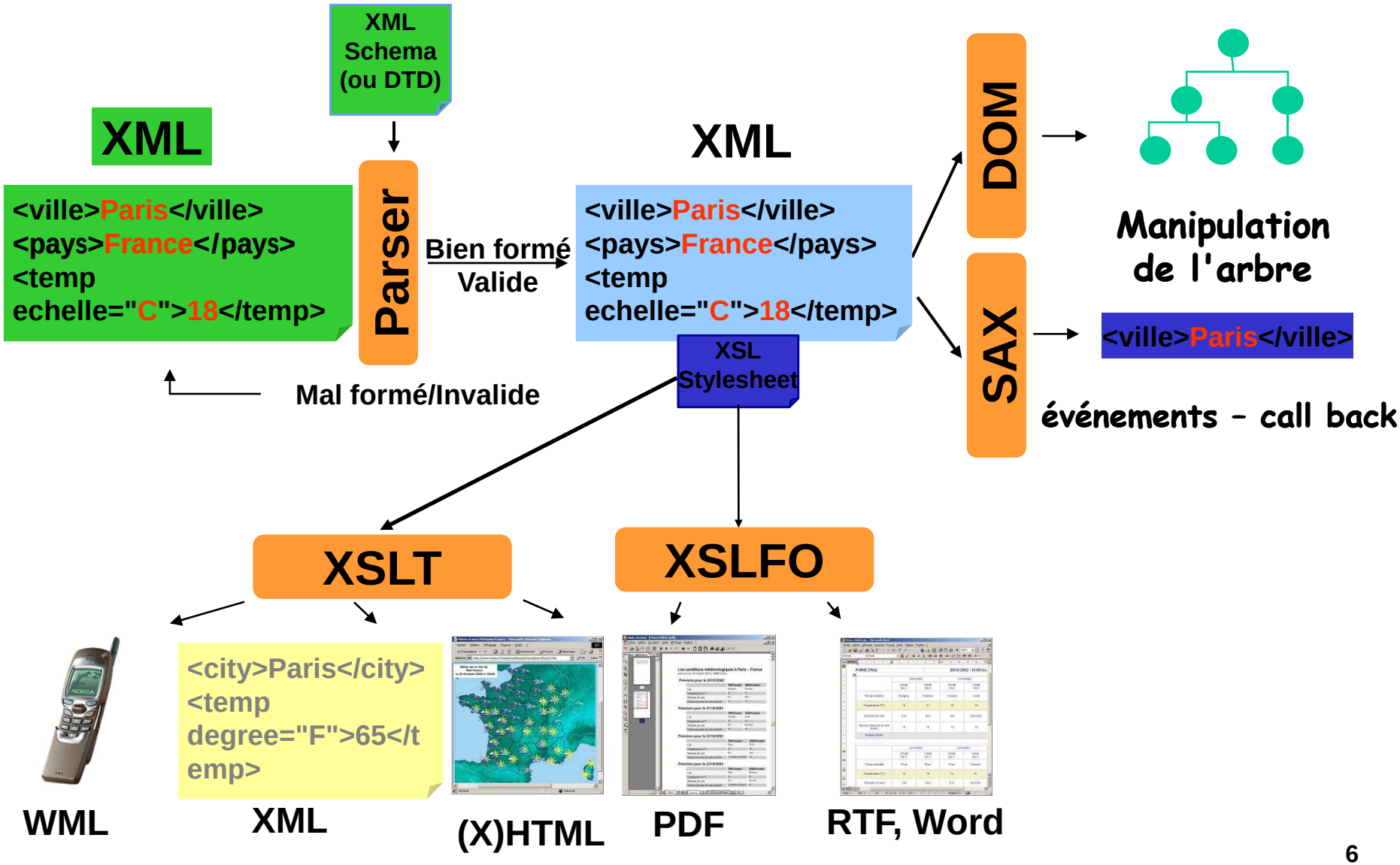
```
<?xml version="1.0"?>
<a:section xmlns:a="http://www.eif.ch/" >
  <a:exercice>
    …
  </a:exercice>
</a:section>
```

préfixe

Déclaration du namespace

Les espaces de noms évitent les collisions

Les technologies vues autour de XML



Grammaire et Typage

- **DTD** définie l'ensemble des éléments et attributs (pas XML).

  ⇒ Modélisation textuelle.

- **XML Schéma** définie les éléments, les attributs et leurs types.

  ⇒ Plusieurs mécanismes de modélisation.

