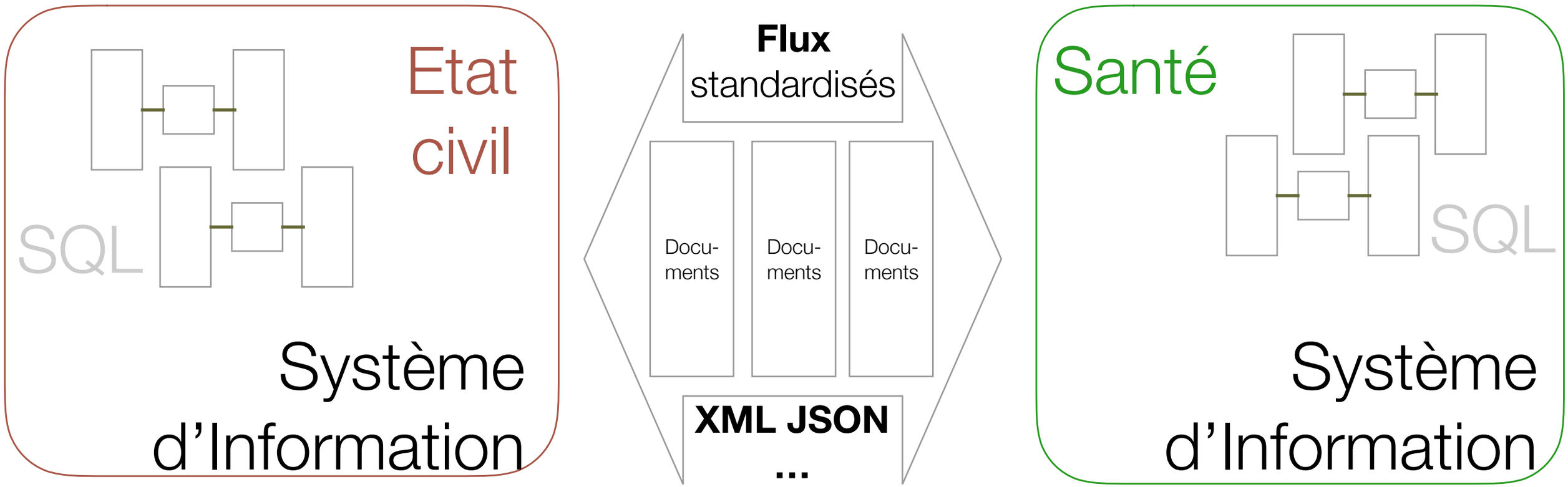
LABD

Master Info M1 2016-2017

Cours 5 : Transformer des données XML

Systèmes d'Informations, Documents - Présentation & Transformation



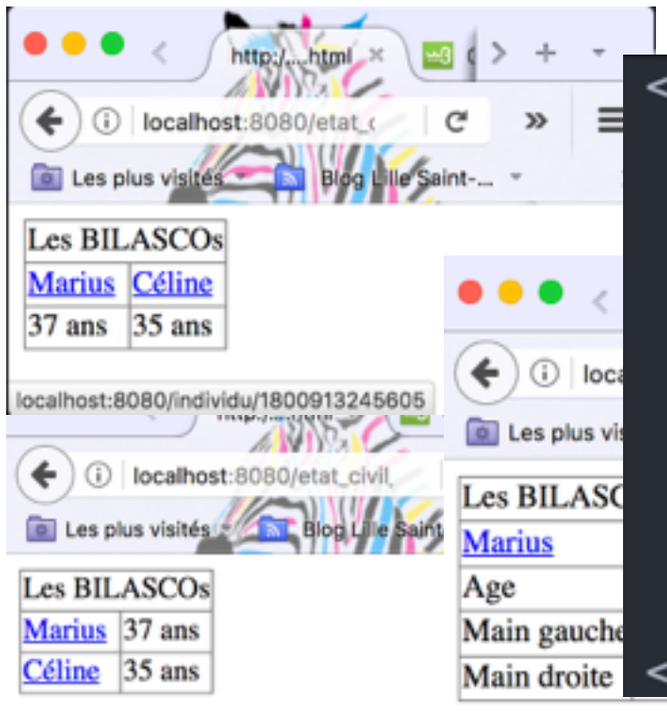

```
<famille nom="bilasco">
 <membre nom="marius"
   no_secu="1800113245605">
  <role nom="parent" rang="1">
  <age>37</age>
 </membre>
 <membre nom="celine"
   no_secu="281034340512">
  <role nom="parent" rang="2"/>
  <age>35</age>
 </membre>
</famille>
```

```
<personne
  id="1800113245605">
 <membre type="pied" rang="1">
  <etat>sain</etat>
 </membre>
 <membre type="main" rang="1">
  <etat>sain</etat>
 </membre>
 ...
</personne>
```

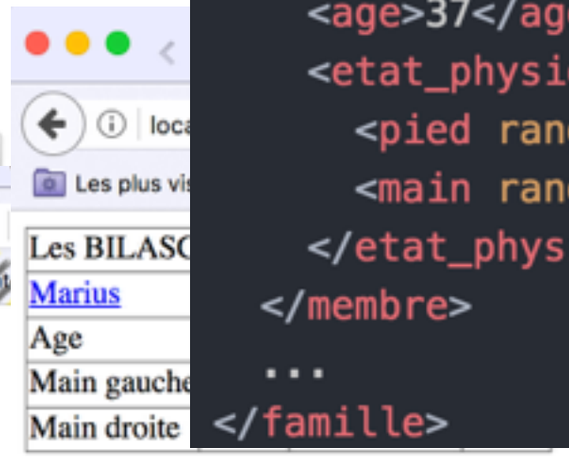
Systèmes d'Informations, Documents - Présentation & Transformation

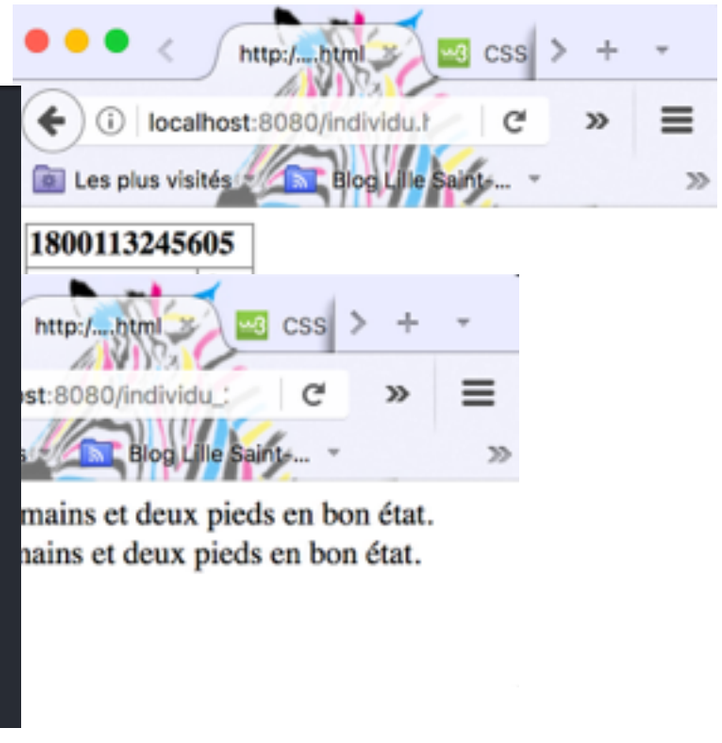
```
<famille nom="bilasco">
 <membre nom="marius"
   no_secu="1800113245605">
  <role nom="parent" rang="1">
  <age>37</age>
 </membre>
 <membre nom="celine"
   no_secu="281034340512">
  <role nom="parent" rang="2"/>
  <age>35</age>
 </membre>
</famille>
```

```
<personne
  id="1800113245605">
 <membre type="pied" rang="1">
  <etat>sain</etat>
 </membre>
 <membre type="main" rang="1">
  <etat>sain</etat>
 </membre>
 ...
</personne>
```




```
<famille nom="bilasco">
  <membre no_secu="1800913245605" nom="marius">
    <role nom="parent" rang="1">
    <age>37</age>
    <etat_physiologique>
      <pied rang="1" etat="sain"/>
      <main rang="1" etat="sain"/>
    </etat_physiologique>
  </membre>
  ...
</famille>
```





Systèmes d'Informations, Documents - Présentation & Transformation

Approche impérative

- parser les données

- construire des entités objets à partir des données

- générer les documents cibles en accédant aux valeurs des entités

Approche document

- identifier les transformations à appliquer à chaque partie du document source afin d'atteindre le document cible

- la structure du document source dirige les transformations

- le document cible est obtenu suite à l'application de l'ensemble de règles

Systèmes d'Informations, Documents - Présentation & Transformation

```
<famille nom="bilasco">
 <membre nom="marius"
   no_secu="1800113245605">
  <role nom="parent" rang="1">
  <age>37</age>
 </membre>
 <membre nom="celine"
   no_secu="281034340512">
  <role nom="parent" rang="2"/>
  <age>38</age>
 </membre>
</famille>
```

Les BILASCOs	
Marius	37 ans
Céline	35 ans

Approche impérative

A partir de données (XML ou autres sources), construire entité Famille contenant un tableau dont les éléments de type Personne contiennent un nom, un age, etc.

Remplir un patron ou générer le contenu HTML de la page

```
<table>
  <tr><td colspan="2">
      Les <?php echo strtoupper($famille["nom"]); ?>s
  </td></tr>
  <tr>
     <td>
       <a href="individu/<?php echo $famille["personne"][0]["no_secu"]; ?>">
         <?php echo maj1erLettre($famille["personne"][0]["nom"]); ?>
       </a></td>
     <td><?php echo $famille["personne"][0]["age"]; ?></td>
  </tr>
  …
</table>
```

Systèmes d'Informations, Documents - Présentation & Transformation

```
<famille nom="bilasco">
 <membre nom="marius"
   no_secu="1800113245605">
  <role nom="parent" rang="1">
  <age>37</age>
 </membre>
 <membre nom="celine"
   no_secu="281034340512">
  <role nom="parent" rang="2"/>
  <age>38</age>
 </membre>
</famille>
```

Approche document

parcourir le document source à partir de la racine

appliquer règle pour <famille nom="bilasco">

```
<table>
  <tr><td colspan="2">BILASCO</td></tr>
  Appliquer règles pour chaque "membre"
</table>
```

appliquer règle pour <membre nom="marius" no_secu="1800113245605">

```
<table>
  <tr><td colspan="2">Les BILASCOs</td></tr>
  <tr>
     <td><a href="individu/1800113245605">Marius</a></td>
     <td>37 ans</td>
  </tr>
  Appliquer règles pour les "membre" restant
</table>
```

appliquer règle pour <membre nom="celine" no_secu="281034340512">

…

Systèmes d'Informations, Documents - Présentation & Transformation

---

Approche document

- comment définir les règles de transformations ?

- comment contrôler l'ordre d'application des règles ?

- quelles structures de contrôle lors de l'application d'une règle ?

