| | |
|---|---|
| EISTI Ecole Internationale des Sciences du Traitement de l'Information | **TRAITEMENT ET ADMINISTRATION DES DONNEES**<br>**EXERCICES DE COMPLEMENTS DE XML + XSL** |
| Rédigé par : Matthias Colin et Hervé de Milleville | Ref : *SE- TAD-XSL-AVA* |
| A l'intention de : Elèves ingénieurs | Créé le : 13/10/2013 |



# Introduction

Dans cette feuille d'exercices, on s'intéresse à des éléments de XSL avancé :

- xsl:for-each-group
- de l'utilisation de fonctions (comme avg, sum, contains, substring-after, ...)
- du chargement dynamique d'un document xml avec la fonction document
- l'écriture de fonctions
- format-number
- result-document pour générer plusieurs sorties
- xsl:import et xsl:include pour factoriser (modularité et réutilisabilité) vos templates ou vos fonctions.

# 1. Utilisation de séquences

## *1.1 Questions*

1) Ecrire la fonction mediane qui prend en paramètre une liste de nombre décimaux et renvoie la valeur médiane.
2) Ecrire la fonction listecarre qui prend en paramètre un entier n et renvoie la liste des carrés des entiers entre 1 et n. Si n < 1 alors la fonction renvoie la séquence vide. On pensera à utiliser le constructeur XSL sequence.
3) Tester ces deux fonctions sur un jeu significatif de données.

# 2. Gestion de lycéens

## 2.1 Enoncé

Dans cet exercice, on s'intéresse à un arbre XML décrivant un ensemble de lycéens. Vous trouvez cet arbre en annexe de ce document. Vous pouvez aussi le télécharger sur le site AREL.

## 2.2 Questions

- Ecrire les deux fonctions XSLT 2.0 appelées identification qui permettent de retourner
  1) le nom et le prénom d'un lycéen si on ne connaît pas son sexe
  2) la civilité (Mlle ou Mr), le nom et le prénom si on connaît son sexe
- En utilisant les fonctions précédentes, écrire la feuille XSLT 2.0 qui permet d'afficher l'identification de tous les lycéens.
- En utilisant la fonction mediane, écrire la feuille XSLT 2.0 qui permet d'afficher la moyenne et la médiane des notes de tous les lycéens.
- En utilisant l'instruction xsl:for-each-group, écrire la feuille xsl 2.0 qui permet d'afficher la moyenne et la médiane des notes de tous les lycéens par sexe.

# 3. Gestion de trains

Dans cet exercice, on s'intéresse à un arbre XML décrivant un ensemble de trains. Vous trouvez cet arbre en annexe de ce document. Vous pouvez aussi le télécharger sur le site AREL.
Les exercices suivants peuvent être réalisés de manière indépendante en repartant des fichiers d'origine.

## 3.1 Démultiplication des entrées/sorties

1) Sortir le template ListeGares du de la feuille trainsHTML.xsl pour le placer dans une feuille gares.xsl. Lier ensuite les 2 feuilles pour obtenir le résultat d'origine.
2) Sortir la liste de gares du fichier trains.xml pour la placer dans un fichier gares.xml et laisser l'élément ListGares dans le fichier d'origine sans fils avec la mention du fichier contenant les gares en attribut. Modifier la feuille de style pour prendre en compte ce changement.
3) Créer une page HTML spécifique pour chaque train (par exemple train8369.html pour le train n°8369) et laisser dans la page principal un menu ou formulaire pour choisir parmi toutes ses pages.

## 3.2 Construction XML

1) Ecrire une feuille XSL qui extrait la liste des trains avec pour élément principal trains et un fils par train à l'identique du fichier d'origine.
2) La même chose mais le format de description des trains change :

<TGV num="8369" depart="Paris" destination="Bordeaux" nb_arrets="0" />

Le nom de l'élément est repris à partir de l'attribut type (TGV et TER dans l'exemple, la liste n'étant pas exhaustive). L'attribut nb_arrets donne le nombre d'arrêts intermédiaires. Vous essaierez de traiter les mentions départ et destination avec le même code.