DTD

Data Type Definition: Définie la structure d'un document XML

```
<!DOCTYPE serie SYSTEM "serie.dtd">

<!ELEMENT serie (exercice*)>
<!ELEMENT exercice (title, (author)?, (paragraph)*)>
 <!ATTLIST exercice
     title CDATA #IMPLIED>

<!ELEMENT paragraph (#PCDATA)>
<!ELEMENT author (#PCDATA)>

<!ENTITY copyright "Copyright EIAFR">
```

#REQUIRED
#FIXED
#IMPLIED

Déclarer une DTD dans XML

La balise !DOCTYPE déclare la DTD

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE serie SYSTEM « serie.dtd">
<serie>
  <exercice title = "exo1">
    <author>Moi</author >
    <paragraph>
      blabla blabla…
      &copyright;
    </paragraph>
  </exercice>
 </serie>
```

DTD et namespace mariage laborieux!

XML Schéma

Un schéma définie la structure et les types

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name= "serie" type="serieType">
  <xsd:complexType name="exerciceType">
   <xsd:sequence>
     <xsd:element name= "exercice" type="exerciceType"/>
   </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="exerciceType">
   <xsd:element name="titre" type="xsd:string" use="required" />
   <xsd:sequence>
      <xsd:element name="paragraph" type= "paragraphType" />
   </xsd:sequence>
   <xsd:attribute name="auteur" type="xsd:string" />
 </xsd:complexType>
</xsd:schema>
...
```

