<*XML num="2"*>Langage Web 2</*XML*>

- Emilien Bondu (Airbus Defence and Space)
- emilien.bondu@cassidian.com

Sommaire

- Cours 1 : Introduction à XML et validation (XSD)
- Cours 2 : Validation (DTD, Relax NG, Schematron) et transformation (XPath, XSL; XSL-FO)
- Cours 3 : Recherche XML (XQuery), Base de données XML, Liens XML, Manipulation XML en Java: Dom, Sax
- Cours 4 : Manipulation XML en Java : StAX, Data-binding, JavaEE : tomcat
- Cours 5 : Manipulation XML en JavaEE: servlet, JSP, ExpressionLanguage, TagLib
- Cours 6 : TagLib (suite), Spring, JSF, AJAX

Plan

- Rappels
- DTD
- RelaxNG
- Schematron
- XSL - XSL-FO

Rappels

- XML est un :
  - Méta-langage
  - Standard OASIS
  - Langage à balise
  - Moyen d’échanger des données entre applications

Rappels

- XML permet :
  - de valider, stocker et rechercher des données
  - de séparer la forme du contenu
  - de créer de nouveaux formats de données
  - de minimiser la taille des messages échangés

Rappels

- Quelle règle ne s'applique pas à XML ?
  - Les documents XML doivent être lisibles par l'homme et raisonnablement clairs
  - La concision dans le balisage de XML est peu importante
  - Le langage XML doit permettre de programmer aisément des applications.
  - XML doit pouvoir être utilisé sans difficulté sur Internet

Rappels

- XML est utilisé dans :
  - le standard HTML
  - le standard de document OpenDocument
  - le format de représentation de connaissance RDF
  - le standard de notation MathML

Rappels

- Un XML Schema permet de :
  - vérifier qu'un document XML est bien formé
  - vérifier qu'un document XML est valide
  - de vérifier la présence d'un élément
  - de vérifier la présence d'un attribut

Rappels

- Un XML-Schéma permet de :
  - définir les éléments fils d'un éléments
  - poser des restrictions sur les types simples
  - utiliser des types simples prédéfinis
  - définir la cardinalité de l'élément racine

Rappels

- Un XML-Schéma est :
  - du XML
  - lui-même conforme à un XML-Schéma
  - une définition d'éléments utilisables dans un espace de nommage cible
  - ne permet pas d'utiliser des éléments définis dans un autre schéma

Validation XML (suite)

W3C VALIDATOR Suite

http://validator.w3.org/

DTD

- DTD = Document Type Definition

- La DTD fournit :
  - la liste des éléments
  - la liste des attributs
  - des notations
  - des entités du document XML associé
  - les règles des relations qui les régissent

DTD interne et externe

- La DTD est déclarée dans le document XML par la balise !DOCTYPE
- Elle peut être :
  - incluse dans le code source du fichier XML, ou DTD **interne** :

```
<!DOCTYPE élément-racine [déclaration des éléments]>
```

  - décrite dans un fichier externe, ou DTD **externe** :

```
<!DOCTYPE élément-racine PUBLIC "SYSTEM" "nom_fichier.dtd">
```

DTD «Building blocks »

- Une DTD contient
  - Une ou plusieurs définitions d'éléments introduites par la balise !ELEMENT

```
<!ELEMENT nom-élément valeur>
```

  - Une ou plusieurs listes d'attributs introduites par la balise !ATTLIST:

```
<!ATTLIST nom-élément attribut type défaut>
```

  - Une ou plusieurs définitions d'entités introduites par la balise !ENTITY :

```
<!ENTITY nom-entité "valeur">
```

ou

```
<!ENTITY nom-entité SYSTEM "nom_fichier">
```

DTD : Eléments

- Combinaison d'un ou plusieurs items :
  - Mot-clé #PCDATA (Parsed Character Data) : le contenu est une chaîne de caractères
  - Nom d'un autre élément de la DTD
- Chaque item peut être suivi par un caractère facultatif d'occurrence :
  - ? : l'item apparaît zéro ou au plus une fois
  - + : l'item apparaît au moins 1 fois
  - * : l'item apparaît zéro fois ou plus

DTD : Eléments

- Les items sont séparés par :
  - Le caractère virgule « , » si les 2 items sont tous deux inclus dans le contenu
  - Le caractère pipe « | » si le contenu est un choix entre les 2 items
- Les items peuvent être regroupés à l'aide de parenthèses pour définir des ordres de priorité ou d'occurrence

DTD : Eléments

- Exemples

`<!ELEMENT livre (chapitre+)>`	Définition d'un élément livre composé de un ou plus éléments chapitre.
`<!ELEMENT chapitre (titre?, corps)>`	Définition d'un élément chapitre, composé de deux éléments : titre (facultatif), corps
`<!ELEMENT titre (#PCDATA)>`	Définition d'un élément titre contenant une chaîne de caractères
`<!ELEMENT image EMPTY>`	Définition d'un élément image de contenu vide

DTD : Attributs

- Déclaration dans la DTD par la balise !ATTLIST :

```
<!ATTLIST nom-élément nom-attribut type valeur-défaut>
```

Où :

- nom-élément est le nom d'un élément défini dans la DTD par la balise !ELEMENT
- attribut est le nom de l'attribut présentement défini pour l'élément nom-élément
- type peut prendre l'une des valeurs:
  - CDATA : la valeur correspond au contenu de l'élément
  - (choix1|choix2|…) : la valeur doit être l'une des choix énumérés
  - ID : la valeur est un identifiant unique
  - ENTITY : la valeur est une entité
- valeur-défaut prend l'une des valeurs :
  - valeur : valeur par défaut donnée à l'attribut
  - #REQUIRED : attribut obligatoire, sans valeur par défaut
  - #IMPLIED : attribut optionnel, sans valeur par défaut
  - #FIXED valeur : valeur fixe

DTD : Attributs

- Exemple

`<!ATTLIST chapitre` `num CDATA #REQUIRED>`	Définition d'un attribut num (obligatoire) pour l'élément chapitre

DTD : Entities

- Définition d'une entité XML
  - Une entité est une variable utilisée pour définir du texte
  - L'intérêt d'une entité consiste à pouvoir remplacer autant de fois que nécessaire dans le document XML l'entité par le texte qui lui est associé
  - Dans le document XML, la référence à une entité est introduite par le caractère "&" suivi du nom de l'entité, et terminée par le caractère ";". Il existe 5 entités prédéfinies en XML : lt, gt, amp, apos et quot (caractères < > & ' ")
- Une entité est définie avec la balise !ENTITY, et dite:
  - Interne si sa valeur est donnée dans la DTD

```
<!ENTITY nom-entité "valeur">
```

  - Externe si sa valeur est fournie dans un fichier externe à la DTD

```
<!ENTITY nom-entité SYSTEM "nom_fichier">
```

Un exemple de DTD

```
<!ELEMENT livre (titre, auteur+, chapitre+)>
<!ELEMENT auteur (nom, prenom)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT chapitre (titre, paragraphe+)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT paragraphe (titre, corps)>
<!ELEMENT corps (#PCDATA)>
<!ATTLIST chapitre
   num CDATA #REQUIRED>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE livre PUBLIC "SYSTEM" "livre.dtd">
<livre>

   <titre>Apprendre le XML</titre>

   <auteur>
      <nom>Erik .T</nom>
      <prenom>Ray</prenom>
   </auteur>

   <!-- Premier chapitre du livre -->
   <chapitre num="1">
      <titre>Chapitre 1</titre>
      <paragraphe>
         <titre>Premier paragraphe</titre>
         <corps>Contenu du premier paragraphe...</corps>
      </paragraphe>
      <paragraphe>
         <titre>Deuxième paragraphe</titre>
         <corps>Contenu du deuxième paragraphe...</corps>
      </paragraphe>
   </chapitre>
</livre>
```

Limites des DTD

- Une DTD est difficile à lire
- Une DTD est non extensible (ce n'est pas un document XML).
- Une DTD ne permet pas de typer les données
- Une DTD ne peut prendre en compte qu'un seul espace de nommage (namespace).