- comment accéder aux données contenues dans les éléments du document source ?

- comment créer des nouveaux éléments / nouveaux attributs ?

LABD

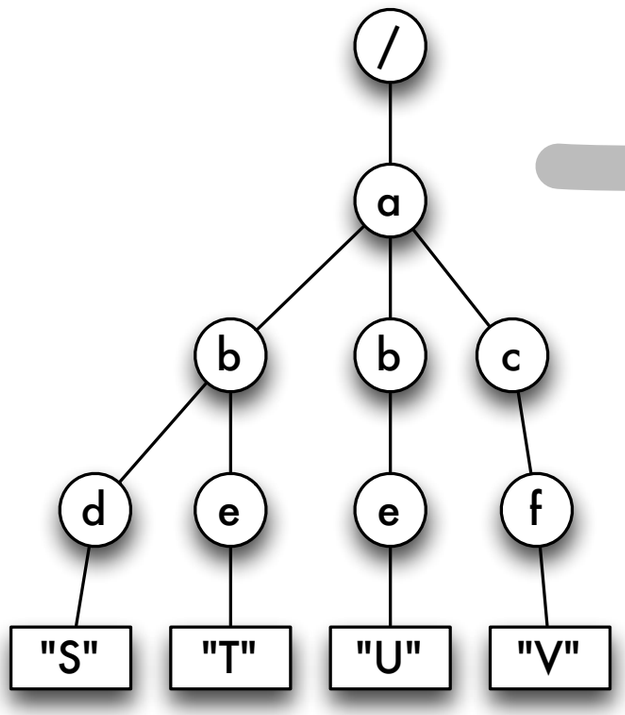
Master Info M1 2016-2017

Cours 5 : Transformer des données XML **avec XSLT**

Motivations

1. Motivation à l'origine : associer un style à un document XML
   - XSL = XML Stylesheet Language
   - Les CSS que l'on utilise pour HTML ne permettent pas d'afficher les valeurs des attributs, de transformer la structure du document, ni de créer de nouvelles données.
   - De plus CSS pour XML est un peu lourd car il n'y a pas de style par défaut comme en XHTML
   - XSL = XSL-FO (pour l'aspect formatage) + XSL-T (pour l'aspect transformation)

2. Transformation d'un document XML en un autre document XML
   - Transformation d'un arbre source en un arbre cible
   - Une transformation est donnée par un ensemble de règles
   - XSLT 1.0 utilise XPath 1.0
   - XSLT 2.0 utilise XPath 2.0



moteur XSLT

{ / }

{ b , b }

<Y>
<X>S</X>
<X></X>
</Y>

programme XSLT

Règle pour "/"

<Y>
travailler sur "**.//b**"
</Y>

Règle pour "a"

<Z>
travailler sur "**.//c**"
</Z>

Règle pour "b"

<X>
valeur de "**./d**"
</X>

Définir une feuille XSLT

- XSLT est un dialecte XML, défini par un schéma d'espace de noms

  `http://www.w3.org/1999/XSL/Transform`

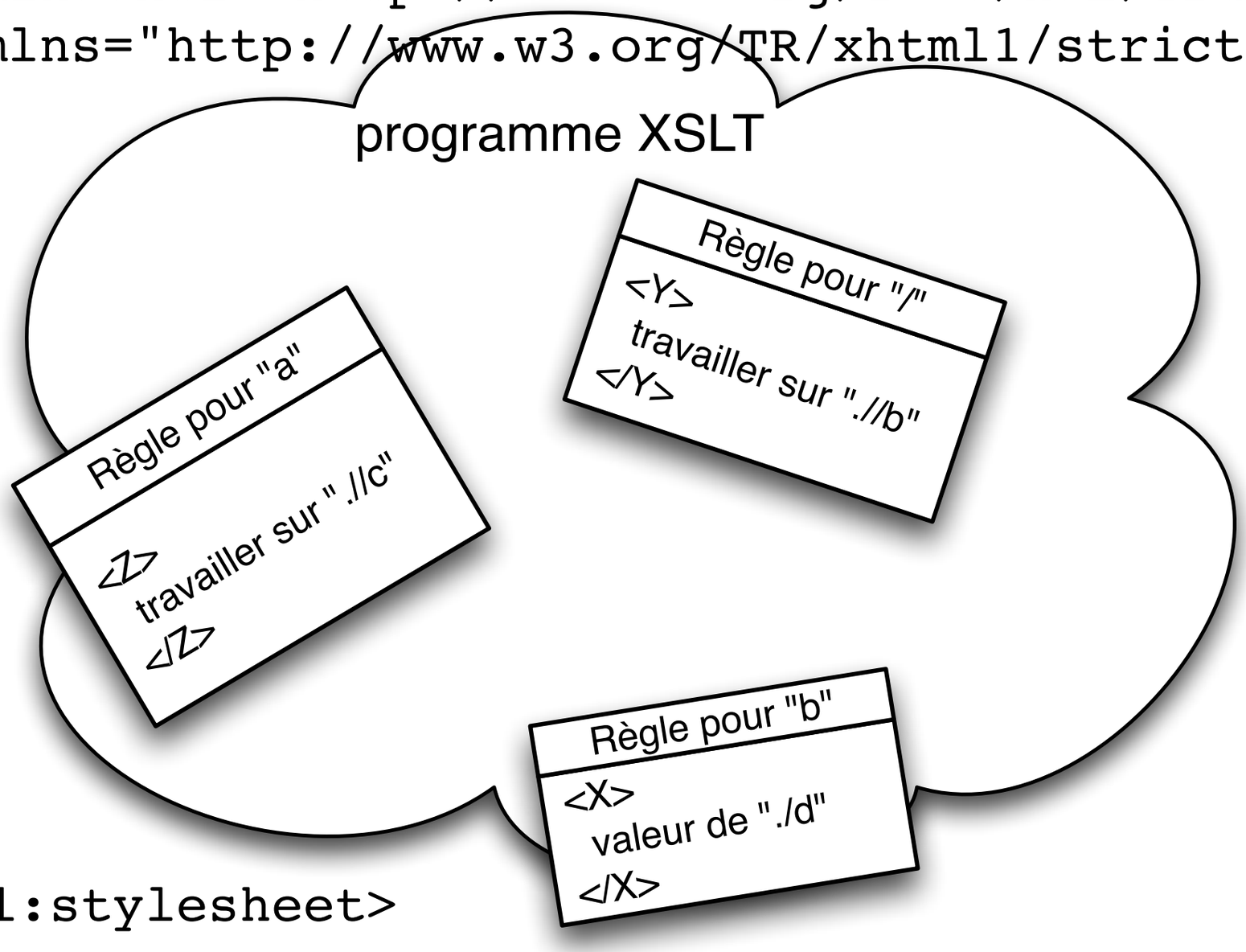
- version 1.0 qui date de novembre 99 (et version 2.0 de janvier 2007).

- La racine d'un document XSLT est un élément `xsl:stylesheet`, ou bien `xsl:transform` (synonyme).

```
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns="http://www.w3.org/TR/xhtml1/strict">
...
</xsl:stylesheet>
```

Définir une feuille XSLT

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns="http://www.w3.org/TR/xhtml1/strict">
```




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

Appliquer une feuille XSLT à un document

1. Attacher la feuille de style au document, comme on le fait pour une feuille CSS.

```
<?xml-stylesheet type="text/xsl" href="transfo1.xsl" ?>
```

2. Utiliser un programme qui applique la transformation au document pour produire un autre document
   - les navigateurs récents supportent *avec certaines restrictions* l'association des feuilles de style XSLT aux documents XML

Préciser le format de sortie

Pour préciser un format de sortie autre que XML qui est le format de sortie par défaut, on utilise l'élément `xsl:output` (c'est une instruction de traitement) en donnant une valeur à son attribut `method` parmi les valeurs `html`, `xml` ou `text`. Cet élément dispose aussi d'un attribut `indent` qu'on peut positionner à `yes` ou `no`

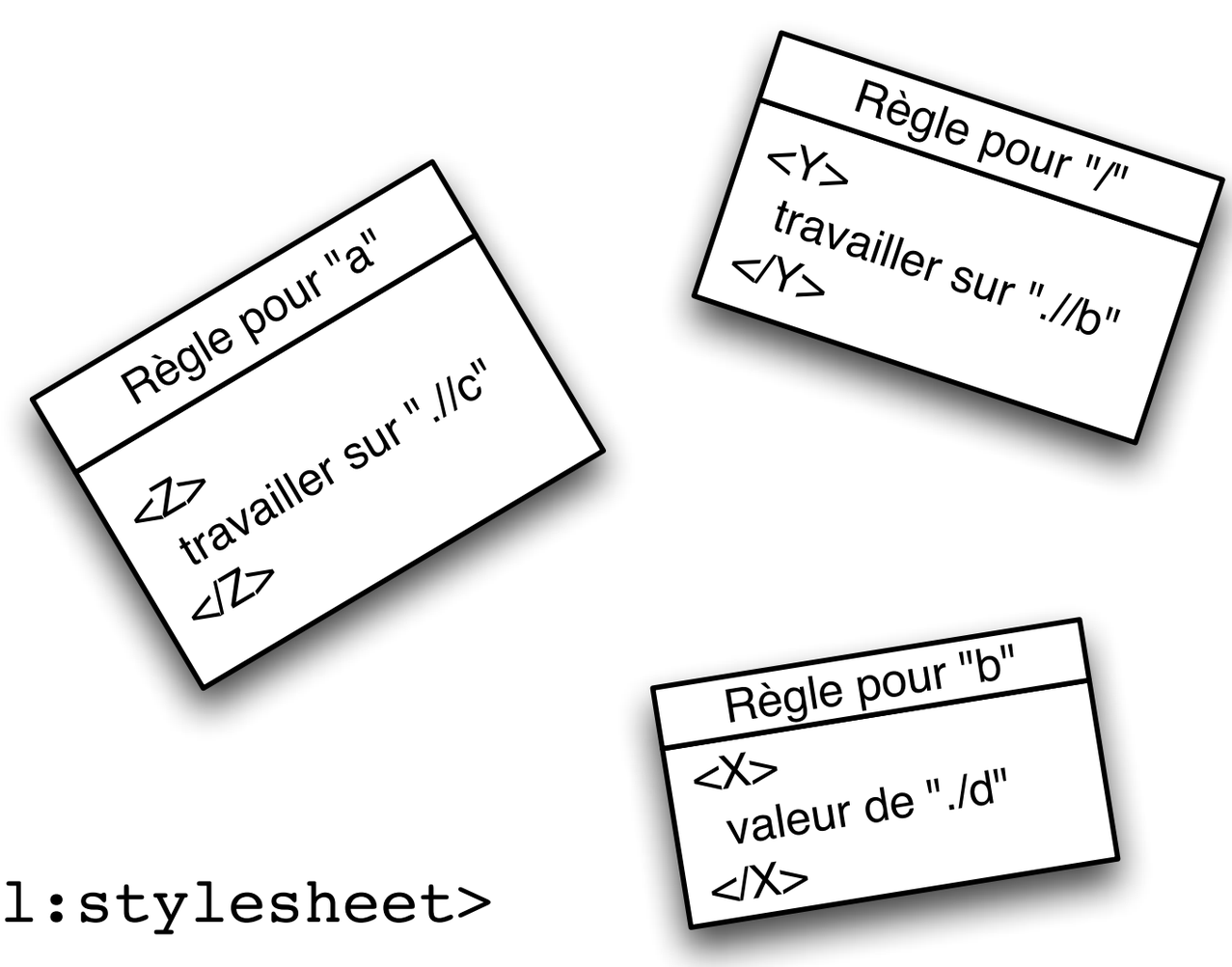
```
<xsl:ouput method="html" indent="yes"/>
```