Déclarer un schéma

Sur l'élément racine, on définie
une instance de schéma

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<paper xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:nonamespaceschemaLocation= "serie.xsd"  >
 <exercice title="exo1">
  <author>Moi</author >
  <paragraph>Le but de cette exercice</paragraph>
 </exercice>
</paper>
```

Ici pas de namespace sinon autre variante

Parseurs / transformateurs

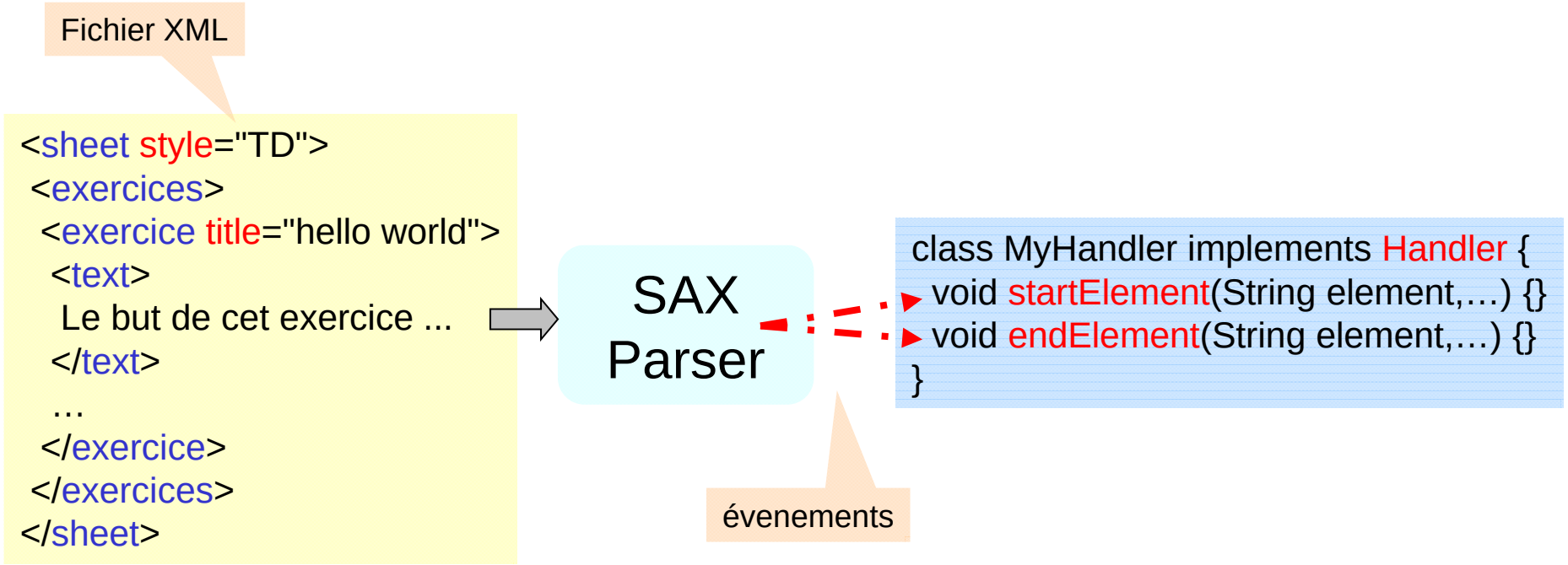
- Parseurs XML, transforme le texte du XML en éléments et attributs.

- 2 APIs :
  - SAX: Simple Api for XML
  - DOM: Document Object Model

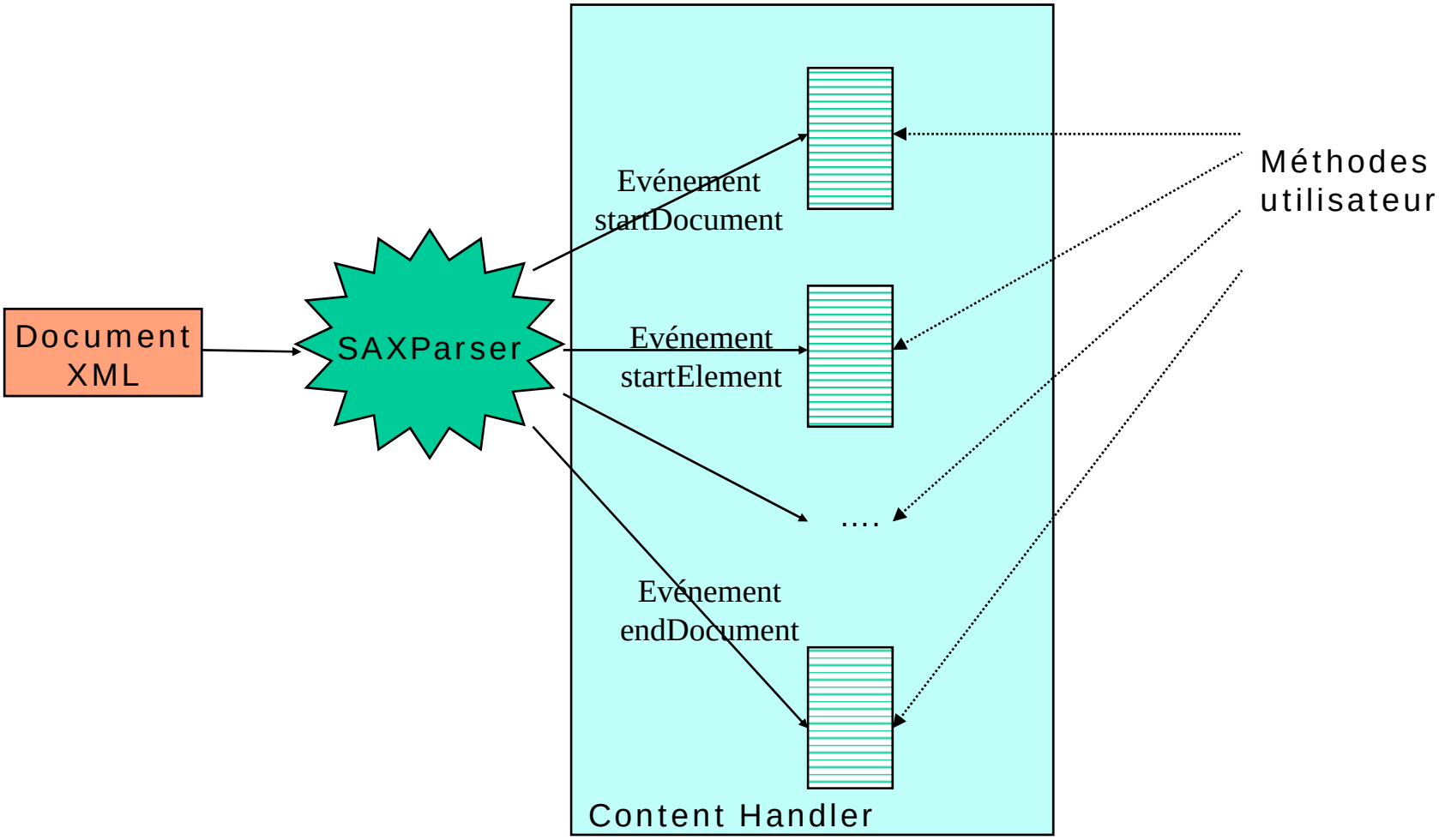
