

Cours 4

Plan

- Le processus de transformation XSLT
- Quelques éléments du langage
- Le langage XSL

1

Introduction: langage XSL

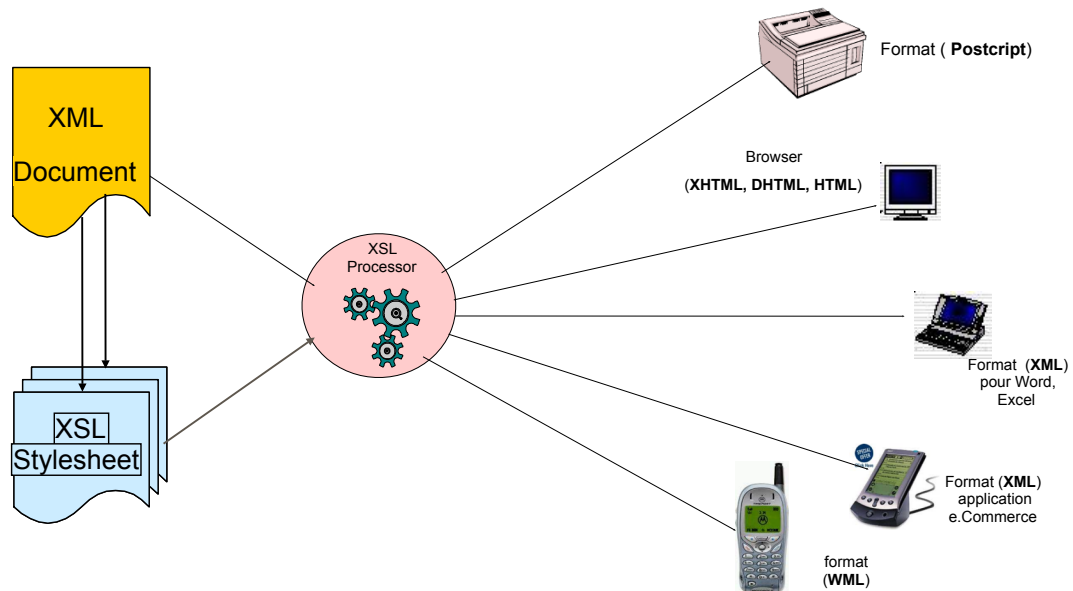
Définition

XSL (e**X**tensible **S**tylesheet **L**anguage) est un langage utilisé pour appliquer des mises en forme aux données XML.

Ce langage permet de décrire des feuilles de style qui présentent les règles de présentation des éléments XML.

Le document XML est transformé en un document (XML, HTML, texte...) par le processus XSLT (XML Transformation)

2



3

Les logiciels de transformation

xalan: Un processeur gratuit écrit en Java, par Apache

(<http://www.apache.org/>)

xt: Un processeur gratuit écrit en Java, par James Clark

(<http://www.jclark.com/>)

saxon: Un processeur gratuit écrit en Java, par Michael Kay

(<http://users.iclway.co.uk/mhkay/saxon>)

Via des outils en ligne

(<http://www.utilities-online.info/xslttransformation/>)

Les navigateurs Internet

Sous linux commande **xsltproc**

xsltproc fichier.xsl fichier.xml

4

Un premier exemple: le fichier XML

- Mettre une instruction de traitement PI en haut de votre document XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="melEx1.xsl"?>

<mel>
  <expediteur>Duraton</expediteur>
  <recepteurs> Duchene</recepteurs>
  <recepteurs> Elcharky</recepteurs>
  <objet> cours xml </objet>
  <contenu xml:lang='fr'>
    <entete> Bonjour</entete>
    <texte>
      je recherche des livres sur xml. as-tu des références?
    </texte>
    <signature email="duraton@iutv.univ-paris13.fr" >*****
    </signature>
  </contenu>
  <fichier_joint/>
</mel>
```

5

Un premier exemple: une feuille de style

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/
  Transform" version="1.0">
  <xsl:output method="text" encoding="ISO-8859-1"/>
  <xsl:template match="/">
    Signature <xsl:value-of select="mel/expediteur"/>!
  </xsl:template>
</xsl:stylesheet>
```