```
<xsl:ouput method="xml" indent="yes"/>
```

```
<xsl:ouput method="text"/>
```

Définir une feuille XSLT

```
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns="http://www.w3.org/TR/xhtml1/strict">
  <xsl:ouput method="html" indent="yes"/>
```




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

Associer une feuille XSLT à un document - exemple

```xml
<?xml-stylesheet type="text/xsl" href="salutation.xsl"?>
<salutation>
  <a>Marius</a>
</salutation>
```

salutation.xml

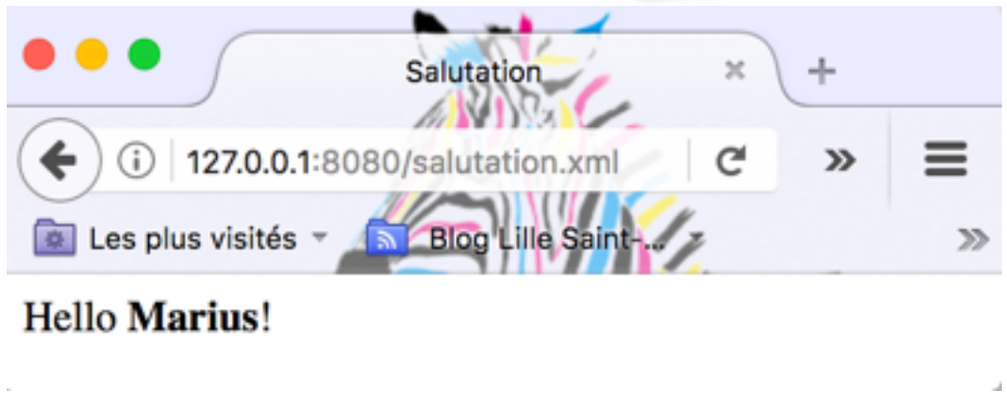
```xml
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html"/>
</xsl:stylesheet>
```

Règle pour **/**

Règle pour **salutation**

Règle pour **a**

salutation.xsl



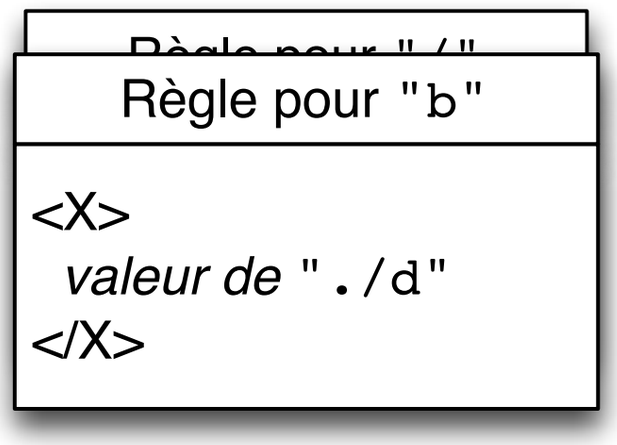

```
<html>
 <head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <title>Salutation</title>
 </head>
 <body>
   Hello
   <b>Marius</b>
   !
 </body>
</html>
```

XSLT

- Règles de transformations
- Application de règles
  - Sequences d’éléments, Priorités, Modes
  - Règles pré-définies
- Opérations sur les noeuds
  - Création des noeuds
  - Structures de contrôle
- Variables / paramètres
  - Règles paramètres
- Espaces de noms
- Exercices

Règles de transformation

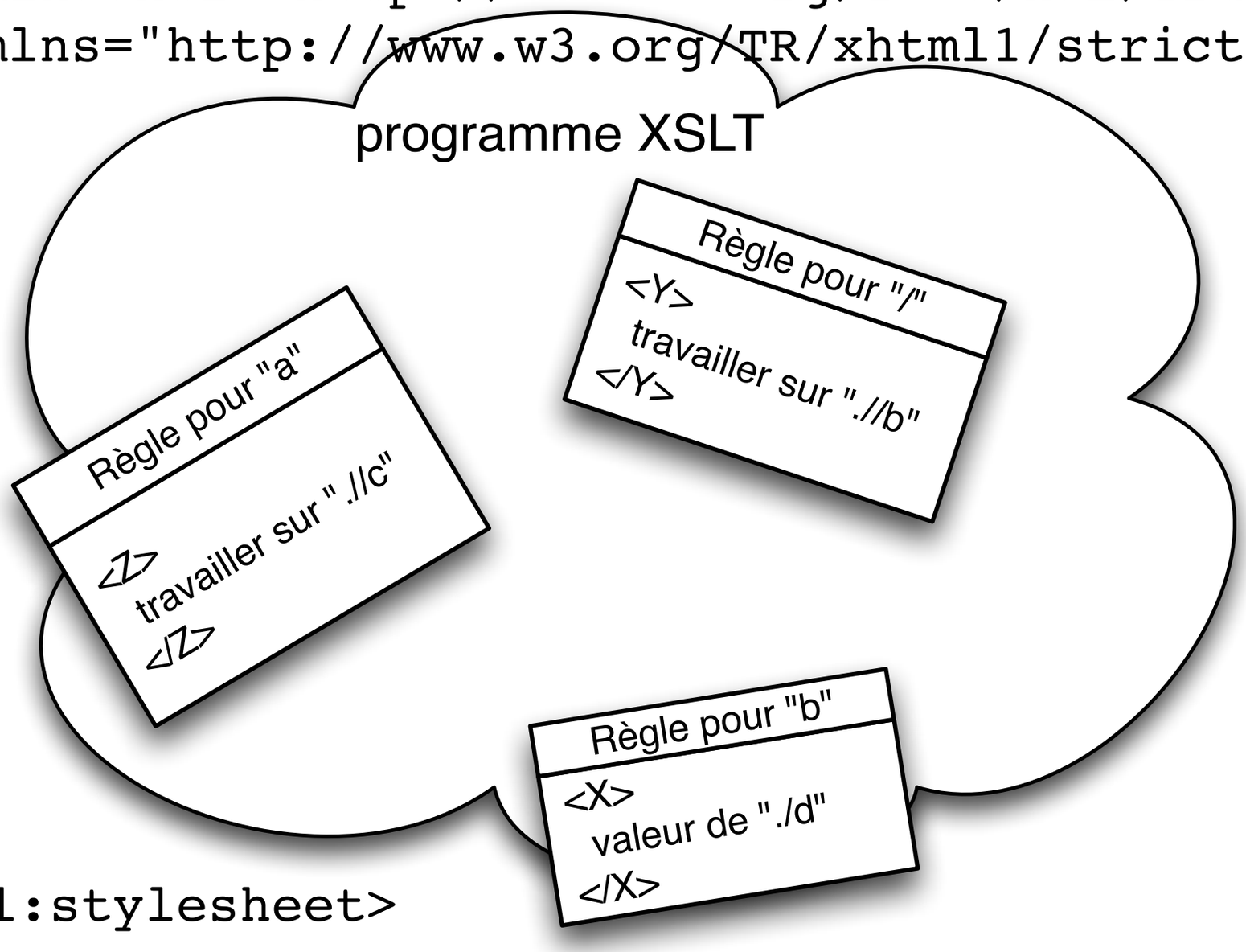
- Une règle est définie avec un élément `xsl:template` et on y trouve :
  - ✓un critère de sélection de nœuds dans le document source (attribut `match`). C'est une requête XPath restreinte aux axes verticaux descendants (mais qui peut contenir des prédicats).
  - ✓Un constructeur de la séquence résultat. Il est évalué pour produire une séquence d'items qui sont écrits dans l'arbre résultat.




```
<xsl:template match="b">
    <X>
        <xsl:apply-templates select=".//b"/>
    </X>
</xsl:template>
```

Définir une feuille XSLT

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns="http://www.w3.org/TR/xhtml1/strict">
```




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

Définir une feuille XSLT

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

    <xsl:output method="html" indent="yes"/>

    <xsl:template match="b">
        <X>
            <xsl:value-of select="d"/>
        </X>
    </xsl:template>

    <xsl:template match="/">
        <Y>
            <xsl:apply-templates select=".//b"/>
        </Y>
    </xsl:template>

    <xsl:template match="a">
        <Z>
            <xsl:apply-templates select=".//c"/>
        </Z>
    </xsl:template>

</xsl:stylesheet>
```

Règle pour `"b"`

<X>
valeur de `"./d"`
</X>

Règle pour `"/"`

<Y>
travailler sur `".//b"`
</Y>

Règle pour `"a"`

<Z>
travailler sur `".//c"`
</Z>

Règles de transformation

- L'attribut `match` du *template* est nécessaire, sauf si le *template* a un attribut `name`. Dans ce cas, l'appel au *template* se fait par son nom :

```
<xsl:call-template name="..."/>
```

```
<xsl:template match="/">
        <xsl:call-templates name="tout_en_majuscules"/>
</xsl:template>
```

```
<xsl:template name="tout_en_majuscules">
        <!-- perte de contexte XML -->
        …
</xsl:template>
```

- L'élément `xsl:apply-templates` permet de continuer la transformation sur une séquence de nœuds définie par une sélection XPath.

```
<xsl:template match="/">
    <Y>
        <xsl:apply-templates select=".//b"/>
    </Y>
</xsl:template>
```

Exemple

```
<xsl:template match="/">
    <html>
        <head>
            <title>biblio</title>
        </head>
        <body>
            <h1>Les Livres</h1>
            <xsl:apply-templates select="library/book"/>
            <h1>Les Auteurs</h1>
            <xsl:apply-templates select="library/author"/>
        </body>
    </html>
</xsl:template>
```

Application des règles

- L'attribut `select` de l'élément `apply-templates` permet de définir la séquence de nœuds sur lesquels il faut appliquer une règle. Sa valeur est une requête XPath, qu'on évalue à partir du nœud courant. Le résultat de cette requête XPath doit être une liste de nœuds.

- En l'absence de `select`, l'instruction `apply-templates` traite tous les nœuds enfants du nœud courant, y compris les nœuds textes mais pas les attributs.

- L'idée est de traiter les nœuds de façon descendante, mais si le chemin XPath valeur de l'attribut `select` permet de remonter dans le document, il est possible de créer une transformation qui ne s'arrête pas (boucle).

Exemple de transformation qui boucle

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
        <a>
            <xsl:apply-templates select=".//book"/>
        </a>
    </xsl:template>
    <xsl:template match="book">
        <b>
            <xsl:apply-templates select="/"/>
        </b>
    </xsl:template>
</xsl:stylesheet>
```

En sortie : `<a><b><a><b><a>.....`

Constructeur de séquence

Séquence de nœuds frères dans la feuille de style (donc en particulier dans un *template*).