- Transformateur XML, XSL effectue des transformations d'arbre

Norme SAX 2.0

- Un événement pour chaque élément XML (start,end)
- Programmation événementielle.



Principe de fonctionnement SAX (1)



Principe de fonctionnement SAX (2)

Les Handlers de SAX

L'interface **XMLReader** définie l'interface d'un parseur SAX 2.0

- set/getContentHandler()
- set/getDTDHandler()
- set/getEntityResolver()
- set/getErrorHandler()
- parse(InputSource)
- parse(String)
- get/setFeature(name,value)

Écoute la DTD

Résout les entités

Réceptionne les éléments XML

Reçoit les erreurs

Démarre le parsing

Change les fonctionnalités

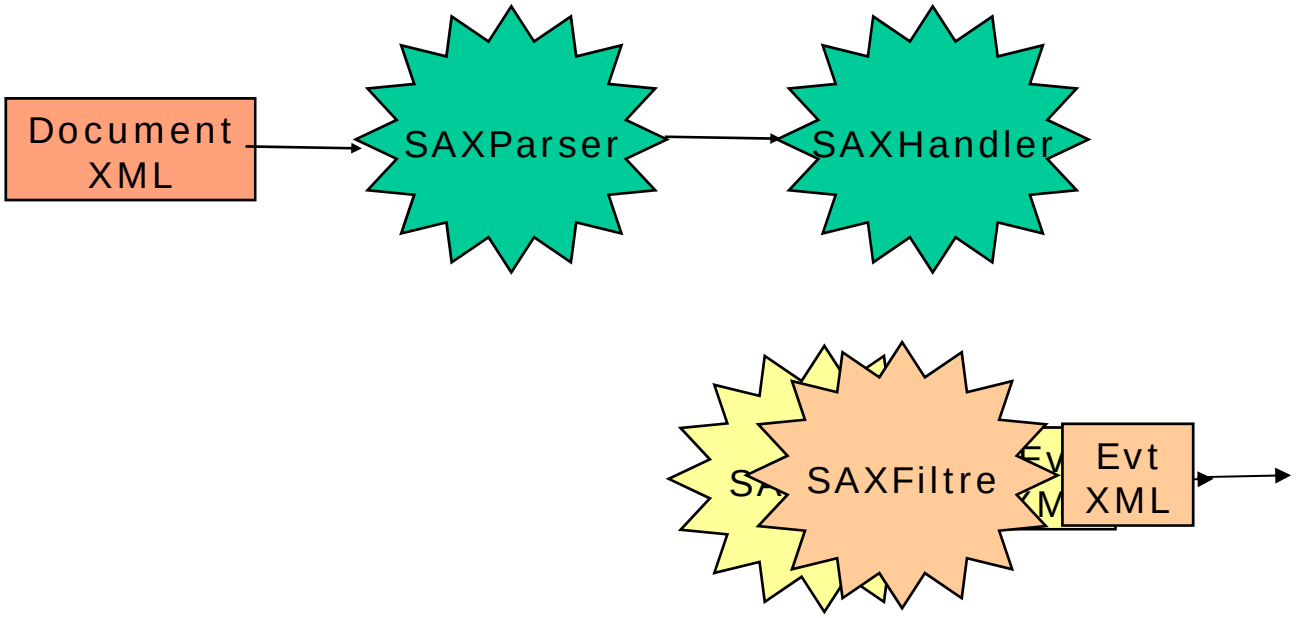
Les Filtres