DTD vs XSD

- La DTD permet de définir facilement et rapidement des grammaires simples.
- XML-Schema permet de définir de manière plus formelle et complète une grammaire => complexité accrue.
- Un document XML-Schema respecte la syntaxe XML.
- Un document XML-Schema est généralement plus volumineux et encore plus difficile à lire qu'une DTD (pour un opérateur humain).

Relax NG

- Standard OASIS
- 2 syntaxes : 1 XML, 1 compacte non XML
- « Easy to learn, easy to write »

Relax NG : « Building blocks »

- *named pattern :* `<grammar>`, `<start>`, `<define>`, `<ref>`

- `<element>`, `<attribute>`
- `<text/>`
- `<choice><value></value><choice>`
- `<zeroOrMore>`, `<oneOrMore>`,`<optional>`

Relax NG : Exemple

```
<addressBook>
  <card>
    <name>John Smith</name>
    <email>js@example.com</email>
  </card>
  <card>
    <name>Fred Bloggs</name>
    <email>fb@example.net</email>
  </card>
</addressBook>
```

```
<element name="addressBook"
         xmlns="http://relaxng.org/ns/structure/1.0">
  <oneOrMore>
    <element name="card">
      <element name="name">
        <text/>
      </element>
      <element name="email">
        <text/>
      </element>
    </element>
  </oneOrMore>
</element>
```

Relax NG *named pattern*: Exemple

```
<livre>
   <chapitre>Premier chapitre</chapitre>
   <chapitre>Second chapitre</chapitre>
</livre>
```

```
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
   <start>
      <element name="livre">
         <oneOrMore>
            <ref name="chapitre"/>
         </oneOrMore>
      </element>
   </start>
   <define name="chapitre">
      <element name="chapitre">
         <text/>
      </element>
   </define>
</grammar>
```

Schematron

- Standard ISO
- Est du XML
- Pas une grammaire mais un langage de règles

Building Blocks

- <schema xmlns="http://purl.oclc.org/dsdl/schematron">

  optional <title> then

  zero or more <ns prefix="toto" uri="http://toto" /> giving the namespaces and prefixes used for XPath, then

  serveral <rule context="xslt_expression"> where the context attribute is an XSLT expression, which contain mixed

  <assert test="xpath_expression"> where the test attribute is an XPATH location, and which contains rich text expressing the statement being asserted in plain language, and

  <report test="xpath_expression"> where test attribute is an XPATH location, and which contains rich text expressing the fact to be reported in plain language

Schematron : Exemples

```
<schema xmlns="http://www.ascc.net/xml/schematron" >
     <pattern name="Print both cases">
          <rule context="AAA">
               <assert test="BBB">BBB element is missing.</assert>
               <report test="BBB">BBB element is present.</report>
               <assert test="@name">AAA misses attribute name.</assert>
               <report test="@name">AAA contains attribute name.</report>
          </rule>
     </pattern>
     <pattern name="Print positive result only">
          <rule context="AAA">
               <report test="BBB">BBB element is present.</report>
               <report test="@name">AAA contains attribute name.</report>
          </rule>
     </pattern>
     <pattern name="Print negative result only">
          <rule context="AAA">
               <assert test="BBB">BBB element is missing.</assert>
               <assert test="@name">AAA misses attribute name.</assert>
          </rule>
     </pattern>
</schema>
```

```
<AAA>
     <BBB/>
</AAA>
```

?

```
<AAA name="toto">
     <CCC/>
</AAA>
```

?

Schematron : Exemples

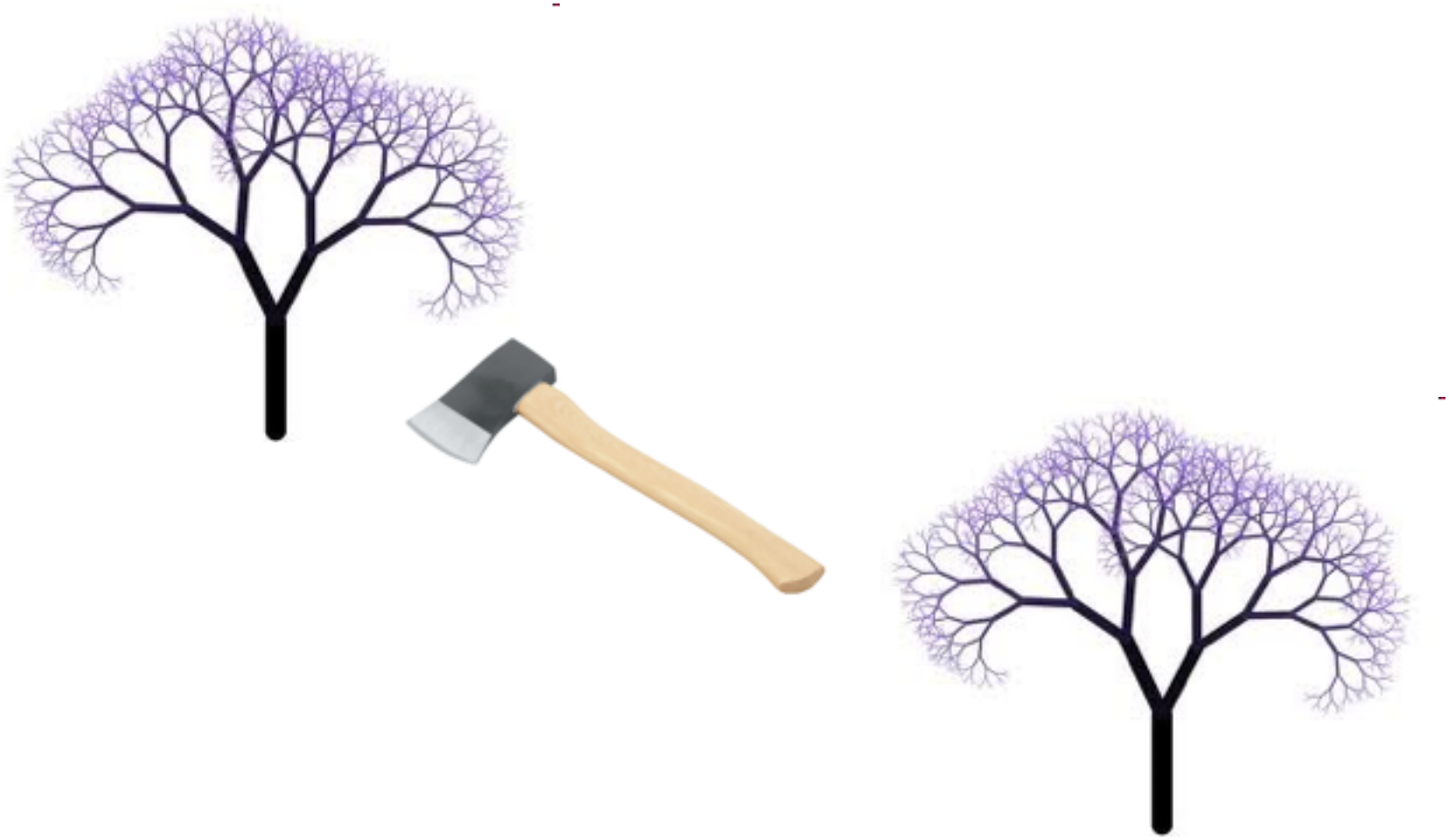
```
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
   <title>A Schematron Mini-Schema for Schematron</title>
   <ns prefix="sch" uri="http://purl.oclc.org/dsdl/schematron" />
   <pattern>
      <rule context="sch:schema">
         <assert test="sch:pattern">
            A schema contains patterns.
         </assert>
         <assert test="sch:pattern/sch:rule[@context]">
            A pattern is composed of rules.
            These rules should have context attributes.
         </assert>
         <assert test="sch:pattern/sch:rule/sch:assert[@test] or sch:pattern/
sch:rule/sch:report[@test]">
            A rule is composed of assert and report statements.
            These rules should have a test attribute.
         </assert>
      </rule>
   </pattern>
</schema>
```

Conclusion

- Document XML Valide
  - Bien formé + Conforme à un(e) :
- DTD
- XML – Schema
- Relax NG
- Schematron

Avez-vous des questions ?

