Universités d'Aix-Marseille I & II **Technologies XML**
Master d'Informatique **Corrigé du partiel novembre 2010**
Année 2010-11 **Documents autorisés**

Exercice 1 : Documents XML et Applications XML *(4 points)*

Georges, un jeune propriétaire un poil maniaque, gère son compte bancaire via un fichier XML. Il écrit chaque mois un fichier XML correspondant à ses opérations du mois. Ce fichier commence par le montant initial du solde de son compte puis viennent toutes les opérations effectuées sur le compte (débit et dépôt). Un exemple représentatif décrivant (une partie) des données est donné dans le document 1 page 10. Il correspond au relevé de compte annoté présenté dans le document 2 page 11.

1. En respectant la structure du document 1, écrivez le fichier XML correspondant au relevé de compte présenté ci-dessous.

Juillet 2010				
Solde initial	131.40			
Chèque de SNCM	1344.88	3-7-10	salaire	salaire de Juin
Carte, SNCF	-88.00	6-7-10	vacances	billets de train Perpignan
Carte, DAB	-40.00	10-7-10	liquide	argent de poche
Chèque n° 1003, Les flots bleus	- 90.45	17-7-10	vacances	camping Perpignan
Chèque n° 1005, Bernard Merle	- 987.32	20-7-10	loyer	loyer de juillet

Corrigé :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<livre-de-comptes>
  <valeur-initiale>131.40</valeur-initiale>
  <depot type="cheque">
    <payeur>SNCM</payeur>
    <montant>1344.88</montant>
    <date>3-7-10</date>
    <description categorie="salaire">salaire de Juin</description>
  </depot>
  <retrait type="debit">
    <destinataire>SNCF</destinataire>
    <montant>88.00</montant>
    <date>6-7-10</date>
    <description categorie="vacances">billets de train Perpignan</description>
  </retrait>
  <retrait type="distributeur">
    <montant>40.00</montant>
    <date>10-7-10</date>
```

```
    <description categorie="liquide">argent de poche</description>
  </retrait>
  <retrait type="cheque" numero="1003">
    <destinataire>Les flots bleus</destinataire>
    <montant>90.45</montant>
    <date>17-7-10</date>
    <description categorie="vacances">camping Perpignan</description>
  </retrait>
  <retrait type="cheque" numero="1005">
    <payeur>Bernard Merle</payeur>
    <montant>987.32</montant>
    <date>20-7-10</date>
    <description categorie="loyer">loyer de Juillet</description>
  </retrait>
</livre-de-comptes>
```

2. Donnez une DTD correspondant aux documents XML ainsi produits.
   Corrigé :