- Le filtre s'intercale entre le parseur et le handler.

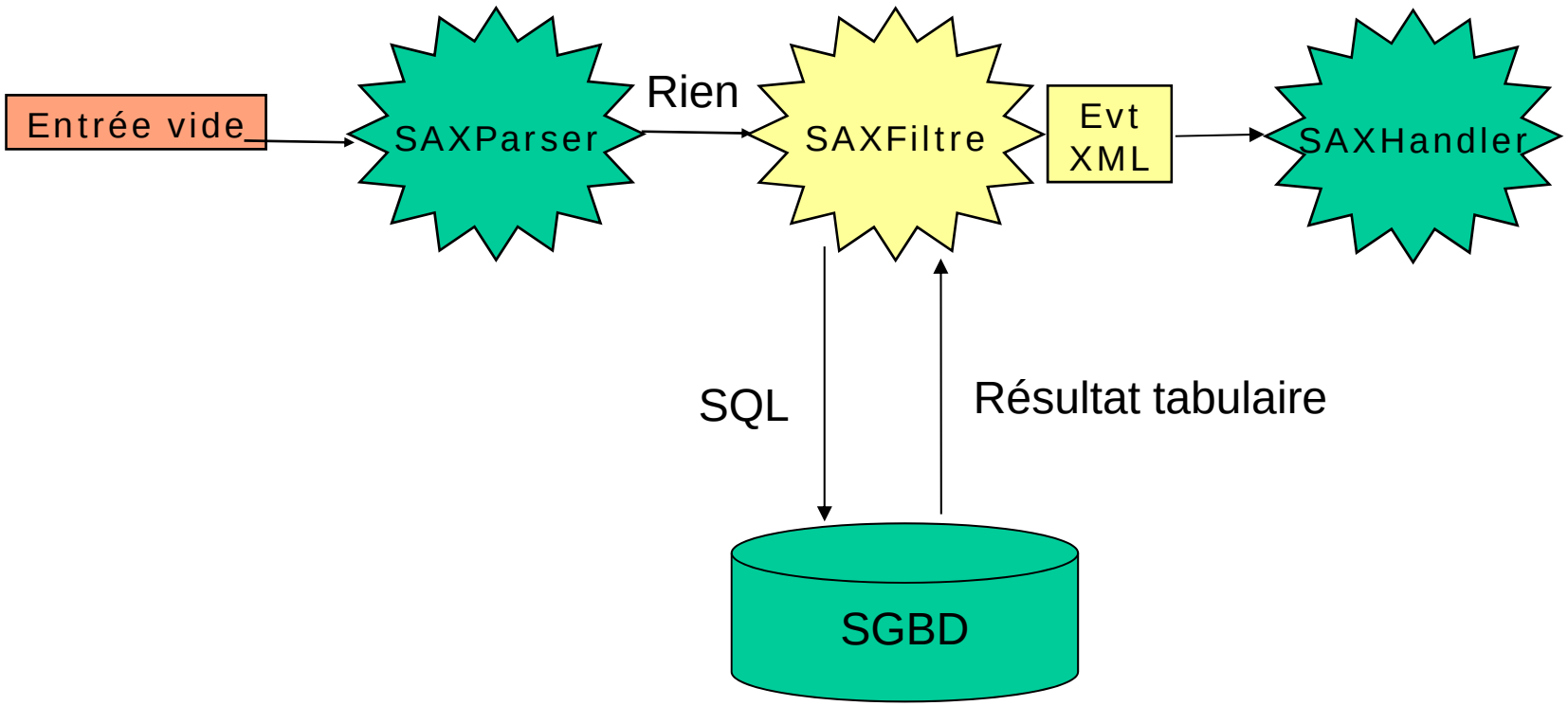
- Il réceptionne tous les évènements générés par le parseur et décide de ce qu'il doit en faire avant de les passer au handler:
  - Peut passer l'évènement tel quel
  - Peut le faire disparaître
  - Faire passer un ou plusieurs évènements à la place!

Implante l'interface XMLFilter qui hérite de XMLReader.

Principe de fonctionnement des filtres SAX



Utilisation possible des filtres SAX

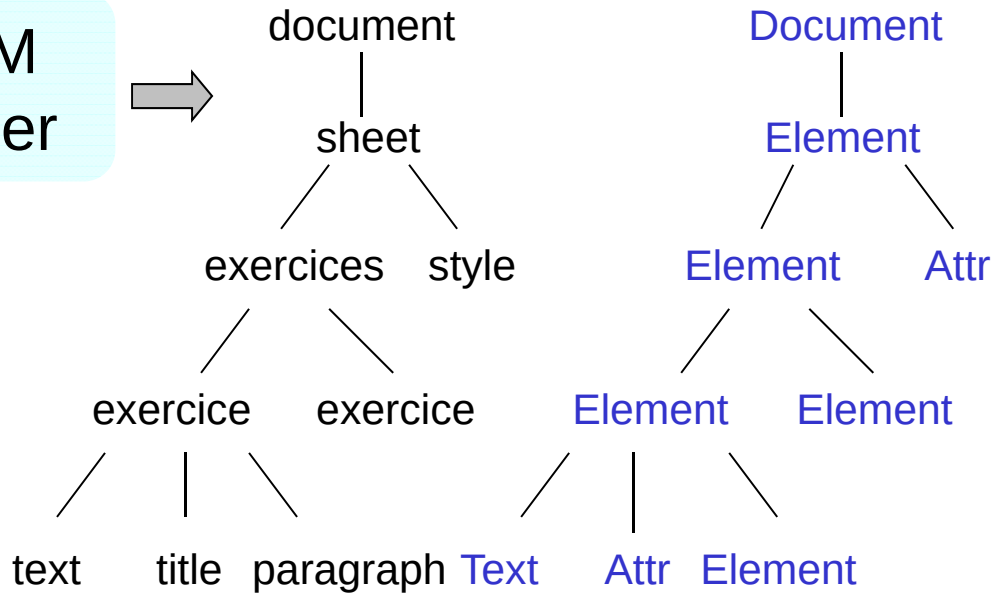


XML et arbre DOM

- Stocke l'arbre en mémoire.
- Pratique à parcourir.
- Des méthodes à dispositions pour manipuler l'arbre.