Transformation XML



Transformation XML

Avant la transformation …

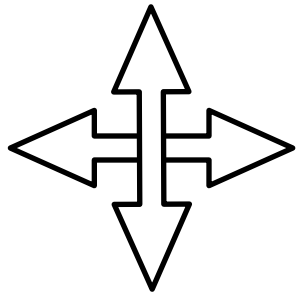
Chemins et arbres…

XPath

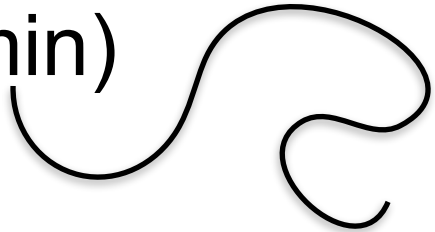
- standard W3C
- syntaxe pour désigner des parties de documents
- naviguer dans des documents au travers d'expressions
- librairie de fonction pour la navigation
- très utilisé pour les transformations XML

XPath

- **Axes** de recherche



- **Selection** de noeuds (pas / chemin)

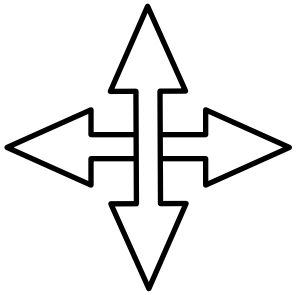


- **Filtrage** des noeuds (prédicats)

```
axis::select_exp[predicate_exp]
```

Axes

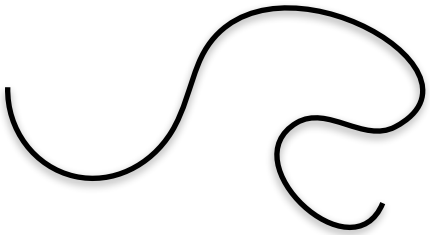
- Ensemble de noeuds
- Relatif à un noeud courant



Axe	*Ensemble*
`child`	tous les fils du noeud courant (axe par défaut)
`descendant`	***tous descendants du noeud courant***
`parent`	***le noeud parent du noeud courant***
`ancestor`	tous les ancêtres du noeud courant
`following-sibling`	***tous les noeuds frère suivants du noeud courant***
`preceding-sibling`	***tous les noeuds frère précédents du noeud***

Selection de noeuds

- Types de noeud
  - `element, attribute, text, namespace, processing-instruction, comment, document`
- Selection d'élément(s)
  - élément : `nodename`
    - Sélectionne tous les fils du noeuds spécifié
  - élément racine : `/`
    - Sélection tous les fils de la racine
  - élément n'importe où dans l'arbre : `//`
  - élément courant : `.`
  - élément parent : `..`
- Selection d'attributs
  - attribut : `@attributename`



Exemples

Expression XPath	*Résultat*
`livre`	tous les éléments ***livre***
`/livre`	l'élément racine ***livre***
`livre/chapitre`	tous les éléments ***chapitre*** qui sont fils des éléments ***livre***
`//titre`	tous les éléments ***titre*** quelque soit leur position dans l'arbre
`//@num`	tous les attributs de nom ***num***

Filtrage par prédicat

- Filtre sur les noeuds sélectionnés

  `selection_exp[predicate_exp]`

Expression XPath	*Résultat*
`/livre/chapitre[1]`	le premier **chapitre** de l'élément racine **livre**
`//chapitre[@num>3]`	tous les **chapitres** dont la valeur de l'attribut **num** est > à 3
`//chapitre[@num>3]/corps`	tous les **corps** des chapitres dont la valeur de l'attribut **num** est > à 3

XPath “jokers” (wildcard)

Expression XPath	*Résultat*
*	tous les **éléments**
@*	tous les **attributs**
node()	tous les **noeuds** de tous type

Expression XPath	*Résultat*
/livre/chapitre/*	tous les fils des chapitres
//*	tous les **éléments** du document
chapitre[@*]	chapitres possèdant n’import quel attribut

XPath : Quelques fonctions

Fonction	Résultat
`fn:concat(string* )`	Retourne la concaténation des chaînes passées en argument
`fn:contains(string, string)`	Retourne vrai si la deuxième chaîne passée en argument est contenue dans la première chaîne passée en argument
`fn:normalize-space(string)`	Retourne la chaîne après normalisation des espaces
`fn:starts-with(string, string)`	Retourne vrai si la première chaîne passée en argument commence par la deuxième chaîne passée en argument
`fn:string(object)`	Retourne la chaîne correspondante
`fn:string-length(string)`	Retourne la longueur de la chaîne passée en argument
`fn:substring(string, number, number)`	Retourne la partie de la chaîne passée en argument qui commence à la position indiquée par le 2e argument sur la longueur indiquée par le 3e argument
`fn:substring-after(string, string)`	Retourne la partie de la chaîne passée en argument qui suit la première occurrence de la 2e chaîne passée en argument
`fn:count(chemin)`	Retourne le nombre d'élément répondant au chemin

```
<functions xmlns:fn="http://www.w3.org/2005/xpath-functions">
```

Exemples

- /livre
- //corps

```
<corps>&gt; Ceci est l'introduction du chapitre 1</corps>
<corps>Ceci est le contenu de la section 1 du chapitre 1</corps>
<corps>Ceci est le contenu de la sous-section 1 de la section 1 du chapitre 1</corps>
<corps>Ceci est le contenu de la sous-section 2 de la section 1 du chapitre 1</corps>
<corps>Ceci est le contenu de la sous-section 2 de la section 1 du chapitre 1</corps>
<corps>Ceci est l'introduction du chapitre 2</corps>
<corps>Ceci est le contenu de la section 1 du chapitre 2</corps>
```

- /livre/chapitre[1]/titre/text()

```
1  Chapitre 1
```

- //*[@source="..."]

```
1  <image source="..."/>
```

- /livre/chapitre[titre="Chapitre 1" and corps]

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE livre SYSTEM "livre.dtd">
3  <livre>
4    <!-- Premier chapitre -->
5    <chapitre num="1">
6      <titre>Chapitre 1</titre>
7      <corps>&gt; Ceci est l'introduction du chapitre 1</corps>
8      <chapitre num="1">
9        <titre>Chapitre 1 - Section 1</titre>
10       <corps>Ceci est le contenu de la section 1 du chapitre 1</corps>
11       <chapitre num="1">
12         <titre>Chapitre 1 - Section 1 - Sous section 1</titre>
13         <corps>Ceci est le contenu de la sous-section 1 de la section 1 du chapitre 1</corps>
14       </chapitre>
15       <chapitre num="2">
16         <titre>Chapitre 1 - Section 1 - Sous section 2</titre>
17         <corps>Ceci est le contenu de la sous-section 2 de la section 1 du chapitre 1</corps>
18       </chapitre>
19     </chapitre>
20     <chapitre num="2">
21       <titre>Chapitre 1 - Section 2</titre>
22       <corps>Ceci est le contenu de la section 2 du chapitre 1</corps>
23     </chapitre>
24   </chapitre>
25   <!-- Deuxième chapitre -->
26   <chapitre num="2">
27     <titre>Chapitre 2</titre>
28     <corps>Ceci est l'introduction du chapitre 2</corps>
29     <chapitre num="2.1">
30       <titre>Chapitre 2- Section 1</titre>
31       <corps>Ceci est le contenu de la section 1 du chapitre 2</corps>
32       <image source="..."/>
33     </chapitre>
34   </chapitre>
35 </livre>
```

