Universités d'Aix-Marseille I & II **Technologies XML**
Master d'Informatique **Corrigé du partiel novembre 2011**
Année 2011-12

Exercice 1 : Documents XML et Applications XML *(3 points)*

1. Un document XML correspondant aux informations données est :

```
<?xml version="1.0" encoding="UTF-8"?>
<rss nom="L'Equipe.fr Actu Sport" date="Mercredi 02 Novembre 2011 à 15h31">
  <desc>Suivez ... sportifs.</desc>
  <item sport="Tennis" cat="ATP" lieu="Bale" date="Wed, 02 Nov 2011 15:29:00">
    <titre>Murray forfait !</titre>
    <desc>Alors qu'il avait ...  forfait...</desc>
  </item>
  <item sport="Moto" cat="Moto-GP" lieu="Valence"
        date="Wed, 02 Nov 2011 15:18:00">
    <titre>La dernière de Capirossi</titre>
    <desc>Après 22 ans... dimanche...</desc>
  </item>
  <item sport="Foot" cat="Euro" lieu="CRO" date="Wed, 02 Nov 2011 15:12:00">
    <titre>Lovren convoqué</titre>
    <desc>La Croatie ... a fait appel...</desc>
  </item>
  <item sport="Voile" cat="Vabre" date="Wed, 02 Nov 2011 15:00:00">
    <titre>C'est parti !</titre>
    <desc>Le départ... le départ...</desc>
  </item>
</rss>
```

2. Une DTD correspondant à ce document XML est :

```
<!ELEMENT rss      (desc, item*) >
<!ATTLIST rss
               nom   CDATA   #REQUIRED
               date  CDATA   #REQUIRED >
<!ELEMENT item     (titre,desc) >
<!ATTLIST item
               cat   CDATA   #REQUIRED
               lieu  CDATA   #IMPLIED
               sport CDATA   #REQUIRED
               date  CDATA   #REQUIRED >
<!ELEMENT titre    (#PCDATA) >
<!ELEMENT desc     (#PCDATA) >
```

Exercice 2 : Requêtes XPath *(4 points)*

1. //programme[@channel=”C1.telerama.fr”]
2. //channel[display-name=”TF1”]/@id
3. //programme[category=”série humoristique” and @channel=”C1.telerama.fr”]/credits/actor
4. //programme[@start<20111104120000 and @stop>20111104120000]
5. //programme[following-sibling : :programme/title=”Meteo”]
6. //programme[@channel=preceding : :channel[display-name=”TF1”]/@id]

Exercice 3 : Transformations XSLT *(6 points)*

1. La feuille de style suivante permet un affichage XHTML des programmes :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
    <head><title>Présentation des programmes</title></head>
    <body>
      <ul>
      <xsl:for-each select="tv/programme">
        <li>
          Titre : <xsl:value-of select="title"/><br/>
          Canal de diffusion : <xsl:value-of select="@channel"/><br/>
          Description : <xsl:value-of select="desc"/><br/>
          Durée : <xsl:value-of select="length"/>
                  <xsl:value-of select="length/@units"/> <br/>
        </li>
      </xsl:for-each>
      </ul>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

2. On veut remplacer le code

```
Canal de diffusion : <xsl:value-of select="@channel"/><br/>
```

de sorte à afficher le nom de la chaîne et pas son identifiant. Plusieurs solutions peuvent être envisagées, nous en présentons une. Ceci consiste à utiliser une requête XPath avec l'axe ”preceding” comme on l'a fait dans la question 5 de l'exercice 2. On obtient le nouveau code suivant :

```
Chaîne de diffusion :
     <xsl:variable name="identifiant" select="@channel"/>
     <xsl:value-of select="preceding::channel[@id=$identifiant]/display-name"/>
   <br/>
```

3. On ajoute la template suivante :

```
<xsl:template name="translate">
  <xsl:param name="date"/>

  <xsl:value-of select="substring($date,7,2)"/>
   
  <xsl:choose>
    <xsl:when test="substring($date,5,2)='01'">
      <xsl:value-of select="'Janvier'"/>
    </xsl:when>
    <xsl:when test="substring($date,5,2)='02'">
      <xsl:value-of select="'Février'"/>
    </xsl:when>
    <xsl:when test="substring($date,5,2)='03'">
      <xsl:value-of select="'Mars'"/>
    </xsl:when>
    <xsl:when test="substring($date,5,2)='04'">
      <xsl:value-of select="'Avril'"/>
    </xsl:when>
  </xsl:choose>
   
  <xsl:value-of select="substring($date,1,4)"/>
  <xsl:text> à </xsl:text>
  <xsl:value-of select="substring($date,9,2)"/>
  <xsl:text>h</xsl:text>
  <xsl:value-of select="substring($date,11,2)"/>
  <xsl:text>m</xsl:text>
  <xsl:value-of select="substring($date,13,2)"/>
</xsl:template>
```

Pou utiliser cette template, on modifie le programme xsl précédent en remplaçant `<xsl:value-of select="@start"/>` par :

```
<xsl:call-template name="translate">
  <xsl:with-param name="date" select="@start"/>
</xsl:call-template>
```

On fait de même avec @stop.