```
<sheet style="TD">
 <exercices>
  <exercice title="hello world">
   <text>
    Le but de cet exercice ...
   </text>
   ...
  </exercice>
 </exercices>
</sheet>
```

DOM Parser



XSLT et XSL-FO

- XSLT permet de spécifier des transformations d'arbre (ajouter/retirer des éléments)

- XSL-FO fournit un langage de description pour la publication de contenu

Transformateur XSLT

Effectue des transformations d'arbres
sur un source XML

```
<sheet style="TD">
 <exercices>
  <exercice title="hello world">
   <text>
    Le but de cet exercice ...
   </text>
   ...
  </exercice>
 </exercices>
</sheet>
```

Fichier XML

```
<xsl:template select='exercices'>
 <ul>
   <xsl:apply-template/>
 </ul>
</xsl:template>
```

XSLT
Transformateur

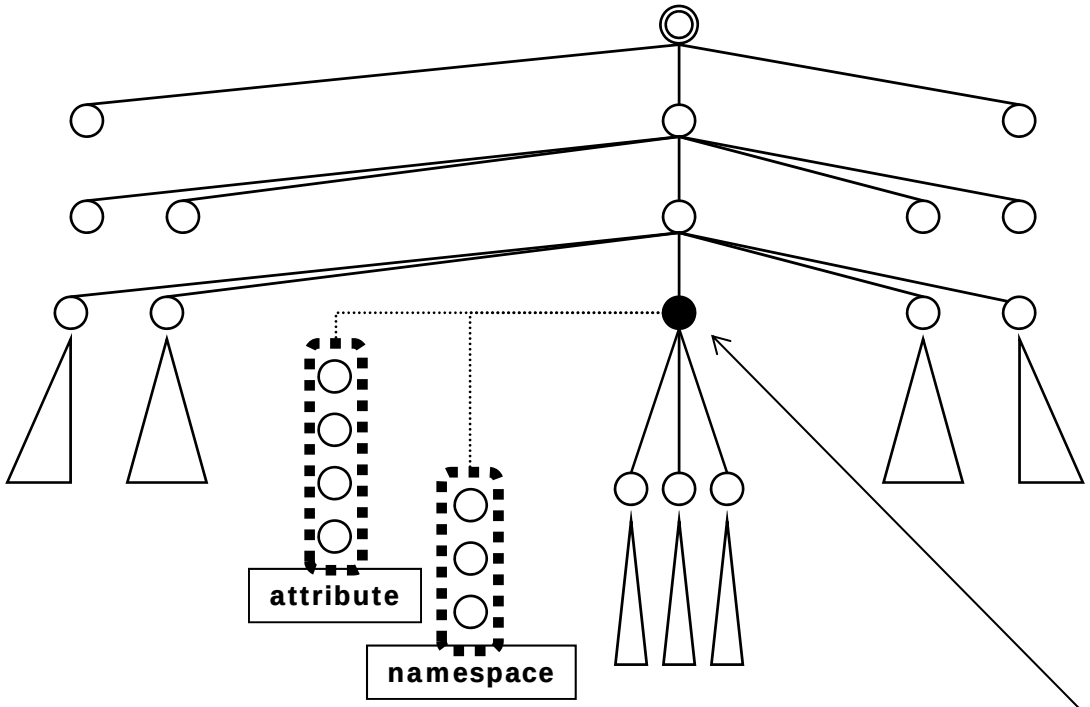
```
<html><body>
 <ul>
  <li>hello world</li>
 </ul>
</body></html>
```

Fichier XML/XHTML

XPATH

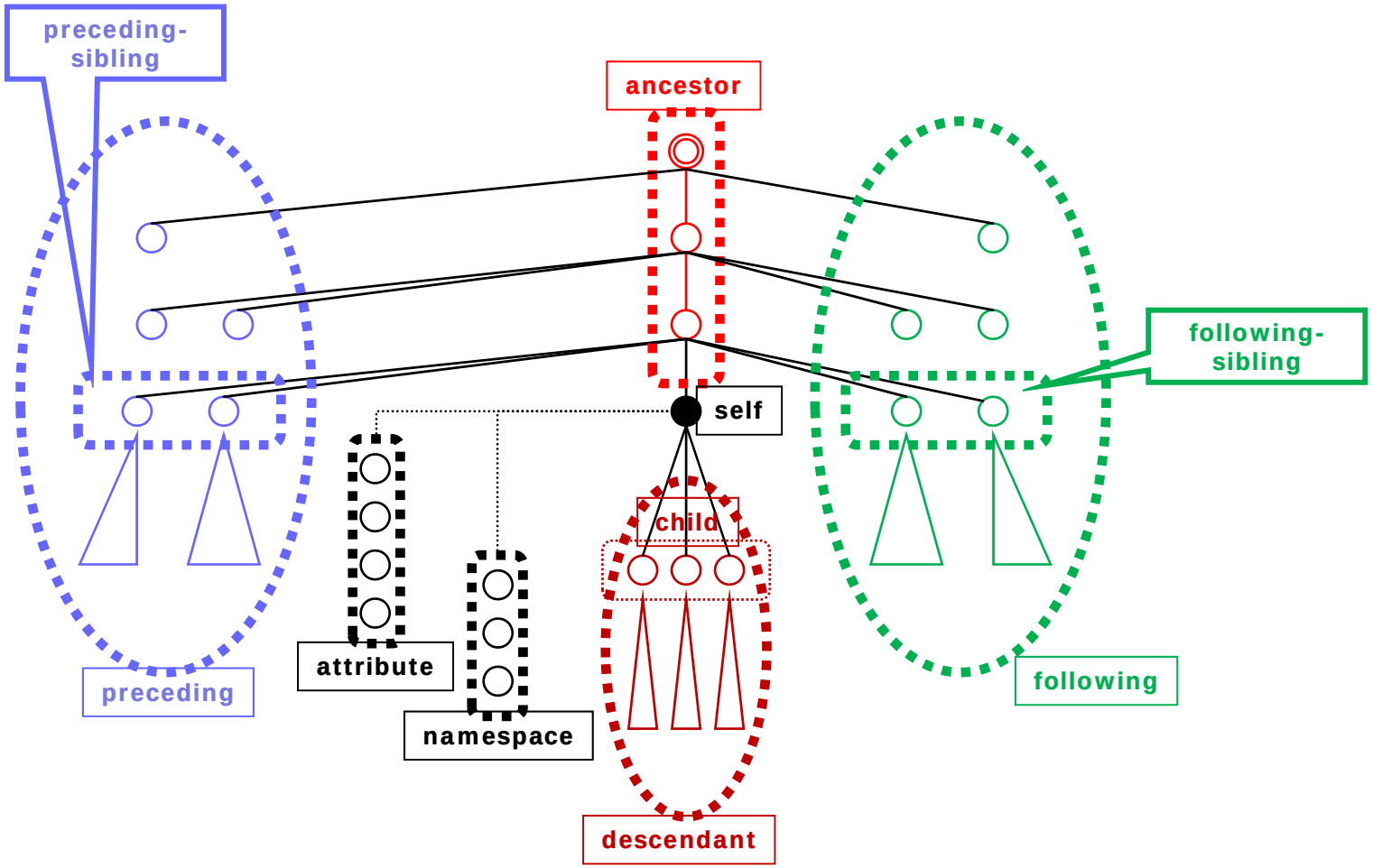
- Langage pour identifier des nœuds sur certains critères
- Non XML, ressemble aux langages OS
- Utilisé dans XSL et Xquery
  - XPATH 1
  - XPATH 2

Axes de navigation de XPATH



Un arbre XML + un nœud courant

Axes de navigation de XPATH



Expression XPath

Permette de spécifier un ensemble de nœuds sous forme de chemin

- exercice/text[.="hello"]

  N'importe quel exercice ayant un élément text contenant hello comme texte

- /feuille/exercices/exercice[@title]
  sélectionne les exercices ayant un attribut title

- exercice[@title="Hello World"]
  sélectionne l'exercice ayant "Hello World" pour valeur de l'attribut title

XPath : les jokers

- /feuille//exercice tous les exercices ayant feuille comme ancètre
- /feuille/. Sous-élément de feuille qui ne sont ni des attributs ni du texte.
- /feuille/node() n'importe quel sous-élément de feuille
- /feuille/@* n'importe quel attribut de feuille

XPath : les fonctions

- /exercice[last()]
  sélectionne le dernier élément exercice

- /exercice[position() <=5]