```
<chapitre num="1">
  <titre>Chapitre 1</titre>
  <corps>&gt; Ceci est l'introduction du chapitre 1</corps>
  <chapitre num="1">
    <titre>Chapitre 1 - Section 1</titre>
    <corps>Ceci est le contenu de la section 1 du chapitre 1</corps>
    <chapitre num="1">
      <titre>Chapitre 1 - Section 1 - Sous section 1</titre>
      <corps>Ceci est le contenu de la sous-section 1 de la section 1 du chapitre 1</corps>
    </chapitre>
    <chapitre num="2">
      <titre>Chapitre 1 - Section 1 - Sous section 2</titre>
      <corps>Ceci est le contenu de la sous-section 2 de la section 1 du chapitre 1</corps>
    </chapitre>
  </chapitre>
  <chapitre num="2">
    <titre>Chapitre 1 - Section 1 - Sous section 2</titre>
    <corps>Ceci est le contenu de la sous-section 2 de la section 1 du chapitre 1</corps>
  </chapitre>
</chapitre>
```

À VOUS DE JOUER!

Exercice : XPath

- Tous les *cinemas* du catalogue
- Toutes les *séances* de l'UGC
- Toutes les heures de séances
- Tous les titres de films

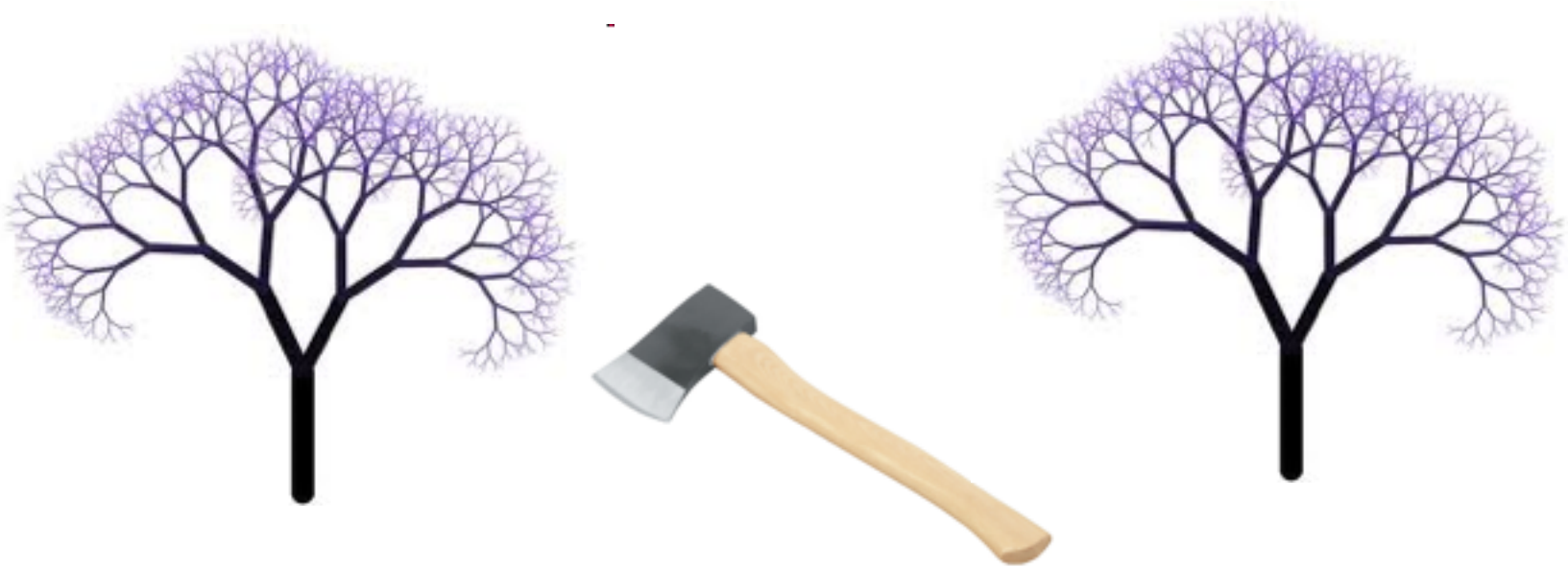
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<catalogue>
  <cinemas>
    <cinema>
      <nom>UGC</nom>
      <adresse>
        <ville>Rouen</ville>
        <lieu>Saint Sever</lieu>
      </adresse>
      <seance heure="16h" ref_film="1"/>
      <seance heure="20h" ref_film="1"/>
      <seance heure="20h" ref_film="3"/>
      <seance heure="20h" ref_film="4"/>
      <seance heure="20h" ref_film="5"/>
    </cinema>
    <cinema>
      <nom>Pathe</nom>
      <adresse>
        <ville>Rouen</ville>
        <lieu>Dock de Rouen</lieu>
      </adresse>
      <seance heure="18h" ref_film="1"/>
      <seance heure="20h" ref_film="1"/>
      <seance heure="21h" ref_film="2"/>
      <seance heure="22h" ref_film="4"/>
      <seance heure="20h" ref_film="5"/>
      <seance heure="23h" ref_film="5"/>
    </cinema>
  </cinemas>
  <films>
    <film film_id="1">
      <titre>Titeuf, le film</titre>
      <année>2011</année>
    </film>
    <film film_id="2">
      <titre>Scream 4</titre>
      <année>2011</année>
    </film>
    <film film_id="3">
      <titre>Source Code</titre>
      <année>2011</année>
    </film>
    <film film_id="4">
      <titre>Rango</titre>
      <année>2011</année>
    </film>
    <film film_id="5">
      <titre>Sucker Punch</titre>
      <année>2011</année>
    </film>
  </films>
</catalogue>
```