```
<!ELEMENT livre-de-compte  (valeur-initiale,(depot|retrait)*)>
<!ELEMENT valeur-initiale  (#PCDATA)>
<!ELEMENT depot            (payeur,montant,date,description)>
<!ATTLIST depot     type CDATA #REQUIRED>
<!ELEMENT retrait          (destinataire?,montant,date,description)>
<!ATTLIST retrait
                 type   CDATA #REQUIRED
                 numero CDATA #IMPLIED>
<!ELEMENT payeur       (#PCDATA)>
<!ELEMENT destinataire (#PCDATA)>
<!ELEMENT montant      (#PCDATA)>
<!ELEMENT date         (#PCDATA)>
<!ELEMENT description  (#PCDATA)>
<!ATTLIST description categorie CDATA #REQUIRED>
```

3. Donnez un schéma XML correspondant aux documents XML ainsi produits.
   Corrigé :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <xsd:element name="livre-de-compte">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="valeur-initiale" type="xsd:string"/>
```

```
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
          <xsd:element ref="depot"/>
          <xsd:element ref="retrait"/>
        </xsd:choice>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="depot">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="payeur"/>
        <xsd:element ref="montant"/>
        <xsd:element ref="date"/>
        <xsd:element ref="description"/>
      </xsd:sequence>
      <xsd:attribute name="type" type="xsd:string" use="required"/>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="retrait">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="destinataire" minOccurs="0"/>
        <xsd:element ref="montant"/>
        <xsd:element ref="date"/>
        <xsd:element ref="description"/>
      </xsd:sequence>
      <xsd:attribute name="type" type="xsd:string" use="required"/>
      <xsd:attribute name="numero" type="xsd:positiveInteger" use="optional"/>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="payeur" type="xsd:string"/>
  <xsd:element name="destinataire" type="xsd:string"/>
  <xsd:element name="montant" type="xsd:decimal"/>
  <xsd:element name="date" type="xsd:date"/>
  <xsd:element name="description" type="xsd:string">
    <xsd:complexType>
      <xsd:attribute name="categorie" type="xsd:string" use="required"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Exercice 2 : API Java SAX et DOM *(5 points)*

Georges, notre jeune propriétaire de l'exercice 1 souhaite maintenant profiter de la puissance d'XML pour gérer son compte facilement.

Remarque : Si des fonctions ou des procédures de votre code sont strictement identiques aux éléments correspondants des programme Java vus en TP, vous pouvez vous contenter d'y faire référence sans tout recopier.

1. Ecrire un programme Java utilisant l'API SAX qui, à partir d'un fichier d'entrée *Livre_de_compte.xml* (voir document 1 page 10), affiche les montants et dates des retraits de type "distributeur".

   Corrigé : Nous allons ajouter dans la classe des variables permettant d'indiquer si l'élément que nous sommes en train de parcourir est ou non un retrait au distributeur. On obtient les modifications suivantes :

```
private boolean print;
private boolean save_montant;
private boolean save_date;
private String montant;
private String date;

public SimpleSaxParser () {
  super();
  this.print=false;
  this.save_montant=false;
  this.save_date=false;
  this.montant="";
  this.date="";
}
public void startElement(String namespaceURI,
                         String localName, // local name
                         String rawName,   // qualified name
                         Attributes atts)
throws SAXException
{
  String eltName = localName;
  if ("".equals(eltName)) eltName = rawName;  // namespaceAware = false
  if ("retrait".equals(eltName)){
    for (int i=0; i <atts.getLength(); i++) {
      String attName = atts.getQName(i);
      if ("".equals(attName)) attName = atts.getQName(i);
      if ("type".equals(attName) && "distributeur".equals(atts.getValue(i)))
        { print = true; }
      }
```

```
  }
  else if ("montant".equals(eltName) && print) {
       save_montant = true;
       }
  else if ("date".equals(eltName) && print) {
       save_date = true;
       }
}

public void characters (char[] ch, int start, int length)
{
  String text = new String (ch, start, length);
  if (save_montant)  montant = text;
  if (save_date) date = text;
}

public void endElement(java.lang.String uri,
                       java.lang.String localName,
                       java.lang.String rawName)
throws SAXException
{
  String eltName = localName;
  if ("".equals(eltName)) eltName = rawName;
  if ("retrait".equals(eltName) && print) {
    System.out.print("Retrait de "+montant+" effectue le "+date+"\n");
    print = false;
  }
  else if ("montant".equals(eltName)) {
    save_montant = false;
    }
  else if ("date".equals(eltName)) {
    save_date = false;
    }
}
```

2. Ecrire un programme Java utilisant l'API DOM qui, à partir d'un fichier d'entrée *Livre_de_compte.xml*, génère un nouveau fichier XML *Livre_de_compte-update.xml* dont la valeur initiale est le nouveau solde (donc la valeur initiale précédente plus les dépôts, moins les retraits). Ce nouveau fichier ne contient pas d'opération bancaire.

Corrigé : Nous effectuons d'abord un arbre DOM du document, puis nous le parcourons pour calculer la nouvelle valeur. Enfin, nous construisons un nouveau document contenant seulement l'élément avec la nouvelle valeur, et nous renvoyons ce document. Nous présentons ci-dessous les définitions correspondant au calcul du nouveau montant en DOM.