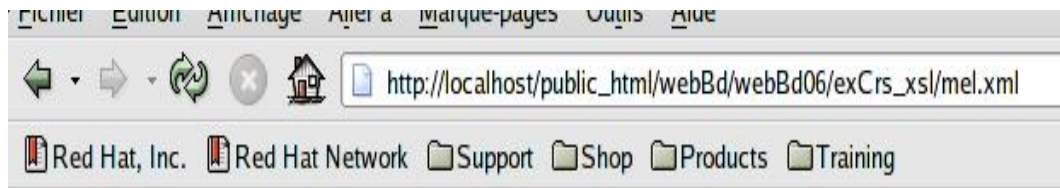
Fichier **melEx1.xsl**

On en est à la version 3 !! <http://www.w3.org/TR/xslt-30/>

W3C Working Draft 10 July 2012 (version 2 rec. 2007)

6

Résultat dans Firefox



Signature Duraton!

7

Un deuxième exemple: feuille de style

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html" encoding="ISO-8859-1"/>
  <xsl:template match="/">
    <H2> Exemple 2 - cours xsl </H2>
    <html>
      <HEAD>
        <TITLE>Bonjour</TITLE>
      </HEAD>
      <body>
        Salut <xsl:value-of select="mel/recepteurs"/>!
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Fichier ex1_bis.xsl

8

Exemple 2 - cours xsl

Salut Duchene!

9

Introduction: Caractéristiques

- Le langage XSL contient des instructions permettant :
 - d'extraire des éléments du document XML
 - d'appliquer des règles de style sur ces éléments XML

Le document contenant les éléments de mise en page s'appelle une **feuille de style**

■ Une feuille de style est un document XML contenu dans un fichier d'extension **xsl**. Les éléments du langage xsl sont préfixés par **l'espace de nom xsl**.

Syntaxe d'une feuille de style

```
<?xml version="1.0" encoding="ISO-8859-1"?>  
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >  
  modèles  
</xsl:stylesheet>
```

-
- Une feuille de style xsl doit respecter:
 - les règles de forme des documents XML et
 - le schéma XSLT du W3C dont l'espace de noms est `= "http://www.w3.org/1999/XSL/Transform"`
 - L'élément racine `xsl:stylesheet` peut contenir plusieurs éléments fils optionnels comme:
 - `<xsl:include href="style.xsl">` inclusion des règles d'une feuille de style externe *style.xsl*.
 - `<xsl:output>` permet de spécifier le format de sortie

