Simple API for XML

Fabrice Rossi

14 novembre 2002

Les exercices présentés dans ce document sont construits en partie grâce aux exercices de [1]. Pour la plupart des exercices, il est vivement conseillé de dessiner l'automate fini (ou à pile) qui donne l'algorithme de traitement du document avant de se lancer dans la programmation proprement dite.

1 Instructions spécifiques au CRIO UNIX

Au début de chaque séance :

1. dans chaque terminal, utilisez tcsh (en tapant `tcsh`)
2. effectuez les réglages nécessaires au bon fonctionnement des différents programmes utilisés, en tapant `source /home/perm/ufrmd/rossi/xml/env.csh`

2 Traitements d'un flux avec mémoire limitée

Les exercices de cette section ont pour point commun de demander un traitement relativement simple d'un document XML, au sens où le résultat est obtenu au fur et à mesure de l'analyse du document. Les mémorisations à effectuer pendant le traitement sont réduites au minimum.

Exercice 2.1

Écrire un `ContentHandler` qui affiche le contenu d'un fichier carnet d'adresses basé sur la DTD carnet définie dans l'exercice 2.1 de [1]. On proposera un affichage formaté.

Exercice 2.2

Écrire un `ContentHandler` qui affiche le contenu d'un fichier XML quelconque en procédant de la façon suivante. Pour afficher un élément, on réalise les étapes suivantes :
– on affiche le nom de l'élément
– on affiche ses attributs sous forme d'une liste `nom=valeur` (sans délimiteur pour la valeur)
– on affiche son contenu, en décalant celui-ci d'un espace
Par exemple, si on considère le fichier XML suivant :

demo-print.xml

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <demo>
3   <a x="y" z="w">
4     <b>Texte</b>
5     <c t="hop"/>
6     Encore du texte
7     <d><e/></d>
8   </a>
9 </demo>
```

le programme doit afficher :

```
1 demo
2  a
```

```
- x=y
- z=w
 b
  Texte
 c
 - t=hop
 Encore du texte
 d
  e
```

Dans cet affichage, on a radicalement simplifié les problèmes liés aux blancs ignorables, au point d'ailleurs de tricher pour l'affichage de "Encore du texte" qui devrait tenir compte des passages à la ligne. Dans le `ContentHandler`, on proposera des techniques permettant de mettre en œuvre les simplifications opérées dans l'exemple.

Exercice 2.3

On considère la DTD suivante :

play.dtd

```
<!-- DTD for Shakespeare    J. Bosak    1994.03.01, 1997.01.02 -->
<!-- Revised for case sensitivity 1997.09.10 -->
<!-- Revised for XML 1.0 conformity 1998.01.27 (thanks to Eve Maler) -->