Chacun d’eux est, au choix,

1. une instruction XSLT
2. un littéral élément
3. un nœud texte

Constructeur de séquence

- Le constructeur de séquence est évalué pour chaque item de la séquence *s* des nœuds à traiter (dans un *apply-templates*, *s* est la séquence renvoyée par le `select`, ou bien la séquence des nœuds fils)

- Quand un constructeur de séquence est évalué, le processeur conserve donc une trace (appelée *focus*) de la séquence *s* des items en cours de traitement :

✓L'item contexte (souvent nœud contexte),

✓la position de l'item contexte dans la séquence *s*.

✓la taille de la séquence *s*.

Résolution des conflits

```
<xsl:template
  name= Qname
  match = Pattern
  priority = number
  mode = QName
</xsl:template>
```

- Il est possible que plusieurs règles puissent s'appliquer sur l'item courant.
- Pour deux règles qui sont applicables :

1. On compare les attributs `priority` des règles, s'ils existent
2. Sinon, le processeur calcule une priorité en fonction de la "sélectivité" du pattern de la règle.
3. Si malgré tout, il subsiste plusieurs règles, alors le processeur peut déclencher une erreur, ou bien choisir la dernière règle dans l'ordre du document XSLT.

Les modes

```
<xsl:template
  name= Qname
  match = Pattern
  priority = number
  mode = QName
</xsl:template>
```

- Les modes permettent pour une même règle de traiter un même nœud du document source, en produisant des résultats différents.

- Grâce à l'attribut `mode` de `template`, on peut définir pour quel mode une règle s'applique

- L'instruction `apply-templates` dispose aussi d'un attribut `mode`, qui permet de dire dans quel mode on veut appliquer une règle.

- Il existe un mode par défaut, qui n'a pas de nom, utilisé quand aucun nom de mode n'est donné explicitement

Les modes

```
<xsl:template
  name= Qname
  match = Pattern
  priority = number
  mode = QName
</xsl:template>
```

```
<xsl:template match="/">
    <h2>Marques</h2>
    <xsl:apply-templates select="voiture" mode="marque"/>

    <h2>Modèles</h2>
    <xsl:apply-templates select="voiture" mode="modele"/>
</xsl:template>

<xsl:template match="voiture" mode="marque">
    ...
</xsl:template>

<xsl:template match="voiture" mode="modele">
    ...
</xsl:template>
```

Règles prédéfinies

- S'il n'existe pas de règle qui s'applique sur un nœud sélectionné par un `apply-templates`, on évalue une règle prédéfinie (quelque soit le mode)

- Pour un nœud élément, on traite les nœuds fils : c'est comme-ci on avait

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

- Pour un nœud texte ou attribut, on construit un nœud texte qui contient la valeur textuelle du nœud contexte.

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

Règles prédéfinies (2)

La feuille suivante (vide !) permet d'extraire tout le texte (pas les attributs) d'un document :

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

salutation_minimale.xsl

```
<?xml-stylesheet type="text/xsl" href="salutation_minimale.xsl">
<salutation>
  <a>Marius</a>
</salutation>
```

salutation.xml

```
Marius
```

Règles prédéfinies (3)

La feuille suivante (équivalente à une feuille vide !) permet d'extraire tout le texte (pas les attributs) d'un document :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="*|/">
1      <xsl:apply-templates/>
  </xsl:template>
  <xsl:template match="text()|@*">
2      <xsl:value-of select="."/>
  </xsl:template>
</xsl:stylesheet>
```

salutation_min.xsl

```xml
<?xml-stylesheet
   type="text/xsl"
   href="salutation_min.xsl">
<salutation>
  <a>Marius</a>
</salutation>
```

salutation.xml

noeud courant -> *traitement* -> noeuds suivants

/ -> *apply-templates sur fils* -> (salutation)
salutation -> *apply-templates sur fils* -> (a)
a -> *apply-templates sur fils* -> (text())
text() de a -> value-of . -> ()

```
Marius
```

XSLT

- Règles de transformations
- Application de règles
  - Sequences d’éléments, Priorités, Modes
  - Règles pré-définies
- Opérations sur les noeuds
  - Création des noeuds
  - Structures de contrôle
- Variables / paramètres
  - Règles paramètres
- Espaces de noms
- Exercices

Créer des nœuds

1. Création d'éléments

2. Création d'attributs

3. Création de nœuds texte

4. Copie partielle/complète de nœuds

Création d'éléments

- On peut utiliser un littéral élément, en écrivant directement les balises que l'on veut en sortie

```
<xsl:template match="salutation">
  <body>
     Hello <xsl:value-of select="a"/>
  </body>
</xsl:template>
```

- On peut utiliser un élément `xsl:element`, ça permet de construire un élément dont le nom (ou le contenu) est calculé dynamiquement et communiqué dans l'attribut `name` sous la forme d'`une expression entre accolades`.

```
<xsl:template match="voiture[@marque='renault']">
  <renault>
    <xsl:element name="{./modele}">
        <xsl:value-of select="./@annee"/>
    </xsl:element>
  </renault>
</xsl:template>
```

Création d'éléments

```
<voitures>
    <voiture marque="renault" annee="2005">
        <modele>megane</modele>
    </voiture>
    <voiture marque="peugeot" annee="2008">
        <modele>307</modele>
    </voiture>
</voitures>
```

```
<xsl:template match="voiture[@marque='renault']">
  <renault>
    <xsl:element name="{./modele}">
        <xsl:value-of select="./@annee"/>
    </xsl:element>
  </renault>
</xsl:template>
```

est transformé en

```
<?xml version="1.0" encoding="UTF-8"?>
<renault>
    <megane>2005</megane>
</renault>
```

Création d'attributs

- On peut écrire l'attribut "en dur" dans un littéral élément
- On peut aussi utiliser des accolades autour de l'expression définissant la valeur de l'attribut ; {*exp*} représente la valeur de l'évaluation de l'expression *exp*. Par exemple :

```
<xsl:template match="photograph">
    <img src="/images/{href}" width="{size/@width}"/>
</xsl:template>
```

évalué sur :

```
<photograph>
    <href>headquarters.jpg</href>
    <size width="300"/>
</photograph>
```

a pour résultat :

```
<img src="/images/headquarters.jpg" width="300"/>
```

Création d’attributs

- On peut aussi utiliser un élément **`xsl:attribute`** pour définir un nœud attribut. Par exemple :

```
<xsl:template match="image_principale">
    <img>
        <xsl:attribute name="src">
            <xsl:value-of select="./@fichier"/>
        </xsl:attribute>
    </img>
</xsl:template>
```

Création d'un nœud texte

- On peut écrire directement du texte dans le constructeur de séquence

- On peut utiliser l'élément `xsl:text`

```
<xsl:text disable-output-escaping="no">bla &lt; blabla</xsl:text>
                produit le texte « bla &lt; blabla »
```

```
<xsl:text disable-output-escaping="yes">bla &lt; bla</xsl:text>
                produit le texte « bla < blabla »
```

- On peut générer un nœud texte dont le contenu n'est pas statique, grâce à l'élément `xsl:value-of` et son attribut `select`

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

Création d'un nœud texte

Autre exemple :

```
<xsl:template match="/">
    <les-titres>
        <xsl:value-of select=".//titre"/>
    </les-titres>
</xsl:template>
```

```
<livres>
    <livre id="1"><titre>La nuit des temps</titre>…</livre>
    <livre id="2"><titre>Ravages</titre>…</livre>
</livres>
```

```
<les-titres>
    La nuit des temps
    Ravage
</les-titres>
```

X

```
<les-titres>
    La nuit des temps
</les-titres>
```

Note: **xsl:value-of** ne traite que le 1er élément de la liste résultant de l'évaluation du **select**

Recopie d'un nœud (*shallow copy*)

- L'élément `xsl:copy` permet de copier l'item contexte.

- Si l'item contexte est une valeur atomique, l'instruction `xsl:copy` retourne cette valeur, et ne tient pas compte du constructeur de séquence.

- Si l'item contexte est un nœud élément ou un nœud document, l'instruction `xsl:copy` retourne un nœud de même type que le nœud contexte, et qui contient le résultat de l'évaluation du constructeur de séquence contenu dans l'élément `xsl:copy`. Donc, les attributs et le contenu du nœud contexte n'est pas recopié.

- Si l'item contexte est un autre type de nœud (nœud attribut, un nœud texte, ...), l'instruction `xsl:copy` retourne un nœud de même type que le nœud contexte, et qui contient la même valeur texte que le nœud contexte.

Copie récursive (*deep copy*)

- Instruction `xsl:copy-of` avec un attribut `select` obligatoire.

- Les items de la séquence résultat du `select` sont traités de la manière suivante :

1. Si l'item est un nœud élément ou un nœud document, alors on ajoute au résultat un nouveau nœud du même type et de même contenu (attributs, texte, sous-éléments ...) que l'item source
2. Si l'item est un nœud d'un autre type (nœud attribut, un nœud texte, ...), alors la copie est la même qu'avec l'instruction `xsl:copy`
3. Si l'item est de valeur atomique, sa valeur est ajoutée à la séquence résultat.

Exemple : transformation identité

```
<xsl:template match="/">
    <xsl:copy-of select="."/>
</xsl:template>
```

équivalent à :

```
<xsl:template match="@*|node()">
    <xsl:copy>
        <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
</xsl:template>
```

XSLT

- Règles de transformations
- Application de règles
  - Sequences d’éléments, Priorités, Modes
  - Règles pré-définies
- Opérations sur les noeuds
  - Création des noeuds
  - Structures de contrôle
- Variables / paramètres
  - Règles paramètres
- Espaces de noms
- Exercices

Structures de contrôle

1. Répétition : `for each`

2. Conditionnelles : `if` et `choose`

3. Tri : `sort`

Répétition

- L'instruction `xsl:for-each` traite chaque item d'une séquence d'items et pour chacun évalue le constructeur de séquence

- Le résultat d'un `xsl:for-each` est la séquence concaténation des séquences obtenues pour chaque item.

Exemple

```
<xsl:template match="/">
    <html xmlns="http://www.w3.org/1999/xhtml">
        <head><title>Clients</title></head>
        <body>
            <table>
                <tbody>
                    <xsl:for-each select="clients/client">
                        <tr>
                            <th>
                                <xsl:apply-templates select="nom"/>
                            </th>
                            <xsl:for-each select="cmde">
                                <td><xsl:apply-templates/></td>
                            </xsl:for-each>
                        </tr>
                    </xsl:for-each>
                </tbody>
            </table>
        </body>
    </html>
</xsl:template>
```

```
<clients>
  <client id="1">
    <nom>Marius Bilasco</nom>
    <cmde date="2017-02-19">
      1 pizza diabolo
    </cmde>
    <cmde date="2017-02-20">
      1 formule jambon
    </cmde>
  </client>
</clients>
```

Transformation équivalente(1)

```
<xsl:template match="/">
  <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
      <title>Clients</title>
    </head>
    <body>
      <table>
        <tbody>
          <xsl:apply-templates select="clients/client"/>
        </tbody>
      </table>
    </body>
  </html>
</xsl:template>
```

```
<clients>
  <client id="1">
    <nom>Marius Bilasco</nom>
    <cmde date="2017-02-19">
      1 pizza diabolo
    </cmde>
    <cmde date="2017-02-20">
      1 formule jambon
    </cmde>
  </client>
</clients>
```

```
<xsl:template match="client">
  <tr>
    <th>
      <xsl:apply-templates select="nom"/>
    </th>
    <xsl:apply-templates select="cmde"/>
  </tr>
</xsl:template>
```

```
<xsl:template match="cmde">
    <td>
        <xsl:apply-templates/>
    </td>
</xsl:template>
```

Tri des nœuds avant traitement

Par défaut les nœuds sont traités dans l'ordre du document. On peut les traiter dans un ordre différent à l'aide de l'élément `xsl:sort` qui ne peut être fils que d'un élément `xsl:for-each` ou d'un élément `xsl:apply-templates`.

