Licence 3 d'Informatique — ASXML — TP 2

Début : mardi 20 février 2024

0 Fichiers à récupérer

La suite des travaux pratiques vise à vous familiariser avec le langage XSLT[1], utilisé pour les transformations de documents XML[2]. Le processeur XSLT 2.0 que nous allons utiliser est la *home edition* du programme Saxon[3]. Nous vous rappelons que pour l'utiliser, il est nécessaire d'effectuer la commande suivante au préalable :

```
export CLASSPATH=${CLASSPATH}:/opt/saxon/saxon-he-11.5.jar
```

comme nous l'avons entrevu lors de la précédente série de travaux pratiques. Après cette commande, l'utilisation de base de ce programme est décrite dans l'annexe L de votre polycopié, déjà diffusée en cours. Les fichiers utilisés ci-après soit vous ont déjà été distribués pour la précédente séance de travaux pratiques, soit ont été réunis dans le nouveau fichier d'archive .tar[4] for-lc-2.tar, à récupérer sur le cours « Programmation fonctionnelle, scripts --- XML » de moodle, dans le répertoire « XML - semestre 6 > Travaux pratiques 2024 ». Outre le présent fichier lc-2.pdf, disponible sous le format PDF[5], ce fichier d'archive contient dix-huit fichiers répartis en cinq sous-répertoires :

— le sous-répertoire saxon — destiné à ceux d'entre vous qui travaillent sur leur propre matériel — contient uniquement le fichier SaxonHE11-5J.zip : c'est un fichier d'archive au format ZIP ; après ouverture, installez son contenu là où vous avez déjà placé des bibliothèques utilisables par des programmes en Java ;

— les quatre autres sous-répertoires sont included, css, verlaine et sf-1, dont les utilités respectives apparaîtront dans les §§ 1.1, 1.2, 2 & 3.

1 Démonstrations

1.1 XInclude

L'inclusion du contenu d'un fichier XML au moyen d'entités externes reste un mécanisme « greffé » car reposant sur l'emploi d'une balise `DOCTYPE`, éventuellement factice, c'est-à-dire définissant un type factice de document, sans se rapporter à des définitions d'éléments ni à

1. *e**X**tensible **S**tylesheet **L**anguage **T**ransformations.*
2. *e**X**tensible **M**arkup **L**anguage.*
3. Tous renseignements complémentaires au sujet de ce programme et de ses différentes versions peuvent être trouvés sur le site *Web* `http://www.saxonica.com`.
4. ***T**ape **AR**chive.*
5. ***P**ortable **D**ocument **F**ormat.*

un fichier DTD[6], auquel cas cette balise `DOCTYPE` factice ne sert en réalité qu'à introduire des entités simples au moyen de balises `ENTITY`[7] :

```
<!DOCTYPE dummy [<!ENTITY ...> ...]>
```

De plus, ces deux balises `DOCTYPE` et `ENTITY` ne sont pas du XML « pur ». Nous montrons les bases de la nouvelle norme XInclude, qui n'est cependant pas prise en compte par tous les outils. Pour bien voir ce qui se passe, utiliser d'abord le programme `xmllint` comme d'habitude sur le fichier several-i.xml, puis avec l'option `--xinclude`. Les cinq fichiers utilisés, dans le sous-répertoire included sont :

several-1.xml	several-3.txt	several.xml
several-2.xml	several-i.xml	

Vous y découvrirez l'espace de noms de XInclude, accessible par le préfixe `xi`, les éléments `xi:include` et `xi:fallback`, ce dernier pouvant être utilisé comme un fils du précédent et traitant les cas où le fichier à insérer n'existe pas ou n'est pas exploitable. Remarquez aussi la possibilité d'insérer non seulement un document XML — l'attribut `parse` de l'élément `xi:include` est lié par défaut à la valeur `xml` —, mais aussi un fichier en texte simple. Notez aussi que le fichier silent.xml — cité dans le courant du fichier several-i.xml — n'existe pas et que l'attribut `xml:base` n'est pas reconnu par tous les systèmes d'exploitation.

1.2 Documents XML et fichiers CSS

La seconde démonstration va montrer comment l'affichage de documents XML dans un *Web browser* peut être contrôlé par des directives du langage CSS[8]. Notez toutefois que ces directives ne permettent qu'un formatage relativement rudimentaire, bien en-deçà de qu'il est possible d'atteindre au moyen d'une feuille de style XSLT construisant des fichiers dans les langages HTML[9] et XHTML[10]. Les fichiers utilisés et qui n'ont pas déjà été distribués sont dans le sous-répertoire css : poemefr0.css et poemefr1.css.