```
public static float explore(Node domNode, boolean mode) {
int type = domNode.getNodeType();
String nodeName = domNode.getNodeName();
float montant = 0;

// Cas d'un element
if (type == 1) {
    // Cas de valeur-initiale
    if ("valeur-initiale".equals(nodeName)) {
    NodeList child = domNode.getChildNodes();
    for (int i = 0; i < child.getLength(); i++)
       montant+=explore(child.item(i), true);
    }
    // Cas de depot
    else if ("depot".equals(nodeName)) {
    NodeList child = domNode.getChildNodes();
    for (int i = 0; i < child.getLength(); i++)
       montant+=explore(child.item(i), false);
    }
    // Cas de retrait
    else if ("retrait".equals(nodeName)) {
    NodeList child = domNode.getChildNodes();
    for (int i = 0; i < child.getLength(); i++)
       montant-=explore(child.item(i), false);
    }
    // Cas de montant
    else if ("montant".equals(nodeName)) {
    NodeList child = domNode.getChildNodes();
    for (int i = 0; i < child.getLength(); i++)
       montant=explore(child.item(i), true);
    }
    // Autres elements
    else {
    NodeList child = domNode.getChildNodes();
    for (int i = 0; i < child.getLength(); i++)
       montant+=explore(child.item(i),mode);
    }
}
// Cas d'un texte
else if (type == 3) {
    if (mode) {
     String text = domNode.getNodeValue().trim();
     float f = Float.valueOf(text).floatValue();
```

```
     montant = f;
    }
}
else {
    NodeList child = domNode.getChildNodes();
    for (int i = 0; i < child.getLength(); i++)
       montant+=explore(child.item(i),mode);
}
return montant;
}
```

Exercice 3 : Requêtes XPath *(5 points)*

On considère les stocks d'un magasin d'articles pour la maison. Les informations associées à un produit sont les suivantes :

- nom
- nom du fournisseur
- prix d'achat
- prix de vente
- nombre d'unités restantes
- nombre d'articles vendus au cours du mois dernier

Un exemple de document XML représentant trois articles est donné dans le document 3 page 11. Les requêtes que vous écrirez par la suite doivent naturellement être valables pour n'importe quel document XML suivant la même spécification.

Donnez des requêtes XPath pour sélectionner les éléments suivants. Vous écrirez des requêtes purement descendantes lorsque cela est possible.

(a) les articles dont le prix de vente est inférieur à 10 euros.
Corrigé :
`/shop/article[@prix_vente<=10]`

(b) les éléments "nom" des articles fournis par l'établissement "meubles morel".
Corrigé :
`/shop/article[fournisseur=''meubles morel'']/nom`

(c) les éléments "fournisseur" des produits dont le nombre restant en stock est inférieur à ce qui a été vendu le mois dernier.
Corrigé :
`/shop/article[qtity<nb_ventes]/fournisseur`

(d) les éléments "nom" des articles pour lesquels la marge est inférieure à 20% (la marge est définie comme le rapport (prix de vente - prix d'achat)/prix de vente).
Corrigé :
`/shop/article[@prix_achat >= 0.8*@prix_vente]/nom`

(*e*) deux requêtes permettant respectivement de calculer le chiffre d'affaire du mois dernier (somme de toutes les ventes), et le bénéfice du mois dernier.
Corrigé :
`sum(/shop/article/nb_ventes)` → donne le nombre total de ventes
`/shop/article[nom="table basse"]/nb_ventes/text()*/shop/article[nom="table basse"]/@prix_vente` → donne le chiffre d'affaire pour le produit "table basse"
Il n'était en fait pas possible d'obtenir le chiffre d'affaire global.

Exercice 4 : Transformations XSLT *(6 points)*

On utilise un modèle plus détaillé de stock intégrant une description du magasin (nom, adresse) et des descriptions des produits accompagnées d'une image. Le document 4 page 12 donne un exemple de document XML obtenu pour ce type de représentation.
Dans la suite, vous n'avez pas besoin de préciser les références xmlns complètes.

Affichage XSL *(4 points)*

1. Proposez une feuille de style XSL permettant un affichage (X)HTML du catalogue du magasin à destination des clients. Cette page donnera d'abord le nom du magasin avec son adresse puis, pour chaque produit, sa description, son illustration, son prix de vente, et enfin affichera le nombre d'éléments en stock. (Balise pour insérer une image en html)

2. Modifiez votre feuille de style pour que le contenu de la page soit sensible au nombre d'éléments en stock :
   - "Produit indisponible" si le stock est vide,
   - "Attention, plus que x éléments en stock" s'il reste $0 < x \leq 5$ éléments en stock, (la valeur de x doit être insérée dans le message)
   - "Produit disponible" sinon.

Corrigé :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <!-- Ainsi le context initial est l'élément shop-->
  <xsl:template match="/shop">
   <html>
     <head>
      <title>Catalogue du magasin</title>
     </head>
     <body>
     <p>
       <xsl:value-of select="boutique/nom"/>
       <text>, situé au </text>
       <xsl:value-of select="boutique/adresse"/>
```

```
      </p>
      <p>
        <ul><xsl:apply-templates select="article"/></ul>
      </p>
      </body>
    </html>
  </xsl:template>

  <xsl:template match="article">
    <li>
    <xsl:value-of select="nom"/> : <xsl:value-of select="description"/>.
    <img src="{image}"/>
    Vendu au tarif de <xsl:value-of select="@prix_vente"/> euros.

    <xsl:if test="qtity='0'">
       Produit indisponible
    </xsl:if>
    <xsl:if test="qtity>0 and qtity<6">
       Attention, plus que <xsl:value-of select="qtity"/> produits en stock
    </xsl:if>
    <xsl:if test="qtity>5">
       Produit disponible
    </xsl:if>
    </li>
  </xsl:template>
</xsl:stylesheet>
```