```
<xsl:template match="/">
    <tbody>
        <xsl:for-each select="niveaux/rdc/piece">
            <xsl:sort   select="@surface" order="descending"
                        data-type="number"/>
        </xsl:for-each>
    </tbody>
</xsl:template>
```

```
<xsl:template match="/">
    <tbody>
        <xsl:apply-templates select="niveaux/rdc/piece">
            <xsl:sort   select="@surface" order="descending"
                        data-type="number"/>
        </xsl:apply-templates>
    </tbody>
</xsl:template>
```

Instructions conditionnelles

- Instruction `xsl:if` avec un attribut `test`.

  ✓Le test est une condition XPath.

  ✓Si le test est évalué à vrai, alors le constructeur de séquence à l'intérieur de l'instruction `if` est évalué ; sinon, la séquence vide est retournée.

```
<xsl:if test = "...">...</xsl:if>
```

- Instruction `xsl:choose`

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

XSLT

- Règles de transformations
- Application de règles
  - Sequences d’éléments, Priorités, Modes
  - Règles pré-définies
- Opérations sur les noeuds
  - Création des noeuds
  - Structures de contrôle
- Variables / paramètres
  - Règles paramètres
- Espaces de noms
- Exercices

Variables et paramètres

Garder une trace de l'information lors des changements de contexte d'évaluation

```
<xsl:template match="/">
  …
  <xsl:for-each select="clients/client">
    <xsl:for-each select="cmde">
      <xsl:if test="…"> </xsl:if>
      <!-- vérifier que @date cmd postérieure
           @date inscription client -->

  </xsl:for-each>
</xsl:for-each>
  …
</xsl:template>
```

1

2

Paramétrer le comportement des templates

```
<xsl:template match="/">
  <xsl:apply-templates
        select="salutation">
    <xsl:with-param name="nom"
                    select="'Salut'"/>
  </xsl:apply-templates>
  …
```

```
  …
  <xsl:apply-templates
        select="salutation">
    <xsl:with-param name="nom"
                    select="'Hello'"/>
  </xsl:apply-templates>
</xsl:template>
```

Définition de variable

- Élément `xsl:variable`, avec un attribut `name` obligatoire, et des attributs `select` et `as` optionnels.
- La valeur de la variable est

  ✓soit la valeur de l'évaluation de l'expression du `select`,

  ✓soit la valeur de l'évaluation du constructeur de séquence de l'élément `xsl:variable`.
- Si l'attribut `select` est présent, le constructeur de séquence doit être vide.
- L'attribut `as` donne le type de la variable.
- S'il est absent, alors la variable prend le type de sa valeur. S'il est présent alors la valeur de la variable est convertie en une valeur de ce type.Une variable sans attribut `as` et sans attribut `select` dont le contenu est non vide, est évaluée en un nœud document qui a pour fils l'évaluation du constructeur de séquence (cf exemples)

Exemples

`<xsl:variable name="i" as="xs:integer*" select="1 to 3"/>` a pour valeur la séquence d'entiers `(1 2 3)`

`<xsl:variable name="i" as="xs:integer" select="@size"/>` a pour valeur l'entier valeur de l'attribut `size`

`<xsl:variable name="doc"><c/></xsl:variable>` a pour valeur un nœud document qui a pour fils un élément vide `c`

Utilisation des variables

Il suffit de préfixer le nom de la variable par `$`. Par exemple :

```
<xsl:variable name="n" select="2"/>

...

<xsl:value-of select="td[$n]"/>
```

Règles paramétrées

- L'élément `xsl:param` est utilisé pour définir un paramètre d'un *template*.

- La définition d'un paramètre ressemble fort à la définition d'une variable. L'attribut `select` ou le constructeur de séquence servant à donner une valeur par défaut.

- On a aussi un attribut `required`, faux par défaut, qui indique si le paramètre est obligatoire.

- Quand un paramètre est obligatoire, il ne peut pas avoir de valeur par défaut.

- Dans un `apply-templates` ou un `call-templates`, on donne une valeur au paramètre grâce à l'élément `with-param`.

Exemple

```
<xsl:template match="client">
    <!-- le parametre indique le nombre minimum de commandes obligatoires-->
    <xsl:param name="mini"/>
    <!-- la variable qui contient le nombre de commandes du client courant-->
    <xsl:variable name="nb-cmdes" select="count(cmde)"/>
    <xsl:if test="$mini &lt;= $nb-cmdes">
        <tr>
            <th><xsl:apply-templates select="nom"/></th>
            <xsl:apply-templates select="cmde"/>
        </tr>
    </xsl:if>
</xsl:template>

<xsl:template match="cmde">
    <td><xsl:apply-templates/></td>
</xsl:template>
```

Exemple (2)

On ne veut conserver que les clients qui ont au moins 2 commandes

```
<xsl:template match="/">
  <html xmlns="http://www.w3.org/1999/xhtml">
   <head>
      <title>Clients</title>
   </head>
   <body>
      <table>
         <tbody>
            <xsl:apply-templates select="clients/client">
               <xsl:with-param name="mini" select="2"/>
            </xsl:apply-templates>
         </tbody>
      </table>
   </body>
  </html>
</xsl:template>
```

XSLT

- Règles de transformations
- Application de règles
  - Sequences d’éléments, Priorités, Modes
  - Règles pré-définies
- Opérations sur les noeuds
  - Création des noeuds
  - Structures de contrôle
- Variables / paramètres
  - Règles paramètres
- Espaces de noms
- Exercices

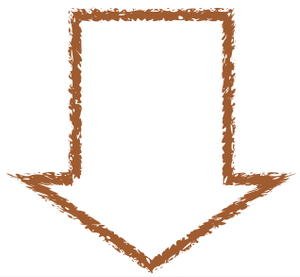
XSLT et espaces de noms

- XSLT est assujetti aux mêmes règles que les documents XML en général
- Les espaces de noms sont introduits par le pseudo-attribut xmlns ou xmlns:*
- Les expressions XPATH employées (match, select, test, etc.) doivent tenir compte des espaces de noms des éléments sources.
  - XSLT 2.0 l'attribut xpath-default-namespace défini au niveau de stylesheet permet de définir l'espace de noms par défaut pour les expression XPATH
- Les éléments créés ne comportant pas d'espace de nom sont associées à l'espace de nom par défaut, s'il est introduit par xmlns
- Tous les namespaces sauf xsl sont exportés par défaut dans l'arbre résultat
  - Il est possible de dissocier les namespaces de certains éléments dans l'arbre résultats en définissant l'attribut exclude-result-prefixes à la racine

Impact des namespaces sur les XPATH dans XSLT

- Les expressions XPATH employées (match, select, test, etc.) doivent tenir des espaces de noms des éléments sources.

```
<?xml-stylesheet type="text/xsl" href="salutation.xsl"?>
<salutation
                                        >
  <a>Marius</a>
</salutation>
```



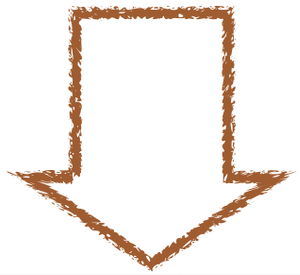
```
Hello Marius!
```

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

        >

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

    <xsl:template match="salutation">
        Hello <xsl:value-of select="a"/>!
    </xsl:template>

</xsl:stylesheet>
```

Impact des namespaces sur les XPATH dans XSLT

- Les expressions XPATH employées (match, select, test, etc.) doivent tenir des espaces de noms des éléments sources.

```
<?xml-stylesheet type="text/xsl" href="salutation.xsl"?>
<salutation
      xmlns="http://labd.fr/salutation">
  <a>Marius</a>
</salutation>
```



X Hello Marius!

X *rien*

OK Marius

car la **règle par défaut** est appliquée

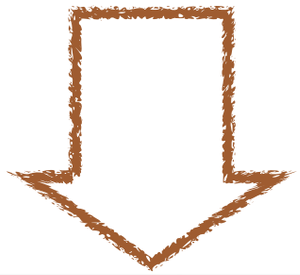
```
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

        >

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

    <xsl:template match="salutation">
        Hello <xsl:value-of select="a"/>!
    </xsl:template>

</xsl:stylesheet>
```

Impact des namespaces sur les XPATH dans XSLT

- Les expressions XPATH employées (match, select, test, etc.) doivent tenir des espaces de noms des éléments sources.

```
<?xml-stylesheet type="text/xsl" href="salutation.xsl"?>
<salutation
      xmlns="http://labd.fr/salutation">
  <a>Marius</a>
</salutation>
```

ici **xmlns** impacte sur la création d'éléments et non pas les XPATH



X `Hello Marius!`

X *rien*

OK `Marius`

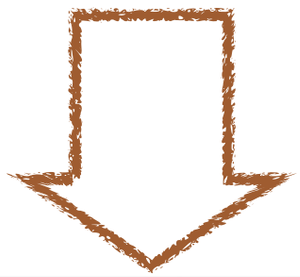
car la **règle par défaut** est appliquée

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

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

    <xsl:template match="salutation">
        Hello <xsl:value-of select="a"/>!
    </xsl:template>

</xsl:stylesheet>
```

Impact des namespaces sur les XPATH dans XSLT

- Les expressions XPATH employées (match, select, test, etc.) doivent tenir des espaces de noms des éléments sources.