## 3.3 Template et mode

Reprendre le template sur un train pour en faire deux versions :

1) Template simple : précisant uniquement le numéro, le type, le départ et la destination.
2) Template complet : comme à l'origine avec les horaires et la liste des arrêts.

En reprenant le 3ème exercice du paragraphe 3.1), vous utiliserez le premier template pour dresser une liste simple des trains dans le document principal et le deuxième template pour la page descriptive de chaque train.

## 3.4 Template et priorité

1) Reprendre la feuille de style de présentation des trains et enlever les templates sur les gares et les trains. Ecrire un template générique qui traite indifféremment ce type d'élément en affichant :

a. le nom de l'élément,
b. la liste des attributs avec pour chacun couple (nom, valeur),
c. traitement récursif des sous-éléments avec le même template.

2) Appliquer le template à la liste des gares et la liste des trains.
3) Réintégrer un template spécifique pour un train. Que constatez-vous ?
4) Réintégrer un template spécifique pour une gare ou une gare terminus. Tester les configurations suivantes :
   a) priorité automatique
   b) priorité à -1
   c) priorité à +1

# 4. Annexes

## *4.1 Annexe Lycéens*

```
<?xml version="1.0" encoding="iso-8859-1"?>
<lyceens>
	<lyceen>
		<nom>Nguyen</nom>
		<prenom>Sarah</prenom>
		<moyenne>18</moyenne>
		<sexe>F</sexe>
		<classe>Terminale</classe>
	</lyceen>
	<lyceen>
		<nom>Dupont</nom>
		<prenom>Christophe</prenom>

		<moyenne>5</moyenne>
		<sexe>M</sexe>
		<classe>première</classe>
	</lyceen>
	<lyceen>
		<nom>Saint Cyr</nom>
		<prenom>Christophe</prenom>

		<moyenne>20</moyenne>
		<sexe>M</sexe>
		<classe>Terminale</classe>
	</lyceen>
	<lyceen>
		<nom>Bulot</nom>
		<prenom>Pierre</prenom>

		<moyenne>12</moyenne>
		<sexe>M</sexe>
		<classe>Seconde</classe>
	</lyceen>
	<lyceen>
		<nom>de Vinci</nom>
		<prenom>Françoise</prenom>

		<moyenne>13</moyenne>
		<sexe>F</sexe>
		<classe>Terminale</classe>
	</lyceen>
	<lyceen>
		<nom>Amboise</nom>
		<prenom>Augustine</prenom>

		<moyenne>7</moyenne>
		<sexe>F</sexe>
```

```
                <classe>première</classe>
        </lyceen>
        <lyceen>
                <nom>Durand</nom>
                <prenom>Pierre</prenom>

                <moyenne>9</moyenne>
                <classe>Seconde</classe>
        </lyceen>
</lyceens>
```