Corrigé : XPath

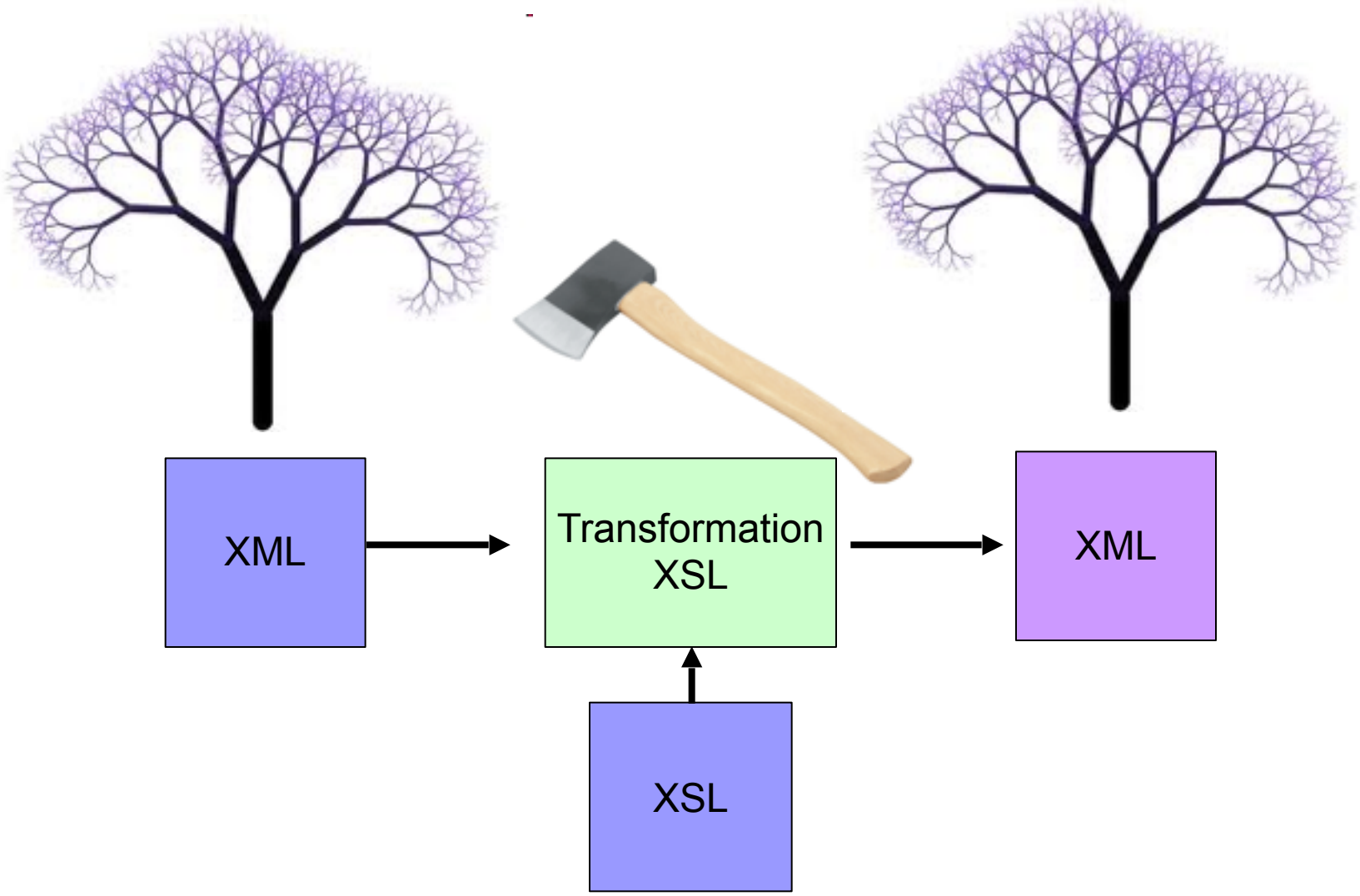
- Tous les *cinemas* du catalogue
  - /catalogue/cinemas/*
- Toutes les *séances* de l'UGC
  - //cinema[nom="UGC"]/seance
- Toutes les *heures* de séances
  - //@heure
- Tous les titres de films
  - //titre/text()

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<catalogue>
 <cinemas>
  <cinema>
   <nom>UGC</nom>
   <adresse>
    <ville>Rouen</ville>
    <lieu>Saint Sever</lieu>
   </adresse>
   <seance heure="16h" ref_film="1"/>
   <seance heure="20h" ref_film="1"/>
   <seance heure="20h" ref_film="3"/>
   <seance heure="20h" ref_film="4"/>
   <seance heure="20h" ref_film="5"/>
  </cinema>
  <cinema>
   <nom>Pathe</nom>
   <adresse>
    <ville>Rouen</ville>
    <lieu>Dock de Rouen</lieu>
   </adresse>
   <seance heure="18h" ref_film="1"/>
   <seance heure="20h" ref_film="1"/>
   <seance heure="21h" ref_film="2"/>
   <seance heure="22h" ref_film="4"/>
   <seance heure="20h" ref_film="5"/>
   <seance heure="23h" ref_film="5"/>
  </cinema>
 </cinemas>
 <films>
  <film film_id="1">
   <titre>Titeuf, le film</titre>
   <année>2011</année>
  </film>
  <film film_id="2">
   <titre>Scream 4</titre>
   <année>2011</année>
  </film>
  <film film_id="3">
   <titre>Source Code</titre>
   <année>2011</année>
  </film>
  <film film_id="4">
   <titre>Rango</titre>
   <année>2011</année>
  </film>
  <film film_id="5">
   <titre>Sucker Punch</titre>
   <année>2011</année>
  </film>
 </films>
</catalogue>
```

Transformation XSL



Transformation XSL



XSL : Exemple

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

<livre xmlns="http://livre.org"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://livre.org livre.xsd">

  <titre>Apprendre le XML</titre>

  <auteur>
    <nom>Bondu</nom>
    <prenom>Emilien</prenom>
  </auteur>

  <!-- Premier chapitre du livre -->
  <chapitre num="1">
    <titre>Les Bases d'XML</titre>
    <paragraphe>
      <titre>Premier paragraphe</titre>
      <corps>Contenu du premier paragraphe...</corps>
    </paragraphe>
    <paragraphe>
      <titre>Deuxième paragraphe</titre>
      <corps>Contenu du deuxième paragraphe...</corps>
    </paragraphe>
  </chapitre>


  <!-- Premier chapitre du livre -->
  <chapitre num="2">
    <titre>La transformation XML</titre>
    <paragraphe>
      <titre>Premier paragraphe</titre>
      <corps>Contenu du premier paragraphe...</corps>
    </paragraphe>
  </chapitre>
</livre>
```

```xml
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Book : Apprendre le XML</title>
</head>
<body>
  A book by Bondu Emilien
  <table>
    <tr>
      <th>Chapter</th>
      <th>Title</th>
      <th>Nb paragraph</th>
    </tr>
    <tr>
      <td>1</td>
      <td>Les Bases d'XML</td>
      <td>2</td>
    </tr>
    <tr>
      <td>2</td>
      <td>La transformation XML</td>
      <td>1</td>
    </tr>
  </table>
</body>
</html>
```

Structure d'un XSL

- Racine du document XSL

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

- Définition de la sortie

```
<xsl:output method="xml | html | text | name"
doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"
doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"
omit-xml-declaration="yes"
encoding="UTF-8"
indent="yes" />
```

- Import et inclusion d'autres XSL
  - différence de priorité si conflit de règle

`<xsl:include href="URI"/>` (priorité de la feuille inclue = priorité de la feuille)

`<xsl:import href="URI"/>` (priorité de la feuille importée < priorité de la feuille)

- Gestion des espaces

`<xsl:strip-space elements="titre prenom" />` (échappement pour les éléments définis)

`<xsl:preserve-space elements="corps"/>` (conservation pour les éléments définis)

Règle de transformation

- Une feuille XSL se compose d'un ensemble de règles (template) permettant de construire le résultat.
- L'application d'une règle produit un fragment du document résultat.
- L'ordre des règles n'a pas d'importance.
- Une règle s'applique dans le contexte d'un nœud de l'arbre (défini par une expression XPath).
- Le corps d'une règle se compose d'un ensemble d'instructions.

```
<xsl:template match="xpath" name="qname">
    <!-- instructions -->

</xsl:template>
```

➡ L'attribut match est optionnel si l'attribut name est défini

Appel des règles

- Appel explicite d'une règle nommée
  - le noeud courant est passé à la règle appelée (pas de changement de contexte)

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

    <!-- appel du template pour le noeud courant -->
  <xsl:call-template name="createTable"/>

  </xsl:template>

 <xsl:template name="createTable">

   </xsl:template>
```

Appel des règles

- Appel de toutes les règles
  - s'applique aux sous-noeud (filtrés ou non) du noeud courant (changement de contexte)
  - seules les règles compatibles sont appliquées
  - la règle la plus appropriée (la plus spécifique) est appliquée

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

       <!-- application de toutes les règles aux fils (filtrés
si expression xpath de sélection) du noeud courant -->
    <xsl:apply-templates select="xpath"/>
     </xsl:template>

  <xsl:template match="titre">

     </xsl:template>
```