```
<?xml-stylesheet type="text/xsl" href="salutation.xsl"?>
<salutation
      xmlns="http://labd.fr/salutation">
  <a>Marius</a>
</salutation>
```



OK Hello Marius!

X *rien*

X Marius

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

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

    <xsl:template match="sal:salutation">
        Hello <b><xsl:value-of select="sal:a"/></b>!
    </xsl:template>

</xsl:stylesheet>
```

Impact des namespaces sur les éléments créés

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

<salutation
     xmlns="http://
labd.fr/salutation">
  <a>Marius</a>
</salutation>
```

X
```
Hello <b>Marius</b>!
```

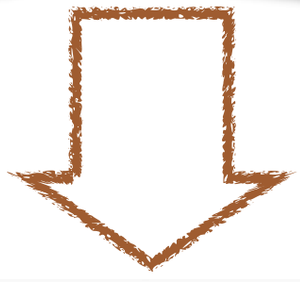
```
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:sal="http://labd.fr/salutation"



        >
    <xsl:template match="/">
      <xsl:apply-templates/>
    </xsl:template>
    <xsl:template match="sal:salutation">
        Hello <b><xsl:value-of select="sal:a"/></b>!
    </xsl:template>
</xsl:stylesheet>
```

OK
```
Hello <b xmlns:sal="http://labd.fr/salutation">Marius</b>!
```

Tous les **namespaces** sauf **xsl** sont exportés par défaut dans l'arbre résultat !

Impact des namespaces sur les éléments créés dans XSLT

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

<salutation
     xmlns="http://
labd.fr/salutation">
  <a>Marius</a>
</salutation>
```



```
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:sal="http://labd.fr/salutation"
        xmlns="http://www.w3.org/1999/xhtml"

        >
    <xsl:template match="/">
      <xsl:apply-templates/>
    </xsl:template>
    <xsl:template match="sal:salutation">
        Hello <b><xsl:value-of select="sal:a"/></b>!
    </xsl:template>
</xsl:stylesheet>
```

X

```
Hello <b>Marius</b>!
```

X

```
Hello <b xmlns="http://www.w3.org/1999/xhtml">Marius</b>!
```

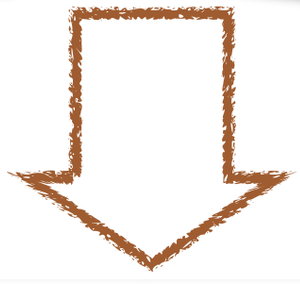
OK

```
Hello <b xmlns="http://www.w3.org/1999/xhtml" xmlns:sal="http://labd.fr/salutation">Marius</b>!
```

Tous les **namespaces** sauf **xsl** sont exportés par défaut dans l'arbre résultat !

Impact des namespaces sur les éléments créés dans XSLT

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

<salutation
     xmlns="http://
labd.fr/salutation">
  <a>Marius</a>
</salutation>
```



```
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:sal="http://labd.fr/salutation"
        xmlns="http://www.w3.org/1999/xhtml"
        exclude-result-prefixes="sal"
        >

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

    <xsl:template match="sal:salutation">
        Hello <b><xsl:value-of select="sal:a"/></b>!
    </xsl:template>

</xsl:stylesheet>
```

X `Hello <b>Marius</b>!`

OK `Hello <b xmlns="http://www.w3.org/1999/xhtml">Marius</b>!`

X `Hello <b xmlns="http://www.w3.org/1999/xhtml" xmlns:sal="http://labd.fr/salutation">Marius</b>!`

exclude-result-prefixes précise les namespaces à ne pas exporter dans l'arbre résultat.

XSLT

- Règles de transformations
- Application de règles
  - Sequences d’éléments, Priorités, Modes
  - Règles pré-définies
- Opérations sur les noeuds
  - Création des noeuds
  - Structures de contrôle
- Variables / paramètres
  - Règles paramètres
- Espaces de noms
- Exercices

Exercices d'application (voir portail)

```
<CATALOG>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <BOTANICAL>
      Arisaema triphyllum
    </BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$3.23</PRICE>
    <AVAILABILITY>020199</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <BOTANICAL>
      Asarum canadense
    </BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.03</PRICE>
    <AVAILABILITY>041899</AVAILABILITY>
  </PLANT>
  …
</CATALOG>
```

plant_catalogue.xml

```
<CLASSIFICATION>
  <FAMILY>
    <NAME>Araceae</NAME>
    <SPECIES>Arisaema triphyllum</SPECIES>
    <SPECIES>Arum italicum</SPECIES>
  </FAMILY>
  <FAMILY>
    <NAME>Aristolochiaceae</NAME>
    <SPECIES>Asarum canadense</SPECIES>
  </FAMILY>
  …
</CLASSIFICATION>
```

plant_families.xml

```
<ORDER>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <QUANTITY>15</QUANTITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <QUANTITY>20</QUANTITY>
  </PLANT>
  …
</ORDER>
```

plant_order.xml

Exercices d'application (voir portail)

1) Recopier tout le document plant_catalog.xml

```
<CATALOG>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <BOTANICAL>
      Arisaema triphyllum
    </BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$3.23</PRICE>
    <AVAILABILITY>020199</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <BOTANICAL>
      Asarum canadense
    </BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.03</PRICE>
    <AVAILABILITY>041899</AVAILABILITY>
  </PLANT>
  …
</CATALOG>
```

plant_catalogue.xml

Exercices d'application (voir portail)

1) Recopier tout le document plant_catalog.xml

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

  <xsl:template match="/">

  </xsl:template>

</xsl:stylesheet>
```

```
<CATALOG>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <BOTANICAL>
      Arisaema triphyllum
    </BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$3.23</PRICE>
    <AVAILABILITY>020199</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <BOTANICAL>
      Asarum canadense
    </BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.03</PRICE>
    <AVAILABILITY>041899</AVAILABILITY>
  </PLANT>
  …
</CATALOG>
```

plant_catalogue.xml

Exercices d'application (voir portail)

1) Recopier tout le document plant_catalog.xml

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

  <xsl:template match="/">
    <xsl:copy-of select="."/>
  </xsl:template>

</xsl:stylesheet>
```

```xml
<CATALOG>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <BOTANICAL>
      Arisaema triphyllum
    </BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$3.23</PRICE>
    <AVAILABILITY>020199</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <BOTANICAL>
      Asarum canadense
    </BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.03</PRICE>
    <AVAILABILITY>041899</AVAILABILITY>
  </PLANT>
  …
</CATALOG>
```

plant_catalogue.xml

Exercices d'application (voir portail)

2) Recopier tout le document plant_catalog.xml en retirant l'élément LIGHT de chaque élément PLANT

```
<CATALOG>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <BOTANICAL>
      Arisaema triphyllum
    </BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$3.23</PRICE>
    <AVAILABILITY>020199</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <BOTANICAL>
      Asarum canadense
    </BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.03</PRICE>
    <AVAILABILITY>041899</AVAILABILITY>
  </PLANT>
  …
</CATALOG>
```

plant_catalogue.xml

Exercices d'application (voir portail)

2) Recopier tout le document plant_catalog.xml en retirant l'élément LIGHT de chaque élément PLANT

```
<CATALOG>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <BOTANICAL>
      Arisaema triphyllum
    </BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$3.23</PRICE>
    <AVAILABILITY>020199</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <BOTANICAL>
      Asarum canadense
    </BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.03</PRICE>
    <AVAILABILITY>041899</AVAILABILITY>
  </PLANT>
  …
</CATALOG>
```

plant_catalogue.xml

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

   <xsl:template match="/">

   </xsl:template>



</xsl:stylesheet>
```

Exercices d'application (voir portail)

2) Recopier tout le document plant_catalog.xml en retirant l'élément LIGHT de chaque élément PLANT

```
<CATALOG>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <BOTANICAL>
      Arisaema triphyllum
    </BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$3.23</PRICE>
    <AVAILABILITY>020199</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <BOTANICAL>
      Asarum canadense
    </BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.03</PRICE>
    <AVAILABILITY>041899</AVAILABILITY>
  </PLANT>
  …
</CATALOG>
```

plant_catalogue.xml

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

    <xsl:template match="/">
     <CATALOG>
        <xsl:apply-templates  select="//PLANT"/>
     </CATALOG>
    </xsl:template>

   <xsl:template match="PLANT">



    </xsl:template>

</xsl:stylesheet>
```

Exercices d'application (voir portail)

2) Recopier tout le document plant_catalog.xml en retirant l'élément LIGHT de chaque élément PLANT

```
<CATALOG>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <BOTANICAL>
      Arisaema triphyllum
    </BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$3.23</PRICE>
    <AVAILABILITY>020199</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <BOTANICAL>
      Asarum canadense
    </BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.03</PRICE>
    <AVAILABILITY>041899</AVAILABILITY>
  </PLANT>
  …
</CATALOG>
```

plant_catalogue.xml

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

   <xsl:template match="/">
    <CATALOG>
       <xsl:apply-templates  select="//PLANT"/>
    </CATALOG>
   </xsl:template>

  <xsl:template match="PLANT">
    <xsl:copy>
       <xsl:copy-of select="*[name() != 'LIGHT']"/>
    </xsl:copy>
   </xsl:template>

</xsl:stylesheet>
```

Exercices d'application (voir portail)

3) Donner une transformation qui classe et regroupe les éléments PLANT du fichier plant-catalog.xml en fonction du contenu de leur élément LIGHT

```
<CATALOG>
    <LIGHT>
        <EXPOSURE>Mostly Shady</EXPOSURE>
        <PLANT>
            <COMMON>Bloodroot</COMMON>
            <BOTANICAL>Sanguinaria canadensis</BOTANICAL>
            <ZONE>4</ZONE>
            <PRICE>$2.44</PRICE>
            <AVAILABILITY>031599</AVAILABILITY>
    </PLANT>
    <PLANT>
        ...
    <EXPOSURE>Mostly Sunny</EXPOSURE>
    <PLANT>
        <COMMON>Marsh Marigold</COMMON>
```

```
<CATALOG>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <BOTANICAL>
      Arisaema triphyllum
    </BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$3.23</PRICE>
    <AVAILABILITY>020199</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <BOTANICAL>
      Asarum canadense
    </BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.03</PRICE>
    <AVAILABILITY>041899</AVAILABILITY>
  </PLANT>
  …
</CATALOG>
```

plant_catalogue.xml

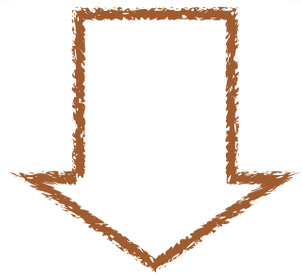
Exercices d'application (voir portail)

3) Donner une transformation qui classe et regroupe les éléments PLANT du fichier plant-catalog.xml en fonction du contenu de leur élément LIGHT

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="xml" indent="yes"/>
   <xsl:template match="/">
    <CATALOG>
    </CATALOG>
   </xsl:template>
   <xsl:template match="PLANT">
    <xsl:copy>
       <xsl:copy-of select="*[name() != 'LIGHT']"/>
    </xsl:copy>
   </xsl:template>
</xsl:stylesheet>
```

```
<CATALOG>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <BOTANICAL>
      Arisaema triphyllum
    </BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$3.23</PRICE>
    <AVAILABILITY>020199</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <BOTANICAL>
      Asarum canadense
    </BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.03</PRICE>
    <AVAILABILITY>041899</AVAILABILITY>
  </PLANT>
  …
</CATALOG>
```

plant_catalogue.xml