  sélectionne les 5 permiers exercices

- /exercice[count(paragraph) =2]
  sélectionne les exercices ayant deux paragraphes

XPath 2.0

Étend XPath 1.0 par:

- L'ajout du "for … return …"
- L'ajout du "if…then…else"
- Des expressions de quantifications some et every
- Ajout de fonctions et d'opérateurs intersect et except
- Respect du XML Data Model (plus que infoset).
  Support des types de XML Schema.

Type dans XPath 2.0

- XPath 1.0
  - Boolean, Number, String
  - Ensemble de nœuds (Nodeset): Pas de duplicata

- XPath 2.0
  - Tous les types primitifs de XML Schema
  - Séquence de nœuds ou de valeurs : à l'encontre du XPath 1.0: Ordonnée (pas forcément l'ordre du document), duplicatas admis.

XML Data Model (1)

- Modèle abstrait data model pour les données XML (l'équivalent du modèle relationnel dans SGBDR)

- Pas de standard de stockage ou d'accès pour l'instant.

- Une instance de ce « data model » est une *séquence* composée de zéro ou plusieurs *items*. La séquence vide est souvent considérée comme "null value".

XML Data Model (2)

- Les items sont:
  - Des nœuds ou des valeurs atomiques

- Les nœuds sont:

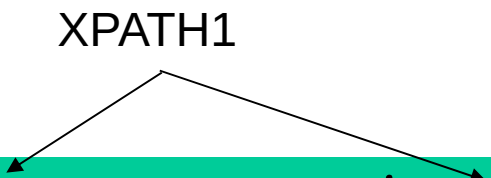
  document | element | attribute | text | namespaces | PI | comment

- Valeurs atomiques sont:
  - Des Instances de tous les types atomiques de XML Schema

  string, boolean, ID, IDREF, decimal, QName, URI, ...
  - untyped atomic values

- Typed (I.e. schema validated) et untyped (I.e. pas de validation XSD) pour les nœuds comme pour les valeurs

Iteration



- for $x in //Book return $x/Price
  - Retourne une séquence de noeuds price
  - équivalent à //Book/Price

(<Price>32.95</Price>, <Price>18.25</Price>, …)

Obtenir les valeurs

- for $x in //Book
  return $x/Price/text()
  - équivalent à //Book/Price/text()

- for $x in //Book
  return number($x/Price)
  - retourne une séquence des valeurs numériques du prix

    (32.95, 18.25, …)

Expressions Conditionelles