## *4.2 Annexe Trains*

### 4.2.1 Le fichier train.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<Ligne nom="Paris-Bordeaux">
 <ListeGares>
   <GareTerminus nom="Paris" />
   <GareTerminus nom="Bordeaux" />
   <Gare nom="Saint Pierre des Corps" />
   <Gare nom="Tours" />
   <Gare nom="Dangé" />
   <Gare nom="Châtellerault" />
   <Gare nom="Futuroscope" />
   <Gare nom="Poitiers" />
   <Gare nom="Angoulême" />
 </ListeGares>
 <ListeTrains>
  <Train num="8369" type="TGV">
    <Depart gare="Paris" heure="10:30:00" />
    <Destination gare="Bordeaux" heure="14:00:00" />
  </Train>
  <Train num="8325" type="TGV">
    <Depart gare="Paris" heure="06:00:00" />
    <Destination gare="Bordeaux" heure="10:00:00" />
    <Arret gare="Saint Pierre des Corps" arrivee="07:00:00" depart="07:03:00" />
    <Arret gare="Châtellerault" arrivee="07:23:00" depart="07:25:00" />
    <Arret gare="Poitiers" arrivee="07:45:00" depart="07:50:00" />
    <Arret gare="Angoulême" arrivee="08:45:00" depart="08:50:00" />
  </Train>
  <Train num="18145" type="TER">
    <Depart gare="Poitiers" heure="14:40:00" />
    <Destination gare="Tours" heure="15:45:00" />
    <Arret gare="Futuroscope" arrivee="14:50:00" depart="14:57:00" />
    <Arret gare="Châtellerault" arrivee="15:10:00" depart="15:12:00" />
    <Arret gare="Dangé" arrivee="15:25:00" depart="15:27:00" />
  </Train>
  <Train num="8145" type="TGV">
    <Depart gare="Poitiers" heure="11:30:00" />
    <Destination gare="Paris" heure="13:15:00" />
    <Arret gare="Châtellerault" arrivee="11:45:00" depart="11:47:00" />
    <Arret gare="Saint Piere des Corps" arrivee="12:05:00" depart="12:10:00" />
  </Train>
</ListeTrains>
</Ligne>


<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:xs="http://www.w3.org/2001/XMLSchema" version="2.0">

  <!--
      Feuille de Style pour l'affichage détaillée d'une ligne de train
```

```
    auteur : Matthias Colin
    version : 1.0 (14/05/2010)
-->

<xsl:output method="html" encoding="utf-8" />

<!-- main template -->
<xsl:template match="Ligne">
 <html>
  <head>
   <title>Ligne de Train : <xsl:value-of select="@nom" /></title>
  </head>
  <body>
   <h1>Ligne de Train : <xsl:value-of select="@nom" /></h1>
   <xsl:apply-templates />
  </body>
 </html>
</xsl:template>

<!-- liste des gares -->
<xsl:template match="ListeGares">
                        <h2>Gares</h2>
                        <ul>
                        <xsl:for-each select="child::*">
                                <xsl:sort select="@nom" order="ascending"/>
                                <li><xsl:value-of select="@nom" />
                                        <xsl:if test="self::GareTerminus"> (terminus)</xsl:if>
                                </li>
                        </xsl:for-each>
                        </ul>
</xsl:template>
```

### 4.2.2 Le fichier trains.xsl

```
<!-- liste des TGV et TER -->
<xsl:template match="ListeTrains">
                        <h2>TGV</h2>
                        <ul>
                        <xsl:for-each select="Train[@type='TGV']">
                                <xsl:sort select="xs:time(Depart/@heure)" order="ascending"/>
                                <!-- NB: l'ordre lexicographique marche aussi -->
                                <li><xsl:apply-templates select="." /></li>
                        </xsl:for-each>
                        </ul>
                        <h2>TER</h2>
                        <ul>
                        <xsl:for-each select="Train[@type='TER']">
                                <xsl:sort select="xs:time(Depart/@heure)" order="ascending"/>
                                <li><xsl:apply-templates select="." /></li>
                        </xsl:for-each>
                        </ul>
</xsl:template>

<xsl:template match="Train">
 Train n°<xsl:value-of select="@num" />
 <ul>
  <li>Départ : <xsl:value-of select="Depart/@gare"/> Ã
    <xsl:value-of select="Depart/@heure"/></li>
  <li>Arrivée : <xsl:value-of select="Destination/@gare"/> Ã
    <xsl:value-of select="Destination/@heure"/></li>
  <xsl:if test="not(Arret)"><li>Ligne Directe</li></xsl:if>
  <xsl:for-each select="Arret">
   <xsl:sort select="xs:time(@arrivee)"/>
   <li>Arrêt : <xsl:value-of select="@gare"/></li>
```

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

</xsl:stylesheet>
```