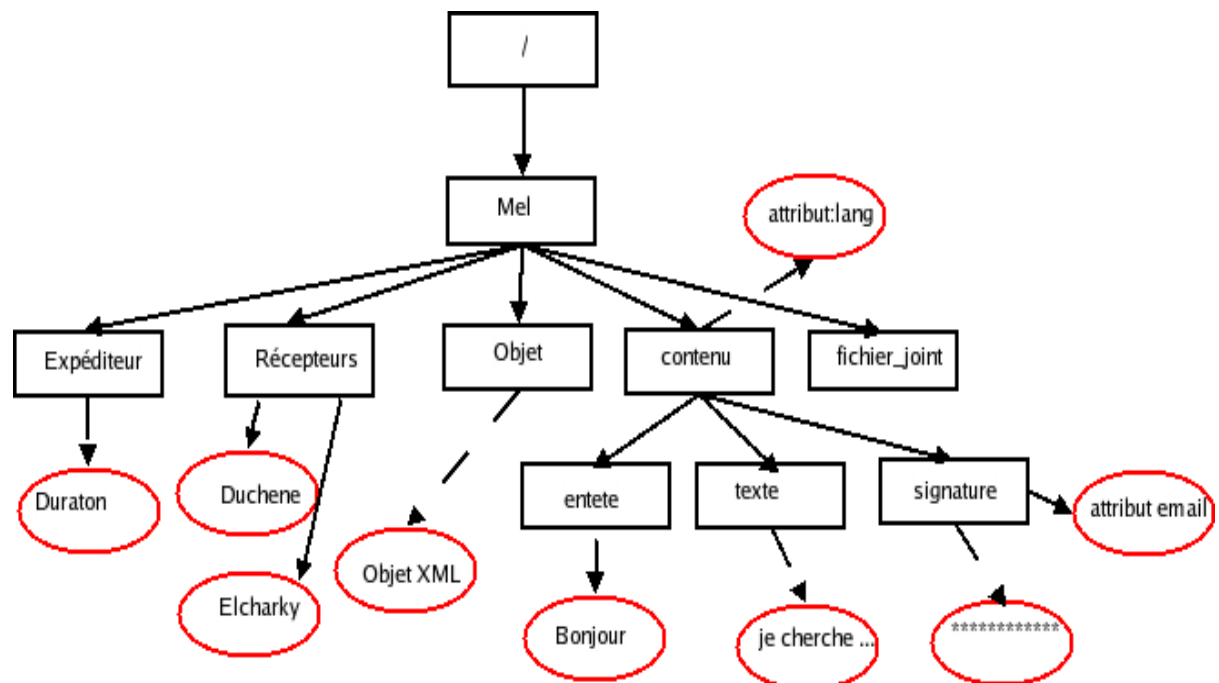
11

Extraction d'éléments du document XML

- Un document XML est matérialisé par un **arbre**
- Des **règles** d'extraction et de transformation sont définies sur les noeuds de l'arbre
- On désigne un noeud de l'arbre par un **chemin par rapport à la racine**

12






Arbre correspondant à l'exemple



13

Le processus de transformation XSLT

5 étapes

-  Un analyseur XML interprète le document XML et forme un arbre
-  L'arbre est passé à un processeur XSLT
-  Le processeur XSLT parcourt les nœuds de l'arbre XML et applique des **règles de transformation** aux nœuds désignés dans la feuille XSL.
-  Lorsque le processeur XSLT trouve une correspondance, il sort un fragment d'arbre.
-  L'ensemble résultat est ensuite sorti au format de sortie définie (html ou autre)

14

Règle de transformation

une **règle de transformation** nommée **template** est définie par la balise `xsl:template`

l'attribut **match** de cette balise définit le **motif** et indique à quel(s) type(s) d'éléments du document s'applique(nt) la règle

le contenu de la balise est le **corps** de la règle et définit le texte produit à chaque fois que la règle s'applique

15

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/
  1999/XSL/Transform">
  <xsl:template match="/">
    <H2> Exemple 2 - cours xsl </H2>
    <html>
      <HEAD>
        <TITLE>Bonjour</TITLE>
      </HEAD>
      <body>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Résultat



16

Exemple

Résultat

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

```
<H2> Exemple 3 - cours xsl </H2>
```

```
<html>
```

```
<HEAD>
```

```
<TITLE>Bonjour</TITLE>
```

```
</HEAD>
```

```
<body>
```

```
Salut <xsl:value-of select="mel/recepteurs"/>!
```

```
<br/>
```

```
signé l'expéditeur: <xsl:value-of select="mel/expediteur"/>!
```

```
</body>
```

```
</html>
```

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

Exemple 3 - cours xsl

Salut Duchene!

signé l'expéditeur: Duraton!

valeur de la balise recepteurs

valeur de la balise expediteur

17

Exemple

Résultat

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

```
<H2> Exemple 3 - cours xsl </H2>
```

```
<html>
```

```
<HEAD>
```

```
<TITLE>Bonjour</TITLE>
```

```
</HEAD>
```

```
<body>
```

```
Salut <xsl:value-of select="recepteurs"/>!
```

```
<br/>
```

```
signé l'expéditeur: <xsl:value-of select="expediteur"/>!
```

```
</body>
```

```
</html>
```

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

Exemple 3 - cours xsl

Salut !


signé l'expéditeur: !