```
for $x in //Book
return
  if (count($x/Author) > 2)
        then $x/Price * .5 -- nombres
        else $x/Price -- noeuds
```

☺ la séquence résultante comporte des noeuds et des nombres

XPath 1.0 vs XPath 2.0

Soit à déterminer si un noeud $x appartient à l'ensemble des noeuds /foo/bar

- XPATH1

```
count(/foo/bar)=count(/foo/bar | $x)
```

- XPATH2

```
$x intersect /foo/bar
```

Expressions de quantification

- //Book[some $a in Author satisfies
  starts-with($a/Lastname, "T")]
  - Books with some author whose lastname starts with T
- //Book[some $a in Author satisfies $a/Lastname="Toto"]
  - Books with some author whose lastname is Toto
- //Book[every $a in Author satisfies $a/Lastname="Toto"]
  - Books all of whose authors‘ lastnames are Toto

Duplication de nœuds dans XPath 2.0

- //Author[Lastname="Toto"]/..

XPATH1

  – Xpath1 élimine les duplicatas, donc un livre avec 2 auteurs ayant un lastname = Toto n'apparait qu'une fois.

- for $a in //Author[Lastname="Toto"]
  return $a/..

XPATH2

  - Ici un livre avec 2 auteurs ayant un lastname = Toto apparait deux fois.

- distinct-nodes(for $a in //Author[Lastname="Toto"]
  return $a/..)

  - Elimination explicite des duplicatas

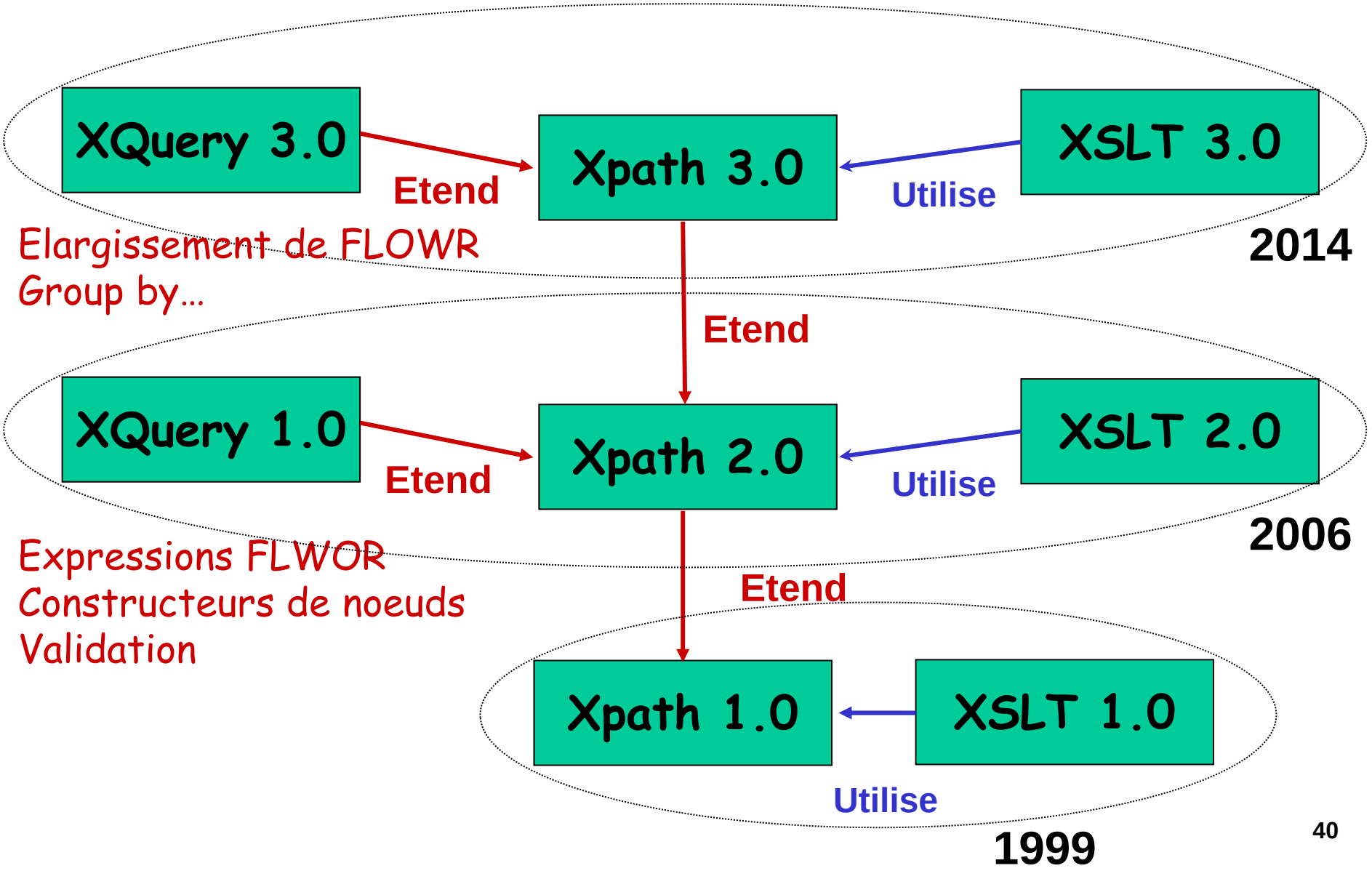
XPath 3.0: W3C Recommendation 08 April 2014

- XPath 2 a été augmenté pour devenir un “langage de programmation” à part entière avec définition de fonction etc. (http://www.w3.org/TR/xpath-30/)

Here are some of the new features in XPath 3.0:
Dynamic function call (**3.2.2 Dynamic Function Call**).
Inline function expressions (**3.1.7 Inline Function Expressions**).
Support for union types.
Support for literal URLs in names, using EQNames.
A string concatenation operator (**3.6 String Concatenation Expressions**). ||
A mapping operator (**3.14 Simple map operator (!)**).

- Trop récent pour avoir des implémentations dans tous les outils qui manipulent du XPath. BaseX implémente déjà la nouvelle norme!

Xpath et XQuery, XSLT



Où intervient XML?

- XML peut être utilisé pour l'échange de données:
  - d'un site Web vers le navigateur d'un utilisateur
  - ou entre applications, de machine à machine.

- Il convient à une architecture n-tiers:
  - le *client*, où les données sont affichées ou traitées par d'autres applications,
  - le *middle-tiers* où les logiques applicatives agissent sur les données
  - les *bases de données*.