```
<CATALOG>
  <LIGHT>
    <EXPOSURE>Mostly Shady</EXPOSURE>
    <PLANT>
     <COMMON>Bloodroot</COMMON>
     <BOTANICAL>Sanguinaria canadensis</
BOTANICAL>
      <ZONE>4</ZONE>
      <PRICE>$2.44</PRICE>
      <AVAILABILITY>031599</AVAILABILITY>
     </PLANT>
     <PLANT>…</PLANT>
  </LIGHT>
  <LIGHT>
     <EXPOSURE>Mostly Sunny</EXPOSURE>
     <PLANT>
       <COMMON>Marsh Marigold</COMMON>
       …
     </PLANT>
  </LIGHT>
</CATALOG>
```

plant_catalogue_light.xml

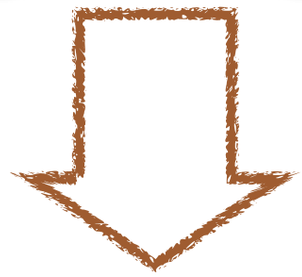
Exercices d'application (voir portail)

3) Donner une transformation qui classe et regroupe les éléments PLANT du fichier plant-catalog.xml en fonction du contenu de leur élément LIGHT

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="xml" indent="yes"/>
   <xsl:template match="/">
    <CATALOG>
       <xsl:for-each select="//LIGHT[not (following::LIGHT = .)]">
          <LIGHT>
          </LIGHT>
       </xsl:for-each>
    </CATALOG>
   </xsl:template>
   <xsl:template match="PLANT">
    <xsl:copy>
       <xsl:copy-of select="*[name() != 'LIGHT']"/>
    </xsl:copy>
   </xsl:template>
</xsl:stylesheet>
```

```
<CATALOG>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <BOTANICAL>
      Arisaema triphyllum
    </BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$3.23</PRICE>
    <AVAILABILITY>020199</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <BOTANICAL>
      Asarum canadense
    </BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.03</PRICE>
    <AVAILABILITY>041899</AVAILABILITY>
  </PLANT>
  …
</CATALOG>
```

plant_catalogue.xml



```
<CATALOG>
  <LIGHT>
    <EXPOSURE>Mostly Shady</EXPOSURE>
    <PLANT>
     <COMMON>Bloodroot</COMMON>
     <BOTANICAL>Sanguinaria canadensis</
BOTANICAL>
      <ZONE>4</ZONE>
      <PRICE>$2.44</PRICE>
      <AVAILABILITY>031599</AVAILABILITY>
    </PLANT>
    <PLANT>…</PLANT>
  </LIGHT>
  <LIGHT>
    <EXPOSURE>Mostly Sunny</EXPOSURE>
    <PLANT>
      <COMMON>Marsh Marigold</COMMON>
      …
    </PLANT>
  </LIGHT>
</CATALOG>
```

plant_catalogue_light.xml

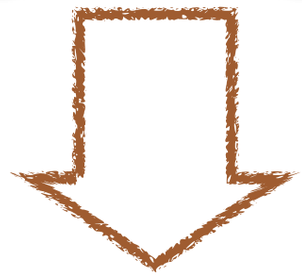
Exercices d'application (voir portail)

3) Donner une transformation qui classe et regroupe les éléments PLANT du fichier plant-catalog.xml en fonction du contenu de leur élément LIGHT

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="xml" indent="yes"/>
   <xsl:template match="/">
    <CATALOG>
       <xsl:for-each select="//LIGHT[not (following::LIGHT = .)]">
          <LIGHT>
             <EXPOSURE>
                <xsl:value-of select="."/>
             </EXPOSURE>


          </LIGHT>
       </xsl:for-each>
    </CATALOG>
   </xsl:template>
   <xsl:template match="PLANT">
    <xsl:copy>
       <xsl:copy-of select="*[name() != 'LIGHT']"/>
    </xsl:copy>
   </xsl:template>
</xsl:stylesheet>
```

```
<CATALOG>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <BOTANICAL>
      Arisaema triphyllum
    </BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$3.23</PRICE>
    <AVAILABILITY>020199</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <BOTANICAL>
      Asarum canadense
    </BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.03</PRICE>
    <AVAILABILITY>041899</AVAILABILITY>
  </PLANT>
  …
</CATALOG>
```

plant_catalogue.xml



```
<CATALOG>
  <LIGHT>
    <EXPOSURE>Mostly Shady</EXPOSURE>
    <PLANT>
     <COMMON>Bloodroot</COMMON>
     <BOTANICAL>Sanguinaria canadensis</
BOTANICAL>
      <ZONE>4</ZONE>
      <PRICE>$2.44</PRICE>
      <AVAILABILITY>031599</AVAILABILITY>
    </PLANT>
    <PLANT>…</PLANT>
  </LIGHT>
  <LIGHT>
    <EXPOSURE>Mostly Sunny</EXPOSURE>
    <PLANT>
      <COMMON>Marsh Marigold</COMMON>
      …
    </PLANT>
  </LIGHT>
</CATALOG>
```

plant_catalogue_light.xml

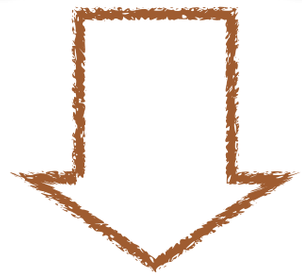
Exercices d'application (voir portail)

3) Donner une transformation qui classe et regroupe les éléments PLANT du fichier plant-catalog.xml en fonction du contenu de leur élément LIGHT

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="xml" indent="yes"/>
   <xsl:template match="/">
    <CATALOG>
       <xsl:for-each select="//LIGHT[not (following::LIGHT = .)]">
          <LIGHT>
             <EXPOSURE>
                <xsl:value-of select="."/>
             </EXPOSURE>

             <xsl:apply-templates select="//PLANT[LIGHT=. "/>
          </LIGHT>
       </xsl:for-each>
    </CATALOG>
   </xsl:template>
   <xsl:template match="PLANT">
    <xsl:copy>
       <xsl:copy-of select="*[name() != 'LIGHT']"/>
    </xsl:copy>
   </xsl:template>
</xsl:stylesheet>
```

```
<CATALOG>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <BOTANICAL>
      Arisaema triphyllum
    </BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$3.23</PRICE>
    <AVAILABILITY>020199</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <BOTANICAL>
      Asarum canadense
    </BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.03</PRICE>
    <AVAILABILITY>041899</AVAILABILITY>
  </PLANT>
  …
</CATALOG>
```

plant_catalogue.xml



```
<CATALOG>
  <LIGHT>
    <EXPOSURE>Mostly Shady</EXPOSURE>
    <PLANT>
     <COMMON>Bloodroot</COMMON>
     <BOTANICAL>Sanguinaria canadensis</
BOTANICAL>
      <ZONE>4</ZONE>
      <PRICE>$2.44</PRICE>
      <AVAILABILITY>031599</AVAILABILITY>
    </PLANT>
    <PLANT>…</PLANT>
  </LIGHT>
  <LIGHT>
    <EXPOSURE>Mostly Sunny</EXPOSURE>
    <PLANT>
      <COMMON>Marsh Marigold</COMMON>
      …
    </PLANT>
  </LIGHT>
</CATALOG>
```

plant_catalogue_light.xml

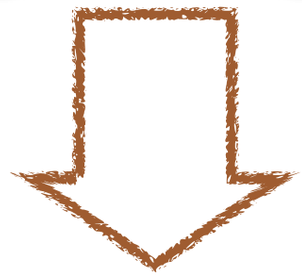
Exercices d'application (voir portail)

3) Donner une transformation qui classe et regroupe les éléments PLANT du fichier plant-catalog.xml en fonction du contenu de leur élément LIGHT

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="xml" indent="yes"/>
   <xsl:template match="/">
    <CATALOG>
       <xsl:for-each select="//LIGHT[not (following::LIGHT = .)]">
          <LIGHT>
             <EXPOSURE>
                <xsl:value-of select="."/>
             </EXPOSURE>
             <xsl:variable name="exposure" select="."/>
             <xsl:apply-templates select="//PLANT[LIGHT=$exposure]"/>
          </LIGHT>
       </xsl:for-each>
    </CATALOG>
   </xsl:template>
   <xsl:template match="PLANT">
    <xsl:copy>
       <xsl:copy-of select="*[name() != 'LIGHT']"/>
    </xsl:copy>
   </xsl:template>
</xsl:stylesheet>
```

```
<CATALOG>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <BOTANICAL>
      Arisaema triphyllum
    </BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$3.23</PRICE>
    <AVAILABILITY>020199</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <BOTANICAL>
      Asarum canadense
    </BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.03</PRICE>
    <AVAILABILITY>041899</AVAILABILITY>
  </PLANT>
  …
</CATALOG>
```

plant_catalogue.xml



```
<CATALOG>
  <LIGHT>
    <EXPOSURE>Mostly Shady</EXPOSURE>
    <PLANT>
     <COMMON>Bloodroot</COMMON>
     <BOTANICAL>Sanguinaria canadensis</
     <ZONE>4</ZONE>
     <PRICE>$2.44</PRICE>
     <AVAILABILITY>031599</AVAILABILITY>
    </PLANT>
    <PLANT>…</PLANT>
  </LIGHT>
  <LIGHT>
    <EXPOSURE>Mostly Sunny</EXPOSURE>
    <PLANT>
      <COMMON>Marsh Marigold</COMMON>
      …
    </PLANT>
  </LIGHT>
</CATALOG>
```

plant_catalogue_light.xml

Exercices d'application (voir portail)

4) Donner une transformation qui ajoute dans chaque élément PLANT du fichier plant-catalog.xml un élément FAMILY contenant le nom de la famille de la plante disponible dans le fichier plant-families.xml comme ci-dessous :

```
<?xml version="1.0" encoding="utf-8"?>
<CATALOG>
   <PLANT>
      <COMMON>Bloodroot</COMMON>
      <BOTANICAL>Sanguinaria canadensis</BOTANICAL>
      <ZONE>4</ZONE>
      <LIGHT>Mostly Shady</LIGHT>
      <PRICE>$2.44</PRICE>
      <AVAILABILITY>031599</AVAILABILITY>
      <FAMILY>Papaveraceae</FAMILY>
 </PLANT>
 <PLANT>
      <COMMON>Columbine</COMMON>
      <BOTANICAL>Aquilegia canadensis</BOTANICAL>
      <ZONE>3</ZONE>
      <LIGHT>Mostly Shady</LIGHT>
      <PRICE>$9.37</PRICE>
      <AVAILABILITY>030699</AVAILABILITY>
      <FAMILY>Ranunculaceae</FAMILY>
 </PLANT>
...
```

```
<CATALOG>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <BOTANICAL>
      Arisaema triphyllum
    </BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$3.23</PRICE>
    <AVAILABILITY>020199</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <BOTANICAL>
      Asarum canadense
    </BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.03</PRICE>
    <AVAILABILITY>041899</AVAILABILITY>
  </PLANT>
  …
</CATALOG>
```

plant_catalogue.xml