Exercice 4 : API SAX *(4 points)*

1. Une solution possible est la classe suivante. Notez que les import sont les mêmes que ceux utilisés pour la classe vue en cours, tout comme la fonction convertToFileURL et la classe MyErrorHandler.

```
public class SaxParserHTML
    extends DefaultHandler {
    private static String urlToParse;

    // Utilises pour stocker la valeur des attributs
    private String canal = new String();
    // Utilise pour savoir s'il faut ou non recopier le texte
    // sur la sortie standard (voir title)
    private int output;

    public SaxParserHTML () {
        super();
        this.output=0;
    }
    }

    static public void main(String[] args)
    {
      [comme SimpleSaxParser]
    }

    // debut du document
    public void startDocument()
        throws SAXException {
        System.out.println(
"<html>
 <head>
 <title>Programme TV</title>
 </head>
 <body>
 Liste des programmes :
 <ul>");
    }

    // debut de l'element
    public void startElement(String namespaceURI,
                             String localName, // local name
                             String rawName,   // qualified name
                             Attributes atts)
```

```
    throws SAXException
{
    // recuperation du nom de l'element
    String eltName = localName;
    if ("".equals(eltName)) eltName = rawName;  // namespaceAware = false
    // Traitement des elements programme
    // Nouvel element programme! On ecrit le debut, et on
    // stocke les attributs
    if ("programme".equals(eltName)) {
        System.out.println("<li>");
        for (int i=0; i <atts.getLength(); i++) {
           // recuperation du nom de l'attribut et de
           // sa valeur
           String attName = atts.getQName(i);
           if ("".equals(attName)) attName = atts.getQName(i);
           // on enregistre la valeur du canal
           if ("canal".equals(attName)) canal = atts.getValue(i);
        }
    }
    // Traitement des elements title
    if ("title".equals(eltName))  {
        System.out.print("Titre : ");
        // Activation de la recopie du texte
        output = 1;
    }
}

// Pour les noeuds textes
public void characters (char[] ch, int start, int length)
{
    if (output) {
       String text = new String (ch, start, length);
       System.out.print(text);
    }
}

// fin d'element
public void endElement(java.lang.String uri,
                       java.lang.String localName,
                       java.lang.String rawName)
    throws SAXException
{
    // Recuperation du nom de l'element
```

```
        String eltName = localName;
        if ("".equals(eltName)) eltName = rawName;
        // Traitement des elements programme
        // L'element title a ete traite, on peut afficher les attributs
        if ("programme".equals(eltName)) {
            System.out.println("Canal de diffusion : ",canal," <br/>");
            System.out.println("</li>");
        }
        // Traitement des elements title
        if ("title".equals(eltName))  {
            // Desactivation de la recopie du texte
            output = 0;
        }
    }

    // fin du document
    public void endDocument()
        throws SAXException {
        System.out.println("</ul> </body> </html>");
    }

    [... comme SimpleSaxParser ... ]

}
```

2. Pour ce traitement, on ajoute deux champs à la classe. Le premier, nommé date, et de type String, permet de stocker la valeur de ce second argument. Le second est un booléen nommé select, initialisé à faux, qui indique si l'élément programme en cours de traitement est ou non sélectionné. Dans la procédure startElement, dans le cas correspondant au traitement d'un élément programme, on compare la valeur des attributs start et stop au champ date, pour déterminer si le programme doit ou non être sélectionné. On met à jour la valeur du booléen select en fonction. Si le programme est sélectionné, on le traite comme précédemment, sinon on ne fait rien. Dans cette même procédure, pour le traitement des éléments title, si le booléen select est faux, on ne fait rien, sinon on fait comme avant. De la même façon, dans la procédure endElement, si le booléen select est faux, on ne fait rien, sinon on fait comme avant. Dans tous les cas, on finit cette procédure en mettant ce booléen à la valeur faux.
3. Cette opération est possible, bien que l'expression XPath associée utilise un axe ascendant. Pour cela, on ajoute à la classe un champ contenant une table d'association (identifiant, nom de chaîne). Cette table est construite lors du parcours des premiers éléments channel du document, et utilisée lors du traitement des éléments programmes.