Mise en forme *(2 points)* On considère une entreprise mère qui possède plusieurs filiales. Chaque filiale envoyant ses chiffres du mois sous la forme d'un document XML (selon le format simple), on souhaite créer une feuille de style XSL permettant de regrouper ces documents. Les filiales vendant certains produits en commun, il faut alors regrouper les chiffres. On supposera que l'on peut identifier un produit par son nom, et qu'alors il a les mêmes prix d'achat, prix de vente, fournisseurs dans toutes les filiales. L'entrée correspondant à ce produit dans le document produit doit donc indiquer le nombre total de produits en stock, et le nombre total de ventes.

On consière un document "maître" qui recense les documents produits par les filiales :

```
<liste>
  <item>
    <nom>Mille Merveilles</nom>
    <path>mm.xml</path>
  </item>
  <item>
    <nom>Paris Direct</nom>
    <path>paris.xml</path>
```

```
    </item>
    <item>
      <nom>Origine Marseille</nom>
      <path>OM.xml</path>
    </item>
  </liste>
```

Ecrivez une feuille de style applicable à ce document maître et permettant de construire un nouveau document XML respectant le format simple de présentation des produits correspondant aux chiffres obtenus pour l'entreprise toute entière.

Indication : Vous pouvez utiliser en XSLT l'opérateur XPath `fn:distinct-values` qui s'applique à une liste de noeuds "texte" et qui permet de filtrer les doublons.

Corrigé :

La solution complète est un peu longue, je ne la donne pas ici...