Appel des règles

- Exemple de gestion de priorité

```
<xml>
  <toto />
  <titi />
  <tata />
</xml>

<!-- sample XSLT snippet -->
<xsl:template match="xml">
  <xsl:apply-templates select="*" /> <!-- trois noeuds sélectionnés -->
</xsl:template>

<xsl:template match="toto"> <!-- sera appelé une fois -->

</xsl:template>

<xsl:template match="*"> <!-- sera appelé deux fois -->

</xsl:template>
```

Instructions

- Instructions à effectuer dans la sortie
  - Création d'un élément

```
<xsl:element name="nom_element">contenu</xsl:element>
```

  - Création d'un attribut

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

  - Création de texte

```
<xsl:text>output text</xsl:text>
```

  - Création d'instructions de traitement

```
<xsl:processing-instruction name="nom_instruction">
 valeur
</xsl:processing-instruction>
```

  - Création de commentaires

```
<xsl:comment>commentaire</xsl:comment>
```

Instructions

- Instructions à effectuer dans la sortie
  - Selection d'une valeur

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

  - Copie d'un noeud

```
<xsl:copy-of select="xpath"/>
```

  - Condition simple

```
<xsl:if test="xpath"> instructions </xsl:if>
```

  - Condition multiple

```
<xsl:choose>
  <xsl:when test="xpath"> instructions </xsl:when>
  <xsl:when test="xpath"> instructions </xsl:when>
  <xsl:otherwise> instructions </xsl:otherwise>
</xsl:choose>
```

Instructions

- Instructions à effectuer dans la sortie
  - Itération

```
<xsl:for-each select="xpath"> instructions </xsl:for-each>
```

  - Trie lors de l'itération

```
<xsl:for-each select="xpath">
    <xsl:sort data-type="text" order="ascending"/>
  </xsl:for-each>
```

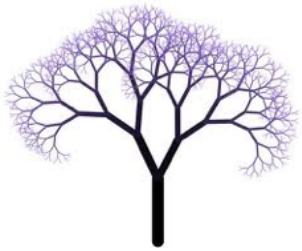
Exemple

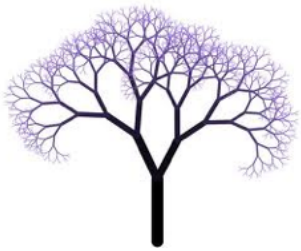
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:book="http://livre.org" version="1.0">

    <xsl:output method="html" doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN" doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"
        omit-xml-declaration="yes" encoding="UTF-8" indent="yes"/>

    <xsl:template match="/book:livre">
      <xsl:element name="html">
        <xsl:element name="head">
          <xsl:element name="title"><xsl:text>Book : </xsl:text>
            <xsl:value-of select="book:titre"/>
          </xsl:element>
        </xsl:element>
        <xsl:element name="body">
          <xsl:text>A book by </xsl:text>
          <xsl:value-of select="book:auteur/book:nom"/><xsl:text> </xsl:text><xsl:value-of select="book:auteur/book:prenom"/>
          <xsl:call-template name="createTable"/>
        </xsl:element>
      </xsl:element>
    </xsl:template>

    <xsl:template name="createTable">
     <xsl:if test="count(book:chapitre) > 0">
      <xsl:element name="table">
        <xsl:element name="tr">
          <xsl:element name="th"><xsl:text>Chapter</xsl:text></xsl:element>
          <xsl:element name="th"><xsl:text>Title</xsl:text></xsl:element>
          <xsl:element name="th"><xsl:text>Nb paragraph</xsl:text></xsl:element>
        </xsl:element>
        <xsl:for-each select="book:chapitre">
          <xsl:element name="tr">
            <xsl:element name="td"><xsl:value-of select="@num"/></xsl:element>
            <xsl:element name="td"><xsl:value-of select="book:titre"/></xsl:element>
            <xsl:element name="td"><xsl:value-of select="count(book:paragraphe)"/></xsl:element>
          </xsl:element>
        </xsl:for-each>
      </xsl:element>
     </xsl:if>
    </xsl:template>

</xsl:stylesheet>
```

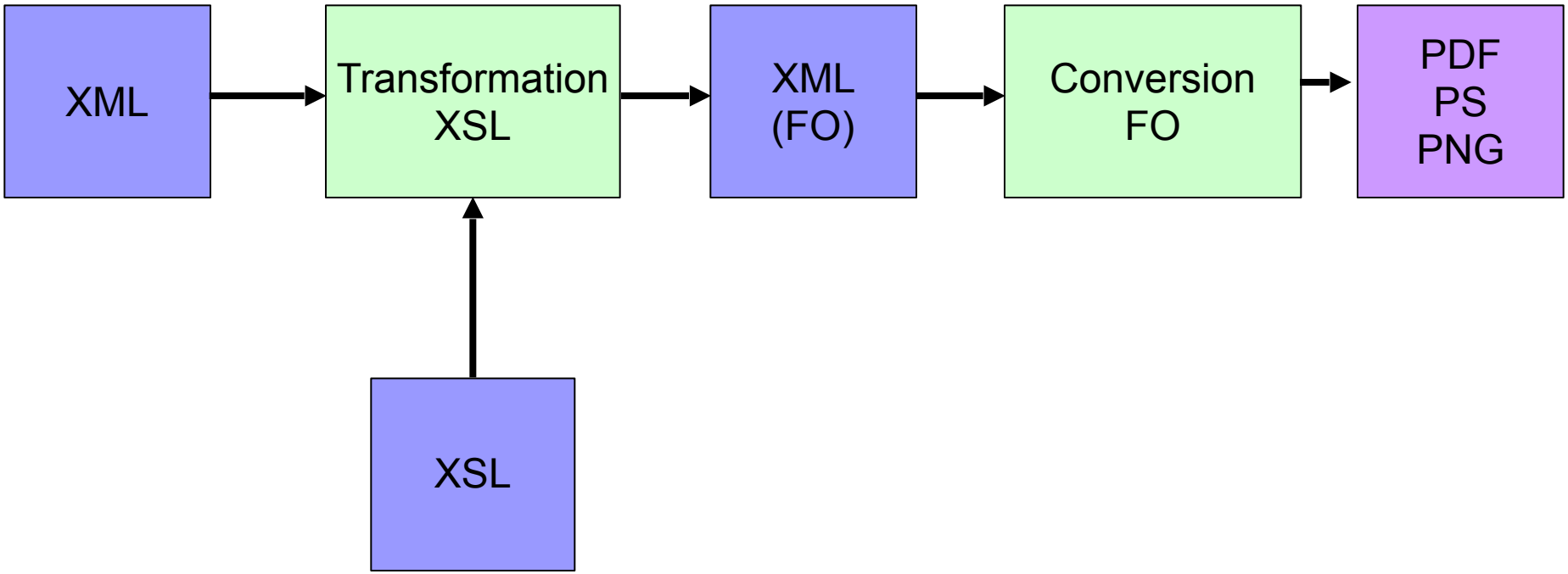
Exemple

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

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

<!-- <!DOCTYPE livre PUBLIC "SYSTEM" "livre.dtd"> -->
<livre xmlns="http://livre.org"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://livre.org livre.xsd">

  <titre>Apprendre le XML</titre>

  <auteur>
    <nom>Bondu</nom>
    <prenom>Emilien</prenom>
  </auteur>

  <!-- Premier chapitre du livre -->
  <chapitre num="1">
    <titre>Les bases d'XML</titre>
    <paragraphe>
      <titre>Premier paragraphe</titre>
      <corps>Contenu du premier paragraphe...</corps>
    </paragraphe>
    <paragraphe>
      <titre>Deuxième paragraphe</titre>
      <corps>Contenu du deuxième paragraphe...</corps>
    </paragraphe>
  </chapitre>


  <!-- Premier chapitre du livre -->
  <chapitre num="2">
    <titre>La transformation XML</titre>
    <paragraphe>
      <titre>Premier paragraphe</titre>
      <corps>Contenu du premier paragraphe...</corps>
    </paragraphe>
  </chapitre>
</livre>
```