<!ENTITY amp "&#38;">
<!ELEMENT PLAY     (TITLE, FM, PERSONAE, SCNDESCR, PLAYSUBT, INDUCT?,
                                          PROLOGUE?, ACT+, EPILOGUE?)>
<!ELEMENT TITLE    (#PCDATA)>
<!ELEMENT FM       (P+)>
<!ELEMENT P        (#PCDATA)>
<!ELEMENT PERSONAE (TITLE, (PERSONA | PGROUP)+)>
<!ELEMENT PGROUP   (PERSONA+, GRPDESCR)>
<!ELEMENT PERSONA  (#PCDATA)>
<!ELEMENT GRPDESCR (#PCDATA)>
<!ELEMENT SCNDESCR (#PCDATA)>
<!ELEMENT PLAYSUBT (#PCDATA)>
<!ELEMENT INDUCT   (TITLE, SUBTITLE*, (SCENE+|(SPEECH|STAGEDIR|SUBHEAD)+))>
<!ELEMENT ACT      (TITLE, SUBTITLE*, PROLOGUE?, SCENE+, EPILOGUE?)>
<!ELEMENT SCENE    (TITLE, SUBTITLE*, (SPEECH | STAGEDIR | SUBHEAD)+)>
<!ELEMENT PROLOGUE (TITLE, SUBTITLE*, (STAGEDIR | SPEECH)+)>
<!ELEMENT EPILOGUE (TITLE, SUBTITLE*, (STAGEDIR | SPEECH)+)>
<!ELEMENT SPEECH   (SPEAKER+, (LINE | STAGEDIR | SUBHEAD)+)>
<!ELEMENT SPEAKER  (#PCDATA)>
<!ELEMENT LINE     (#PCDATA | STAGEDIR)*>
<!ELEMENT STAGEDIR (#PCDATA)>
<!ELEMENT SUBTITLE (#PCDATA)>
<!ELEMENT SUBHEAD  (#PCDATA)>
```

Elle peut être utilisée pour représenter en XML une pièce de théâtre (racine du document : `PLAY`). La structure de la pièce est donnée par les éléments `ACT` (un acte) et `SCENE` (une scène). La pièce, les actes et les scènes possèdent tous un titre (`TITLE`). De plus, la pièce peut avoir une introduction (`INDUCT`) ou un prologue (`PROLOGUE`), ainsi qu'un épilogue (`EPILOGUE`). Les actes peuvent aussi avoir un prologue et/ou un épilogue. Les éléments facultatifs ont un titre (`TITLE`). Tous les éléments de structure peuvent avoir un sous-titre (`SUBTITLE`). Notons pour finir que chaque vers de la pièce est représenté par le contenu d'un élément `LINE`.

Questions :

1. Écrire un `ContentHandler` qui affiche le nombre d'actes, le nombre de scènes et le nombre de vers d'une pièce au format `play`.
2. La liste des personnages d'une pièce est représentée dans le format `play` par le contenu des éléments `PERSONA`. Écrire un `ContentHandler` qui affiche la liste des personnages d'une pièce au format `play`.
3. Écrire un `ContentHandler` qui affiche le plan d'une pièce au format `play`, c'est-à-dire son titre, la liste des actes et la liste des scènes (le tout organisé comme une table des matières). Avant d'écrire le code proprement dit, donner le schéma d'un automate fini permettant de mettre en œuvre le traitement demandé.

On testera les programmes réalisés sur les pièces de Shakespeare au format `play`, en particulier sur `pericles.xml`, `hen_viii.xml`, et `hen_iv_2.xml`, ainsi que sur des exemples faisant apparaître toute les possibilités de la DTD.

3 Traitements avec mémorisation

Exercice 3.1

Dans une pièce de théâtre au format `play` (cf exercice 2.3), le personnage qui prononce une tirage (`SPEECH`) est identifié comme le `SPEAKER` de cette tirade.

Questions :

1. Écrire un `ContentHandler` qui affiche la liste des scènes d'une pièce au format `play` en donnant pour chaque scène la liste des personnages qui parlent dans cette scène.
2. Écrire un `ContentHandler` qui affiche la liste des `SPEAKER`s d'une pièce au format `play`, classés par ordre alphabétique.
3. Écrire un `ContentHandler` qui affiche la liste des `SPEAKER`s d'une pièce au format `play` en indiquant pour chacun d'eux le nombre total de vers prononcés (ainsi que le nombre de tirages).
4. Écrire un `ContentHandler` et un programme `Java` permettant une recherche ciblée, c'est-à-dire l'affichage des vers d'un personnage donné contenant un mot donné. On numérotera les `LINE`s et on affichera le numéro d'ordre des vers trouvés.

Exercice 3.2

On propose de représenter un bon de commande sous la forme d'un document XML dont voici un exemple :

facture.xml

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <facture>
3  <client>
4  <nom>Doe</nom>
5  <prénom>John</prénom>
6  <adresse>
7  <numéro>15</numéro>
8  <rue>de la gare</rue>
9  <ville code-postal="75016">PARIS</ville>
10 <pays état="">France</pays>
11 </adresse>
12 <téléphone>01 44 37 28 72</téléphone>
13 </client>
14 <contenu>
15 <produit référence="38381199">
16 <description>CD Linkin Park (Hybrid Theory)</description>
17 <prix réduction="non" monnaie="FF">129</prix>
18 <quantité>1</quantité>
19 </produit>
```

```
<produit référence="44000392">
<description>DVD Muse (live au Zénith de Paris)</description>
<prix réduction="non" monnaie="Euro">21</prix>
<quantité>1</quantité>
</produit>
</contenu>
</facture>
```

Questions :

1. Proposer un schéma ou une DTD (au choix) représentant une structure de facture compatible avec le fichier exemple.
2. Programmer un `ContentHandler` qui calcule le montant total de la facture exprimé en euros (avec 1 €=6.55957 FF), en effectuant les arrondis correctement pour se limiter aux centimes d'euro.

Exercice 3.3

On reprend dans cet exercice le dialecte inspiré de MathML proposé dans l'exercice 3.2 de [1]. Écrire un `ContentHandler` qui évalue une expression mathématique respectant le format proposé. Pour ce faire, on suggère d'écrire un automate de calcul disposant de deux piles, une pour les opérandes, une pour les opérateurs.

4 Références croisées

Exercice 4.1

On veut écrire un `ContentHandler` qui affiche le contenu d'un fichier carnet d'adresses basé sur la DTD carnet professionnel définie dans l'exercice 2.2 de [1]. Cette DTD est basée sur un système ID/IDREF qui associe une adresse à un contact. Pour simplifier le traitement, on suppose que les adresses (ID) sont écrites dans le fichier avant les contacts (IDREF). Écrire alors un `ContentHandler` qui affiche la liste des contacts contenus dans le fichier en indiquant pour chaque contact son adresse effective. On utilisera une structure `Java` (par exemple un `Map`) pour faciliter le stockage des adresses et leur récupération. On donnera l'automate fini utilisé pour analyser le fichier.

Exercice 4.2

Écrire un `ContentHandler` qui vérifie la cohérence d'une structure d'entreprise décrite par un fichier XML au format proposé à l'exercice 3.1 de [1] (question 1). On proposera deux modes de vérification : un mode strict qui force une structure d'arbre (exactement un seul supérieur pour chaque employé sauf pour le patron qui est la racine de l'arbre) et un mode relâché qui interdit simplement les cycles. Le `ContentHandler` devra afficher la première erreur rencontrée (ou un message indiquant que tout va bien).

Références

[1] Fabrice Rossi. Dtd et schémas. Recueil d'exercices, Université Paris-IX Dauphine, 2002. `http://apiacoa.org/teaching/xml/exercices/dtd-et-schemas.pdf`.