```
<CLASSIFICATION>
  <FAMILY>
   <NAME>Araceae</NAME>
   <SPECIES>Arisaema triphyllum</SPECIES>
   <SPECIES>Arum italicum</SPECIES>
  </FAMILY>
  <FAMILY>
   <NAME>Aristolochiaceae</NAME>
   <SPECIES>Asarum canadense</SPECIES>
  </FAMILY>
  …
</CLASSIFICATION>
```

plant_families.xml

Exercices d'application (voir portail)

4) Donner une transformation qui ajoute dans chaque élément PLANT du fichier plant-catalog.xml un élément FAMILY contenant le nom de la famille de la plante disponible dans le fichier plant-families.xml

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

  <xsl:template match="/">
    <CATALOG>
      <xsl:apply-templates/>
    </CATALOG>
  </xsl:template>

  <xsl:template match="PLANT">









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

```
<CATALOG>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <BOTANICAL>
      Arisaema triphyllum
    </BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$3.23</PRICE>
    <AVAILABILITY>020199</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <BOTANICAL>
      Asarum canadense
    </BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.03</PRICE>
    <AVAILABILITY>041899</AVAILABILITY>
  </PLANT>
  …
</CATALOG>
```

plant_catalogue.xml

```
<CLASSIFICATION>
  <FAMILY>
    <NAME>Araceae</NAME>
    <SPECIES>Arisaema triphyllum</SPECIES>
    <SPECIES>Arum italicum</SPECIES>
  </FAMILY>
  <FAMILY>
    <NAME>Aristolochiaceae</NAME>
    <SPECIES>Asarum canadense</SPECIES>
  </FAMILY>
  …
</CLASSIFICATION>
```

plant_families.xml

Exercices d'application (voir portail)

4) Donner une transformation qui ajoute dans chaque élément PLANT du fichier plant-catalog.xml un élément FAMILY contenant le nom de la famille de la plante disponible dans le fichier plant-families.xml

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

  <xsl:template match="/">
    <CATALOG>
      <xsl:apply-templates/>
    </CATALOG>
  </xsl:template>

  <xsl:template match="PLANT">

    <xsl:copy>
      <xsl:copy-of select="*"/>
      <FAMILY>
        <xsl:value-of select=     />

      </FAMILY>
    </xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```

```
<CATALOG>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <BOTANICAL>
      Arisaema triphyllum
    </BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$3.23</PRICE>
    <AVAILABILITY>020199</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <BOTANICAL>
      Asarum canadense
    </BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.03</PRICE>
    <AVAILABILITY>041899</AVAILABILITY>
  </PLANT>
  …
</CATALOG>
```

plant_catalogue.xml

```
<CLASSIFICATION>
  <FAMILY>
   <NAME>Araceae</NAME>
   <SPECIES>Arisaema triphyllum</SPECIES>
   <SPECIES>Arum italicum</SPECIES>
  </FAMILY>
  <FAMILY>
   <NAME>Aristolochiaceae</NAME>
   <SPECIES>Asarum canadense</SPECIES>
  </FAMILY>
  …
</CLASSIFICATION>
```

plant_families.xml

Exercices d'application (voir portail)

4) Donner une transformation qui ajoute dans chaque élément PLANT du fichier plant-catalog.xml un élément FAMILY contenant le nom de la famille de la plante disponible dans le fichier plant-families.xml

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

  <xsl:template match="/">
    <CATALOG>
      <xsl:apply-templates/>
    </CATALOG>
  </xsl:template>

  <xsl:template match="PLANT">

    <xsl:copy>
      <xsl:copy-of select="*"/>
      <FAMILY>
        <xsl:value-of select=
        "document('plant_families.xml')//FAMILY[SPECIES =
                                                ]/NAME/text()"/>
      </FAMILY>
    </xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```

```
<CATALOG>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <BOTANICAL>
      Arisaema triphyllum
    </BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$3.23</PRICE>
    <AVAILABILITY>020199</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <BOTANICAL>
      Asarum canadense
    </BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.03</PRICE>
    <AVAILABILITY>041899</AVAILABILITY>
  </PLANT>
  …
</CATALOG>
```

plant_catalogue.xml

```
<CLASSIFICATION>
  <FAMILY>
    <SPECIES>Asarum lemmonii</SPECIES>
  </FAMILY>
  <FAMILY>
   <NAME>Aristolochiaceae</NAME>
   <SPECIES>Asarum canadense</SPECIES>
  </FAMILY>
  …
</CLASSIFICATION>
```

plant_families.xml

Exercices d'application (voir portail)

4) Donner une transformation qui ajoute dans chaque élément PLANT du fichier plant-catalog.xml un élément FAMILY contenant le nom de la famille de la plante disponible dans le fichier plant-families.xml

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

  <xsl:template match="/">
    <CATALOG>
      <xsl:apply-templates/>
    </CATALOG>
  </xsl:template>

  <xsl:template match="PLANT">
    <xsl:variable name="botanical" select="BOTANICAL"/>
    <xsl:copy>
      <xsl:copy-of select="*"/>
      <FAMILY>
        <xsl:value-of select=
        "document('plant_families.xml')//FAMILY[SPECIES = $botanical]/NAME/text()"/>
      </FAMILY>
    </xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```

```
<CATALOG>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <BOTANICAL>
      Arisaema triphyllum
    </BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$3.23</PRICE>
    <AVAILABILITY>020199</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <BOTANICAL>
      Asarum canadense
    </BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.03</PRICE>
    <AVAILABILITY>041899</AVAILABILITY>
  </PLANT>
  …
</CATALOG>
```

plant_catalogue.xml

```
<CLASSIFICATION>
  <FAMILY>
    <NAME>Aristolochiaceae</NAME>
    <SPECIES>Asarum canadense</SPECIES>
  </FAMILY>
  <FAMILY>
    <NAME>Aristolochiaceae</NAME>
    <SPECIES>Asarum canadense</SPECIES>
  </FAMILY>
  …
</CLASSIFICATION>
```

plant_families.xml

Exercices d'application (voir portail)

5) Donner une transformation qui calcule le montant total pour chaque plante achetée de la commande décrite dans plant-order.xml comme dans l'exemple :

```
<PRICES>
   <PLANT>
      <COMMON>Bloodroot</COMMON>
      <PRICE>36.6</PRICE>
   </PLANT>
   <PLANT>
      <COMMON>Hepatica</COMMON>
      <PRICE>89</PRICE>
</PLANT>
<PLANT>
      <COMMON>Phlox, Blue</COMMON>
      <PRICE>27.95</PRICE>
</PLANT>
<PLANT>
      <COMMON>Trillium</COMMON>
      <PRICE>97.5</PRICE>
</PLANT>
<PLANT>
      <COMMON>Adder's-Tongue</COMMON>
      <PRICE>47.9</PRICE>
```

```
<CATALOG>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <BOTANICAL>
      Arisaema triphyllum
    </BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$3.23</PRICE>
    <AVAILABILITY>020199</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <BOTANICAL>
      Asarum canadense
    </BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.03</PRICE>
    <AVAILABILITY>041899</AVAILABILITY>
  </PLANT>
  …
</CATALOG>
```

plant_catalogue.xml

```
<ORDER>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <QUANTITY>15</QUANTITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <QUANTITY>20</QUANTITY>
  </PLANT>
  …
</ORDER>
```

plant_order.xml

Exercices d'application (voir portail)

5) Donner une transformation qui calcule le montant total pour chaque plante achetée de la commande décrite dans plant-order.xml

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

   <xsl:template match="/">
    <PRICES>
       <xsl:apply-templates select="//PLANT"/>
    </PRICES>
   </xsl:template>

   <xsl:template match="PLANT">









   </xsl:template>

</xsl:stylesheet>
```

```
<CATALOG>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <BOTANICAL>
      Arisaema triphyllum
    </BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$3.23</PRICE>
    <AVAILABILITY>020199</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <BOTANICAL>
      Asarum canadense
    </BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.03</PRICE>
    <AVAILABILITY>041899</AVAILABILITY>
  </PLANT>
  …
</CATALOG>
```

plant_catalogue.xml

```
<ORDER>
  <PLANT>
   <COMMON>Jack-In-The-Pulpit</COMMON>
   <QUANTITY>15</QUANTITY>
  </PLANT>
  <PLANT>
   <COMMON>Ginger, Wild</COMMON>
   <QUANTITY>20</QUANTITY>
  </PLANT>
  …
</ORDER>
```

plant_order.xml

Exercices d'application (voir portail)

5) Donner une transformation qui calcule le montant total pour chaque plante achetée de la commande décrite dans plant-order.xml

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

   <xsl:template match="/">
    <PRICES>
       <xsl:apply-templates select="//PLANT"/>
    </PRICES>
   </xsl:template>

   <xsl:template match="PLANT">
    <xsl:copy>
       <xsl:copy-of select="COMMON"/>

       <PRICE>



       </PRICE>
    </xsl:copy>
   </xsl:template>

</xsl:stylesheet>
```

```
<CATALOG>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <BOTANICAL>
      Arisaema triphyllum
    </BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$3.23</PRICE>
    <AVAILABILITY>020199</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <BOTANICAL>
      Asarum canadense
    </BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.03</PRICE>
    <AVAILABILITY>041899</AVAILABILITY>
  </PLANT>
  …
</CATALOG>
```

plant_catalogue.xml

```
<ORDER>
  <PLANT>
   <COMMON>Jack-In-The-Pulpit</COMMON>
   <QUANTITY>15</QUANTITY>
  </PLANT>
  <PLANT>
   <COMMON>Ginger, Wild</COMMON>
   <QUANTITY>20</QUANTITY>
  </PLANT>
  …
</ORDER>
```

plant_order.xml

Exercices d'application (voir portail)

5) Donner une transformation qui calcule le montant total pour chaque plante achetée de la commande décrite dans plant-order.xml

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

   <xsl:template match="/">
    <PRICES>
        <xsl:apply-templates select="//PLANT"/>
    </PRICES>
   </xsl:template>

   <xsl:template match="PLANT">
    <xsl:copy>
        <xsl:copy-of select="COMMON"/>
        <xsl:variable name="name" select="COMMON"/>
        <PRICE>
            <xsl:value-of select=
"QUANTITY * number(substring-after(document('plant_catalog.xml')//PLANT[COMMON=$name]/
PRICE,'$'))"/>
        </PRICE>
    </xsl:copy>
   </xsl:template>

 </xsl:stylesheet>
```

```
<CATALOG>
  <PLANT>
    <COMMON>Jack-In-The-Pulpit</COMMON>
    <BOTANICAL>
      Arisaema triphyllum
    </BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$3.23</PRICE>
    <AVAILABILITY>020199</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Ginger, Wild</COMMON>
    <BOTANICAL>
      Asarum canadense
    </BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.03</PRICE>
    <AVAILABILITY>041899</AVAILABILITY>
  </PLANT>
  …
</CATALOG>
```

plant_catalogue.xml

```
  </PLANT>
  <PLANT>
   <COMMON>Ginger, Wild</COMMON>
   <QUANTITY>20</QUANTITY>
  </PLANT>
  …
</ORDER>
```

plant_order.xml