Annexes

Document 1 : Livre de compte

```
<?xml version="1.0" encoding="iso-8859-1"?>
<livre-de-comptes>
  <valeur-initiale>22.00</valeur-initiale>
  <depot type="cheque">
    <payeur>SNCM</payeur>
    <montant>1344.88</montant>
    <date>3-6-10</date>
    <description categorie="salaire">salaire de Mai</description>
  </depot>
  <retrait type="cheque" numero="980">
    <destinataire>Sac à pof</destinataire>
    <montant>132.77</montant>
    <date>17-6-10</date>
    <description categorie="loisirs">equipement d'escalade</description>
  </retrait>
  <retrait type="cheque" numero="978">
    <payeur>Bernard Merle</payeur>
    <montant>987.32</montant>
    <date>21-6-10</date>
    <description categorie="loyer">loyer de Juin</description>
  </retrait>
  <retrait type="distributeur">
    <montant>40.00</montant>
    <date>24-6-10</date>
    <description categorie="liquide">argent de poche</description>
```

```
  </retrait>
  <retrait type="debit">
    <destinataire>Pizzeria Florence</destinataire>
    <montant>36.86</montant>
    <date>26-6-10</date>
    <description categorie="repas">restau avec Greg</description>
  </retrait>
  <depot type="direct">
    <payeur>Assurance maladie</payeur>
    <montant>137.32</montant>
    <date>21-7-10</date>
    <description categorie="remboursement">Tamiflu</description>
  </depot>
</livre-de-comptes>
```

Document 2 : Relevé de compte

Juin 2010				
Solde initial	22.00			
Chèque de SNCM	1344.88	3-6-10	salaire	salaire de Mai
Chèque n° 980, Sac à pof	- 132.77	17-6-10	loisirs	equipement d'escalade
Chèque n° 978, Bernard Merle	- 987.32	21-6-10	loyer	loyer de juin
Carte, DAB	-40.00	24-6-10	liquide	argent de poche
Carte, Pizzeria Florence	-36.86	26-6-10	repas	restau avec Greg
Depot, Assurance maladie	137.32	29-6-10	remboursement	Tamiflu

Document 3 : Exemple de stock simplifié

```
<shop>
  <article prix_achat="150" prix_vente="225">
    <nom>table basse</nom>
    <fournisseur>meubles morel</fournisseur>
    <qtity>15</qtity>
    <nb_ventes>3</nb_ventes>
  </article>
  <article prix_achat="3.5" prix_vente="8">
    <nom>bougie fleurs</nom>
    <fournisseur>TPC</fournisseur>
    <qtity>133</qtity>
    <nb_ventes>43</nb_ventes>
  </article>
  <article prix_achat="40" prix_vente="49.9">
    <nom>tabouret bar</nom>
```

```
    <fournisseur>Artisans et Cie</fournisseur>
    <qtity>2</qtity>
    <nb_ventes>10</nb_ventes>
  </article>
</shop>
```

Document 4 : Exemple de stock avec descriptions

```
<shop>
  <boutique>
    <nom>La maison pour tous</nom>
    <adresse>6, rue Saint Férréol, 13006 Marseille</adresse>
  </boutique>
  <article prix_achat="150" prix_vente="225">
    <nom>table basse</nom>
    <fournisseur>meubles morel</fournisseur>
    <description>Réalisée en chêne massif, cette table ornera à merveille
     votre salon de style ancien. Disponible dans différents coloris,
     elle vous sera livrée gratuitement en 48 heures.
    </description>
    <image>data/images/capture332.jpg</image>
    <qtity>15</qtity>
    <nb_ventes>3</nb_ventes>
  </article>
  <article prix_achat="3.5" prix_vente="8">
    <nom>bougie fleurs</nom>
    <fournisseur>TPC</fournisseur>
    <description>Bougie en cire végétale. Produit entièrement naturel...
     ...et s'intègre dans les autres accessoires de cette collection.
    </description>
    <image>data/images/capture007.jpg</image>
    <qtity>133</qtity>
    <nb_ventes>43</nb_ventes>
  </article>
</shop>
```