Pourquoi n'y a-t-il pas d'affichage?

18

Règles implicites

2 règles implicites

 commencer le rendu du document XML source en partant de la racine et **itérer sur les nœuds fils**:

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

`<xsl:apply-templates>` : Indique au processeur XSL de traiter les éléments enfants directs en leur appliquant les règles définies dans la feuille XSL.

 copier tous les nœuds texte dans l'arbre résultat:

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

19

Règles explicites

On peut créer autant de règles explicites que voulues en utilisant la même syntaxe.

Syntaxe:

Une règle est définie par la balise `<xsl:template match=... >` la valeur de l'attribut **match** détermine à quel(s) nœud(s) de l'arbre source doit s'appliquer la règle à l'aide d'un filtre ou motif de sélection exprimé dans le langage XPATH.

La transformation s'effectue par un modèle traitant un nœud donné

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

actions

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

20

Exemple: Parcours de l'arbre itératif: xsl:for-each

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">
  <xsl:output method="html" encoding="ISO-8859-1"/>
  <xsl:template match="/mel">
    <html>
      <br/>
      expéditeur:
      <xsl:value-of select="expéditeur"/>
      <br/>
      <p> Les récepteurs sont: </p>
        <xsl:for-each select="recepteurs">
          <xsl:value-of select="."/><br/>
        </xsl:for-each>
      </html>
    </xsl:template>
  </xsl:stylesheet>
```

melEx_ite.xsl

21

Parcours de l'arbre récursif : xsl:apply-templates

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html" encoding="ISO-8859-1"/>
  <xsl:template match="/mel">
    <html>
      <xsl:apply-templates/>
    </html>
  </xsl:template>
  <xsl:template match="expéditeur">
    <br/>expéditeur:
    <xsl:value-of select="."/>
  </xsl:template>
  <xsl:template match="recepteurs">
    <br/><p/> Les récepteurs sont:
    <xsl:value-of select="."/><br/>
  </xsl:template>
</xsl:stylesheet>
```

22

Autre solution

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="html" encoding="ISO-8859-1"/>
<xsl:template match="/mel">
  <html>
    <xsl:apply-templates select="expediteur" />
  <p/> Les récepteurs sont:
    <xsl:apply-templates select="recepteurs"/>
  </html>

</xsl:template>
<xsl:template match="expediteur">
  <br/>
  expediteur:
  <xsl:value-of select="."/><br/>
</xsl:template>
<xsl:template match="recepteurs">
  <br/> <xsl:value-of select="."/>
</xsl:template>
</xsl:stylesheet>
```

L'attribut **select** permet de spécifier certains éléments enfants auxquels la transformation doit être appliquée

23

Fichier livres.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="livre.xsl"?>
<Livres>
  <livre isbn="681-3-048" titre="Comprendre XSLT" éditeur="O'Reilly"
  langue="français" nbPages='517'prix='40' >
    <Auteur idAuteur='10' prénom= "Cyril" nom= "Vincent"/>
  </livre>
  <livre isbn="2-7460-2004-1" titre="XML et les services Web"
  éditeur="ENI" langue="français" nbPages='460' prix='27' >
    <Auteur idAuteur='11' prénom= "Bernd" nom= "Amann"/>
    <Auteur idAuteur='12' prénom= "Philippe" nom= "Rigaux" />
  </livre>
  <livre isbn="2-011-55052-1" titre="XML" éditeur="ENI"
  langue="français" nbPages='400' prix='27' >
    <Auteur idAuteur='10' prénom= "Cyril" nom= "Vincent"/>
  </livre>
</Livres>
```

24

Affichage sous forme de tableau

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- affichage sous forme de tableau de la liste des Livres -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="ISO-8859-1"/>
<xsl:template match="Livres">
<html>
<head>
<title>Table Livre</title>
</head>
<body>
<table border="2" bgcolor="white">
<tr>
<th>isbn</th>
<th>Titre</th>
<th>éditeur</th>
<th>langue</th>
<th>nbPages</th>
<th>prix</th>
</tr>
```

25