2 Toujours le poème de Paul Verlaine

Le fichier d'archive for-lc-2.tar comprend *huit* exemples de transformations XSLT dans le sous-répertoire verlaine :

fr0-2-fr1-plus.xsl	fr0-2-xhtml-plus.xsl	fr0-cumulate-count.xsl
fr0-2-html-plus.xsl	fr0-count-plus.xsl	fr1-2-fr0-plus.xsl
fr0-2-latex-plus.xsl	fr0-cumulate-count-plus.xsl	

Il s'agit pour la plupart de fonctionnalités qui ont déjà été commentées en cours, mais nous vous en avons donné pour la plupart des versions révisées et complétées au moyen d'*annotations de types*[11]. Vous pouvez essayer la transformation d'un document XML en conformité avec le tout premier fichier DTD que nous avons donné (poemefr0.dtd) vers un texte source utilisable par le

6. ***D**ocument **T**ype **D**efinition.*
7. C'est notamment ainsi qu'il faut procéder si l'on cherche à introduire de nouvelles entités dans un document décrit au moyen non pas d'une DTD mais d'un *schéma*.
8. ***C**ascading **S**tyle **S**heet.*
9. ***H**yper**T**ext **M**arkup **L**anguage.*
10. *e**X**tensible **H**yper**T**ext **M**arkup **L**anguage.*
11. C'est ce que signale le suffixe « `-plus.xsl` » dans le nom du fichier correspondant.

traitement de texte LaTeX : fr0-2-latex-plus.xsl. Si vous possédez des rudiments de LaTeX, vous pouvez les mettre à profit pour découvrir ce texte source. Quoi qu'il en soit, vous pouvez aussi, pour juger du résultat obtenu, taper au choix les commandes suivantes, après la transformation :

— soit `latex` ⟨*file*⟩, puis `xdvi` ⟨*file*⟩ pour visionner le fichier DVI [12], et `dvips` ⟨*file*⟩ pour fabriquer un fichier PostScript prêt à être imprimé ;

— soit `pdflatex` ⟨*file*⟩ pour construire un fichier PDF [13]. Utilisez la commande `evince` pour visualiser un tel fichier PDF sur le réseau enseignement, à partir duquel vous travaillez [14].

De même, le fichier fr0-count-plus.xsl vous donne le tout premier exemple vu en cours : le comptage des strophes d'un poème écrit suivant le fichier DTD poemefr0.dtd. Votre gentil enseignant vous a déjà donné en cours magistral quelques conseils au sujet de la meilleure méthode pour la spécification des fins de ligne dans le résultat en sortie et de la gestion des nœuds blancs dans le document XML en entrée. Quant au fichier fr0-cumulate-count-plus.xsl, lui aussi fondé sur une première version vue en cours, il réalise le comptage cumulé des lignes de la première strophe, puis des lignes des deux premières strophes, puis des trois premières strophes, et ainsi de suite jusqu'à la dernière strophe. Votre gentil animateur pourra aussi vous montrer comment compléter les *annotations de types* [15] — utilisant l'attribut `as` — de ce dernier fichier.

$\Longrightarrow$ Écrire des transformations XSLT fr1-count.xsl et fr1-cumulate.xsl réalisant elles aussi ces deux fonctionnalités, mais ces fichiers seront applicables à un poème tel que automne-1.xml, écrit suivant le fichier DTD poemefr1.dtd [16].

Vous pouvez également essayer les fichiers fr0-2-fr1-plus.xsl et fr1-2-fr0-plus.xsl : ils permettent de passer de la spécification arborescente, telle qu'elle est décrite dans le fichier DTD poemefr0.dtd, à l'organisation plus linéaire qui est décrite dans le fichier DTD poemefr1.dtd, et *vice versa*.

Ensuite, examiner le fichier fr0-2-html-plus.xsl, qui permet de générer une page en HTML à partir d'un document érit suivant le fichier DTD poemefr0.dtd. En fait, ce qui est construit par le processeur XSLT dans une première étape, c'est une structure intermédiaire dont les balises de HTML suivent une syntaxe « à la XML », c'est-à-dire que les règles syntaxiques concernant les éléments et les attributs sont celles de XML [17]. Puis une seconde étape, dite de *sérialisation*, restitue le fichier HTML standard [18] — c'est cette phase de sérialisation qui est la raison d'être de la valeur `html` pour l'attribut `method` de l'élément `xsl:output` de XSLT —, utilisant les conventions originelles de SGML [19] : vous pouvez remarquer que le résultat s'affiche sans problème dans un

12. ***De**Vice-**I**ndependent*. Il s'agit d'un format créé par l'auteur de TeX, permettant dès 1978 (l'année de la première version de TeX) de spécifier le placement d'objets sur une feuille de papier indépendamment de la résolution des imprimantes utilisées. Vous pouvez voir les informations manipulées par ce format en tapant la commande `dvitype` ⟨*file*⟩.