Transformation XSL-FO







Parseur FOP

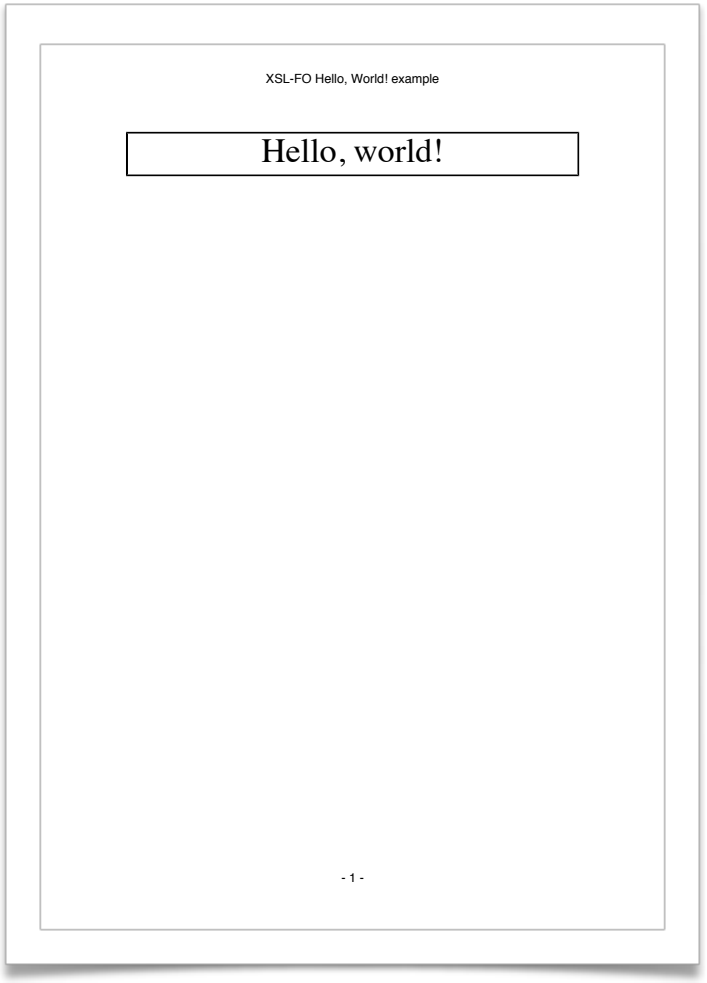
- Apache FOP

- Glassfish VelFOP

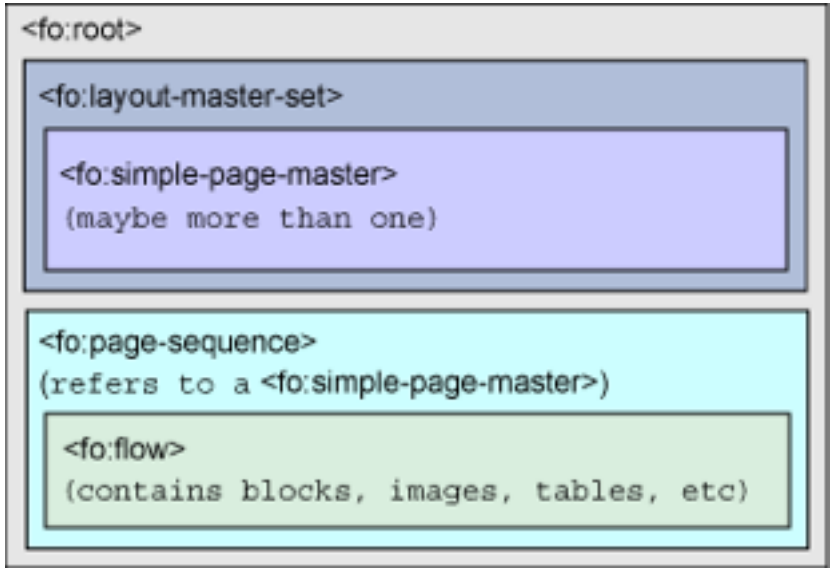
- AntillesXML FOP

XSL-FO Exemple

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!-- Hello, World!  en XSL-FO -->
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <!-- Modèle de pages -->
  <fo:layout-master-set>
    <fo:simple-page-master master-name="A4"
      page-width="210mm" page-height="297mm"
      margin="1cm">
      <!-- Région principale -->
      <fo:region-body   margin="2cm"/>
      <!-- Tête de page aka header -->
      <fo:region-before extent="1cm"/>
      <!-- Pied de page aka footer -->
      <fo:region-after  extent="1cm"/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <!-- Contenus -->
  <fo:page-sequence master-reference="A4">
    <!-- Contenu de la tête de page -->
    <fo:static-content flow-name="xsl-region-before">
      <fo:block text-align="center">XSL-FO Hello, World! example</fo:block>
    </fo:static-content>
    <!-- Contenu du pied de page : numéro de la page -->
    <fo:static-content flow-name="xsl-region-after">
      <fo:block text-align="center">- <fo:page-number/> -</fo:block>
    </fo:static-content>
    <!-- Contenu de la partie centrale -->
    <fo:flow flow-name="xsl-region-body">
      <fo:block text-align="center"
    font="32pt Times"
    border="black solid thick">Hello, world!</fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```



Structure document XSL-FO




```
<?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
<fo:layout-master-set>
  <fo:simple-page-master master-name="A4">
    <!-- Page template goes here -->
  </fo:simple-page-master>
</fo:layout-master-set>
<fo:page-sequence master-reference="A4">
  <!-- Page content goes here ->
  <fo:flow>

  </fo:flow>
</fo:page-sequence>
</fo:root>
```

XSL-FO Building Blocks

- `<fo:root>` racine XSL-FO contient un master-set et des page-sequence
- `<fo:layout-master-set>` contient une ou plusieurs templates de pages
- `<fo:simple-page-master>` définition d'un template de page
  - A un nom unique master-name
  - Défini les dimensions et marges de la page

```
<fo:simple-page-master
    master-name="main"
    margin-top="36pt"
    margin-bottom="36pt"
    page-width="8.5in"
    page-height="11in"
    margin-left="72pt"
    margin-right="72pt">

  <fo:region-body margin="50pt" />

</fo:simple-page-master>
```

XSL-FO Regions

Margin Top

REGION BEFORE		
REGION START	REGION BODY	REGION END
REGION AFTER		

Margin Left

Margin Right

Margin Bottom

XSL-FO Building Blocks

- `<fo:page-sequence>` definition du contenu des pages qui :
  - référence un layout `master-reference`
  - contient un `<fo-flow>` définissant le contenu de la sortie
  - `flow-name` : `xsl-region-body, xsl-region-before, xsl-region-after, xsl-region-start, xsl-region-end`
- chaque paragraphe est donné dans un `<fo-block>`
- formatage spécifique du contenu dans les block :

```
<fo:inline font-style="italic" font-weight="bold" color="red">
```

```
<fo:page-sequence master-reference="main">
   <fo:flow flow-name="xsl-region-body">
      <fo:block>Hello</fo:block>
      <fo:block>Hello again</fo:block>
   </fo:flow>
</fo:page-sequence>
```