```
<xsl:for-each select="livre">
<xsl:sort select="@isbn"/>
<tr>
<td> <xsl:value-of select = "@isbn" /> </td>
<td> <xsl:value-of select = "@titre" /> </td>
<td> <xsl:value-of select = "@éditeur" /> </td>
<td> <xsl:value-of select = "@langue" /> </td>
<td> <xsl:value-of select = "@nbPages" /> </td>
<td> <xsl:value-of select = "@prix" /> </td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Accès à
un attribut

Résultat

Home	Bookmarks	The Mozilla Organization	IUT	loisirs	mi
isbn	Titre	éditeur	langue	nbPages	prix
681-3-048	Comprendre XSLT	O'Reilly	français	517	40
2-7460-2004-1	XML et les services Web	ENI	français	460	27
2-011-55052-1	XML	ENI	français	400	27

26

Tri des éléments `xsl:sort`

- `<xsl:sort>` : élément qui permet d'effectuer un tri sur un jeu de noeuds. Il doit être placé
 - soit dans un élément `<xsl:for-each>`
 - soit dans un élément `<xsl:apply-templates>`.
- C'est un élément vide qui peut être appelé plusieurs fois pour effectuer un tri multicritères.
- Chaque appel à cet élément provoque un tri sur un champ spécifique, dans un ordre prédéfini. L'utilisation de cet élément est de la forme :

```
<xsl:sort select="noeud" data-type="text | number" order="ascending | descending" />
```

27

`xsl:sort`

`select` permet de spécifier un nœud comme clé de tri. (par défaut : .)

`data-type` correspond au type des données à trier.

Dans le cas où le type est *number*, les données sont converties puis triées. (par défaut : text)

`order` correspond à l'ordre de tri. Cet attribut peut être *ascending* ou *descending*. (par défaut : ascending)

28

```
<xsl:for-each select="livre">

  <xsl:sort select="auteur" order="descending" />

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

    <br/>

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

  </li>

</xsl:for-each>
```

29

Eléments du langage XSL: tests

Les conditions

```
<xsl:if test="motif"> ... </xsl:if> → si alors
<xsl:choose>
  <xsl:when test="motif"> .... </xsl:when>
  <xsl:when test="motif"> .... </xsl:when>
  <xsl:otherwise> .... </xsl:otherwise>
</xsl:choose>
```

30

Exemple

```
<xsl:for-each select="livre">
  <xsl:if test="Auteur/@prénom='Cyril' ">
    <xsl:value-of select="Auteur/@nom" />
  </xsl:if>
</xsl:for-each>
```

31

Les macros

XSLT permet la définition d'une macro (règle nommée), avec passage de paramètres

```
<xsl:template name="nom_macro">
  <xsl:param name="nom_param"/>
  .....
</xsl:template>
```

L'appel de la macro se fait par

```
<xsl:call-template name="nom-macro">
  <xsl:with-param name=" nom_param" select="valeur_param"/>
</xsl:call-template>
```

32

Exemple

```
<xsl:template match="/">
<html>
<!-- appel de la macro compteur -->
<xsl:call-template name="compteur">
<xsl:with-param name="iteration" select="0"/>
<xsl:with-param name="fin" select="3"/>
</xsl:call-template>
</html>
</xsl:template>
```

33

```
<!-- définition de la macro compteur -->
<xsl:template name="compteur">
<xsl:param name="iteration"/>
<xsl:param name="fin"/>
<xsl:if test="$iteration < $fin">
trace: <xsl:value-of select="'bonjour!'" />
<xsl:call-template name="compteur">
<xsl:with-param name="iteration" select="$iteration + 1"/>
<xsl:with-param name="fin" select="$fin"/>
</xsl:call-template>
</xsl:if>
</xsl:template>
```