13. ***P**ortable **D**ocument **F**ormat*, le format d'Adobe.

14. Si vous travaillez sur le système Mac OS X, utilisez l'application Aperçu (Preview) pour visualiser ce fichier, ou mieux, tapez `open` ⟨*file*⟩`.pdf`.

15. Non exigées des étudiants lors des examens, empressons-nous de le préciser.

16. Attention ! ne *pas* mélanger les bobines de films différents : si vous appliquez à un document XML respectant le fichier DTD poemefr0.dtd une feuille de style écrite pour un document conforme à poemefr1.dtd (ou *vice versa*), le résultat va être très curieux. Utilisez bien la numérotation dont nous sommes convenus au départ — « 0 » ou « 1 » — pour être sûrs de vous y retrouver parmi les deux spécifications.

17. Par exemple, l'écriture de la balise indiquant un passage à la ligne suivante y est `<br/>`.

18. C'est dans ce résultat qu'on pourra remarquer que la balise indiquant un passage à la ligne suivante est `<br>`, sans utilisation de balise fermante. De même, une balise « à la XML » telle que `<a href="..."/>` est remplacée par « `<a href="..."></a>` », la notation « `<.../>` » étant inconue dans les premières versions du langage HTML.

19. ***S**tandard **G**eneralized **M**arkup **L**anguage*, l'ancêtre de XML qui, à l'heure actuelle, n'a plus guère qu'un intérêt historique.

Web browser. La génération d'un fichier XHTML, suivant strictement la recommandation la plus récente du W3C [20] pour ce langage [21] vous est donnée par la feuille de style fr0-2-xhml-plus.xsl. Là aussi, vous pouvez remarquer que le résultat s'affiche correctement dans un *Web browser*. La principale raison d'être de la valeur `xhtml` pour l'attribut `method` de l'élément `xsl:output` est l'obligation, pour un texte en XHTML, de redéfinir l'espace de noms par défaut, au moyen de l'attribut `xmlns` placé à la racine du document généré.

$\Longrightarrow$ Reprendre la transformation fr0-2-html.xsl et tenter d'optimiser les commandes de transformations du préambule : essayer de donner une présentation plus élégante au résultat obtenu pour ce document en ligne. Vous pouvez aussi remarquer que les années de naissance et de décès du poète ne sont affichées dans le résultat que si elles sont renseignées toutes les deux [22] : étudier la mise en œuvre d'une convention beaucoup moins restrictive. Après quoi, donner un fichier fr1-2-html.xsl, réalisant la même opération, mais à partir d'un document conforme au fichier DTD poemefr1.dtd. À titre d'exercice supplémentaire, vous pouvez aussi vous entraîner à l'écriture d'un programme fr1-2-xhtml.xsl, réalisant la même fonctionnalité à partir du même fichier DTD, mais avec un résultat suivant les conventions du langage XHTML.

3 Retour aux références bibliographiques

Nous considérons à nouveau les références bibliographiques vues dans la séance précédente, mais cette fois dans la version où toutes les informations concernant les personnes — que ce soit des auteurs, anthologistes ou traducteurs — sont regroupées dans des éléments situés en fin de document, et accessibles depuis les références bibliographiques proprement dites au moyen d'attributs de type `IDREF(S)`. Cette version incorpore également les éléments `skippable` correspondant aux préfixes des titres qui ne doivent pas être pris en compte pour le classement par ordre alphabétique, par exemple :

```
<title><skippable>The </skippable>Running Man</title>
```

qui indique que le titre correspondant doit être rangé alphabétiquement comme « *Running Man* ». Cette nouvelle version se constitue de deux fichiers, books-1.xml et sf-1.dtd, déjà utilisés dans la séance précédente, une version corrigée et complétée du fichier sf-1.dtd ayant été incorporée au fichier d'archive for-lc-2.tar, dans le sous-répertoire sf-1.