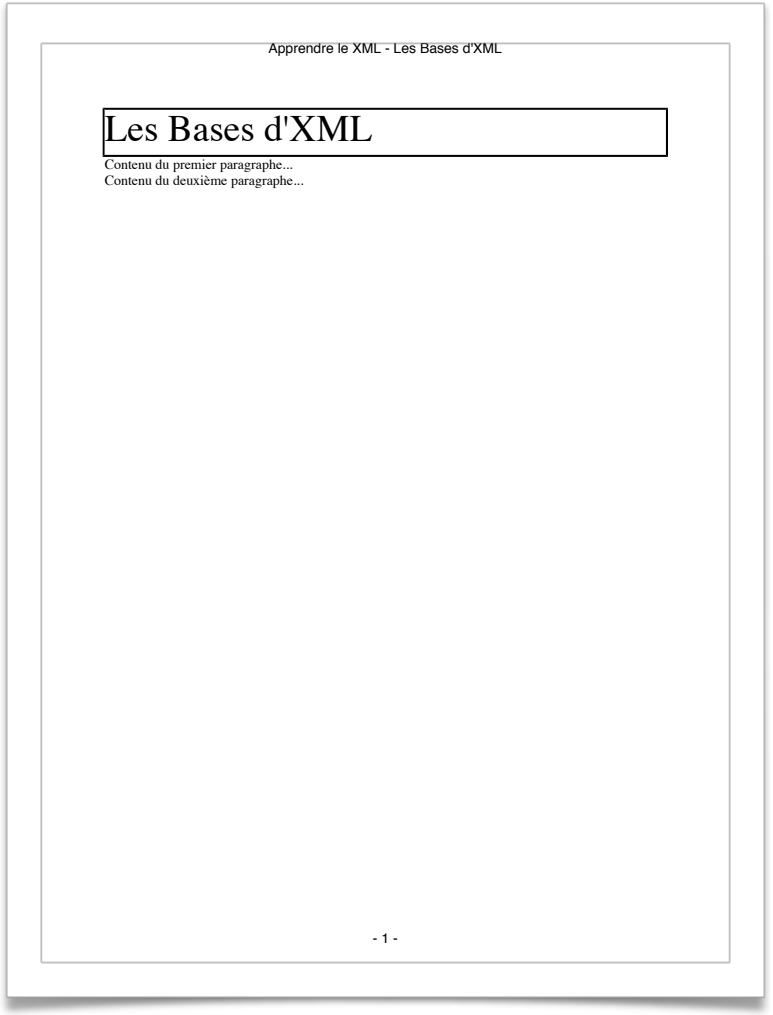
XSL-FO Building Blocks

- D’autres « objets » graphiques
  - `fo:list` pour ajouter des listes
  - `fo:table` pour ajouter des tableaux
  - `fo:basic-link` pour ajouter des liens hypertext
  - `fo:external-graphic` pour ajouter des images

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:livre="http://livre.org" xmlns:fo="http://www.w3.org/1999/XSL/Format"
                version="1.0">

    <xsl:output method="xml" encoding="UTF-8" indent="yes"/>

    <xsl:template match="/livre:livre">
      <fo:root>
        <fo:layout-master-set>
          <fo:simple-page-master master-name="A4">
             <fo:region-body margin="2cm"/>
             <fo:region-before extent="1cm"/>
             <fo:region-after  extent="1cm"/>
          </fo:simple-page-master>
        </fo:layout-master-set>

        <xsl:apply-templates select="livre:chapitre"/>
      </fo:root>
    </xsl:template>

    <xsl:template match="livre:chapitre">
    <fo:page-sequence master-reference="A4">
       <fo:static-content flow-name="xsl-region-before">
            <fo:block text-align="center">
              <xsl:value-of select="/livre:livre/livre:titre"/>
                  - <xsl:value-of select="livre:titre"/>
            </fo:block>
        </fo:static-content>
       <fo:static-content flow-name="xsl-region-after">
            <fo:block text-align="center">
                  - <fo:page-number/> -
            </fo:block>
        </fo:static-content>
        <fo:flow flow-name="xsl-region-body">
          <fo:block text-align="left" font="32pt Times"
                    border="black solid thick">
            <xsl:value-of select="livre:titre"></xsl:value-of>
          </fo:block>
          <xsl:for-each select="livre:paragraphe">
            <fo:block text-align="left" font="12pt Times">
              <xsl:value-of select="livre:corps"></xsl:value-of>
            </fo:block>
          </xsl:for-each>
        </fo:flow>
    </fo:page-sequence>
    </xsl:template>
</xsl:stylesheet>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"
  xmlns:livre="http://livre.org">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="A4">
      <fo:region-body margin="2cm" />
      <fo:region-before extent="1cm" />
      <fo:region-after extent="1cm" />
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="A4">
    <fo:static-content flow-name="xsl-region-before">
      <fo:block text-align="center">Apprendre le XML - Les Bases d'XML</fo:block>
    </fo:static-content>
    <fo:static-content flow-name="xsl-region-after">
      <fo:block text-align="center">- <fo:page-number /> -</fo:block>
    </fo:static-content>
    <fo:flow flow-name="xsl-region-body">
      <fo:block text-align="left" font="32pt Times"
        border="black solid thick">Les Bases d'XML</fo:block>
      <fo:block text-align="left" font="12pt Times">Contenu du premier paragraphe...</fo:block>
      <fo:block text-align="left" font="12pt Times">Contenu du deuxième paragraphe...</fo:block>
    </fo:flow>
  </fo:page-sequence>
  <fo:page-sequence master-reference="A4">
    <fo:static-content flow-name="xsl-region-before">
      <fo:block text-align="center">Apprendre le XML - La transformation XML</fo:block>
    </fo:static-content>
    <fo:static-content flow-name="xsl-region-after">
      <fo:block text-align="center">- <fo:page-number /> -</fo:block>
    </fo:static-content>
    <fo:flow flow-name="xsl-region-body">
      <fo:block text-align="left" font="32pt Times"
        border="black solid thick">La transformation XML</fo:block>
      <fo:block text-align="left" font="12pt Times">Contenu du premier paragraphe...</fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"
  xmlns:livre="http://livre.org">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="A4">
      <fo:region-body margin="2cm" />
      <fo:region-before extent="1cm" />
      <fo:region-after extent="1cm" />
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="A4">
    <fo:static-content flow-name="xsl-region-before">
      <fo:block text-align="center">Apprendre le XML - Les Bases d'XML</fo:block>
    </fo:static-content>
    <fo:static-content flow-name="xsl-region-after">
      <fo:block text-align="center">- <fo:page-number /> -</fo:block>
    </fo:static-content>
    <fo:flow flow-name="xsl-region-body">
      <fo:block text-align="left" font="32pt Times"
        border="black solid thick">Les Bases d'XML</fo:block>
      <fo:block text-align="left" font="12pt Times">Contenu du premier paragraphe...</fo:block>
      <fo:block text-align="left" font="12pt Times">Contenu du deuxième paragraphe...</fo:block>
    </fo:flow>
  </fo:page-sequence>
  <fo:page-sequence master-reference="A4">
    <fo:static-content flow-name="xsl-region-before">
      <fo:block text-align="center">Apprendre le XML - La transformation XML</fo:block>
    </fo:static-content>
    <fo:static-content flow-name="xsl-region-after">
      <fo:block text-align="center">- <fo:page-number /> -</fo:block>
    </fo:static-content>
    <fo:flow flow-name="xsl-region-body">
      <fo:block text-align="left" font="32pt Times"
        border="black solid thick">La transformation XML</fo:block>
      <fo:block text-align="left" font="12pt Times">Contenu du premier paragraphe...</fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```



Conclusion

- Sélection par chemin dans un XML
  - XPath
- Transformation en XML
  - XSL, XSL-FO

Références

- Cours de référence
  - Jenny Benois-Pineau, Georges Gardarin, Yves Bekkers, Khaled Khelif
- Livres
  - XML in a Nutshell, Third Edition Ed O'Reilly
- En ligne
  - http://www.w3.org/TR/xml/
  - http://www.w3schools.com/xml/default.asp