34

Copie d'éléments

<xsl:copy> transformations </xsl:copy>

copie le noeud courant dans le document final, mais pas ses attributs, ni ses descendants; ces dernières sont à charge des **transformations**.

```
<A>
  <B>
    <E nb = "4" />
    <E nb= "3" />
  </B>
  <B>
    <E nb="1" />
  </B>
  <C>
    <F> a </F>
  </C>
  <E nb= "2" />
</A>
```

```
...
<xsl:output method="xml"
encoding="ISO-8859-1"/>
<xsl:template match="/">
  <xsl:apply-templates select="A/B[1]" />

  </xsl:template>
  <xsl:template match="A/B[1]">

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

```
<?xml version="1.0"
encoding="ISO-8859-1"?>
<B> Bonjour </B>
```

résultat

Selection du
premier fils B de
A

Fichier xsl

35

Instuction copy-of

- **<xsl:copy-of> transformations </xsl:copy-of>** reproduit un ensemble de noeuds récupéré par un select, avec la sous-arborescence de chacun

```
<A>
  <B>
    <E nb = "4" />
    <E nb= "3" />

  </B>
  <B>
    <E nb="1" />
  </B>
  <C>
    <F> a </F>
  </C>
  <E nb= "2" />
</A>
```

```
<xsl:template match="/">
  <xsl:apply-templates select="A/B[1]" />

  </xsl:template>
  <xsl:template match="A/B[1]">

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

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

<B>

  <E nb="4"/>

  <E nb="3"/>

</B>
```

36

Ajout d'élément et d'attribut

XSLT permet aussi d'ajouter des éléments et attributs XML au document final donc de créer un fichier XML à partir d'un autre fichier XML:

`<xsl:element name="nom"> transformations </xsl:element>`
crée un élément du `nom` spécifié dans le document final.

`<xsl:attribute name="nom"> valeur </xsl:attribute>`
crée et ajoute un attribut du `nom` et de la valeur spécifiés à l'élément courant.

37

```
<Stations>
  <Station>
    <nomStation>Venusa</nomStation>
    <capacite>350</capacite>
    <lieu>Guadeloupe</lieu>
    <region>Antilles</region>
    <tarif>1200.00</tarif>
  </Station>
  <Station>
    <nomStation>Farniente</nomStation>
    <capacite>200</capacite>
    <lieu>Seychelles</lieu>
    <region>Ocean Indien</region>
    <tarif>1500.00</tarif>
  </Station>
  <Station>
    <nomStation>Santalba</nomStation>
    <capacite>null</capacite>
    <lieu>Martinique</lieu>
    <region>Antilles</region>
    <tarif>1200.00</tarif>
  </Station>
  <Station>
    <nomStation>Passac</nomStation>
    <capacite>400</capacite>
    <lieu>Alpes</lieu>
    <region>Europe</region>
    <tarif>1000.00</tarif>
  </Station>
</Stations>
```

38

```
<?xml version="1.0"?>
<!--nom du fichier transXML_XML.xml -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" encoding="UTF-8"/>
  <xsl:template match="/">
    <Regions>
      <xsl:apply-templates select="Stations/Station/region"/>

    </Regions>
  </xsl:template>
  <!-- regle pour chaque Station -->

  <xsl:template match="region">
    <xsl:element name="MaRegion" >
      <xsl:attribute name="nom">
        <xsl:value-of select="."/>
      </xsl:attribute>
    </xsl:element>

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

39

Résultat de la transformation

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

<Regions>

  <MaRegion nom="Antilles"/>

  <MaRegion nom="Ocean Indien"/>

  <MaRegion nom="Europe"/>

</Regions>
```

40

Définition de variables

➤ `<xsl:variable name=Qname select=Expression />`

➤ Exemple

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

```
  <xsl:variable name="nb_mel" select="count(mel)" />
```

```
  <xsl:if test="$nb_mel mod 2 = 0">
```

nombre de mels pair

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

nombre de mels egal `<xsl:value-of select="$nb_mel" />`

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