3.1 *Tema*

Donner une feuille de style XSLT qui permet l'affichage, suivant l'ordre d'apparition, des spécifications de livres (*books*) d'un fichier tel que `books-1.xml`. Le résultat sera un texte simple : utiliser la valeur `text` pour l'attribut `method` de l'élément `xsl:output`. Pour chacun de ces livres, on précisera les noms des auteurs, suivis du titre de l'ouvrage, du nom de l'éditeur (*publisher*) et de l'année de publication. En ce qui concerne les auteurs, le nom de famille doit apparaître en

20. ***W**orld **W**ide **W**eb **C**onsortium.*

21. . . . qui malheureusement n'a pas été mis à jour lors de la spécification de HTML 5. Cette version conserve quelques caractères archaïques des premières spécifications du langage HTML à travers la notion de *balisage polyglotte*, et la génération de pages HTML utilisant des constructions introduites par cette nouvelle version à partir de programmes XSLT est possible, mais sort du cadre de nos exercices.

22. Ce défaut — car, comme nous vous le demandons, il est possible de faire mieux — peut être aussi observé dans les transformations précédentes vers des fichiers sources LaTeX — c'est-à-dire dans les fichiers fr0-2-latex.xsl et fr0-2-latex-plus.xsl — et vers des pages XHTML — le fichier fr0-xhtml-plus.xsl.

premier, suivi du prénom entre parenthèses, s'il y a lieu. Dans le cas de plusieurs auteurs, leurs noms respectifs seront séparés par des virgules. Par exemple :

Souvestre (Pierre), Allain (Marcel). Le policier apache. Fayard, 1912. (1)

Nous rappelons que si x est une valeur de type `IDREFS`, l'expression XPath `id(`x`)` retourne la séquence constituée des sous-arbres portant comme `ID` les valeurs de x. Si y est une valeur de type `IDREF`, l'expression `id(`y`)` a le même comportement, mais la séquence retournée est bien entendu réduite à un seul membre.

Remarque Les résultats retournés par la fonction `id` de XPath appliquée à une donnée de type `IDREFS` sont toujours ordonnés dans le sens du document, ce qui peut compliquer certains affichages et tend — par exemple — que Marcel Allain soit présenté avant Pierre Souvestre dans l'affichage de la référence (1). Dans un premier temps, nous ne nous préoccuperons pas de ce problème, et par la suite, votre gentil animateur vous montrera comment procéder pour le résoudre.

Revenant au programme XSLT, on tentera en outre d'utiliser autant que possible les annotations de type au moyen de l'attribut `as`, qui peut s'appliquer aux éléments suivants de XSLT :

`xsl:template` `xsl:param` `xsl:with-param` `xsl:variable` `xsl:function`

3.2 *Variazioni*

Ensuite, même exercice, mais en présentant les références bibliographiques suivant les ordres décrits ci-après :

— l'ordre inverse de celui du document XML original,

— l'ordre croissant des années de parution : remarquez qu'il s'agit en fait d'un *préordre* ;

— l'ordre croissant des années de parution, puis l'ordre alphabétique des deux premiers auteurs — ou anthologistes — pour les ouvrages parus la même année : suivant les noms de famille d'abord, suivant les prénoms ensuite [23] ;

— l'ordre lexicographique donné par les titres, sans considérer les contenus des éléments `skippable` [24].

Nous rappelons que si plusieurs éléments `xsl:sort` se succèdent, un ordre `xsl:sort` trie les éléments qui n'ont pas été départagés par les tris précédents. L'attribut `select` de l'élément `xsl:sort` est une expression XPath, pour les valeurs associées aux autres attributs mentionnés ci-après pour cet élément, elles sont prises dans des types énumérés. Cet élément `xsl:sort` s'utilise comme suit — les valeurs par défaut des attributs sont soulignées — :

```
<xsl:sort select="." lang="en | fr | de | ..." data-type="text | number"
          order="ascending | descending" case-order="upper-first | lower-first"
          stable="yes | no"/>
```

les attributs s'employant comme suit :

23. En étant perfectionnistes, nous pourrions considérer *tous* les auteurs, mais à vrai dire, cela commence à être de la haute voltige... du très grand jeu... Á titre de complément, on vous montrera dans le corrigé comment c'est possible.

24. Pour ce faire, utiliser la construction `except` de XPath.

— `select` donne la clé du tri, la valeur par défaut est le chemin XPath « . » ;

— `data-type` indique s'il s'agit d'un tri numérique ou alphabétique ;

— `order` indique le sens — croissant ou décroissant — du tri ;

— les deux attributs suivants n'ont de sens que lorsque l'attribut `data-type` vaut `"text"` :

  ⋆ `lang` est un code désignant une langue naturelle, utilisé pour déterminer l'ordre alphabétique correspondant à cette langue [25] ;

  ⋆ `case-order` indique comment ordonner les lettres capitales par rapport au bas de casse (les lettres minuscules) ; la valeur par défaut dépend de la langue considérée [26] ;

— la liaison de la valeur `yes` à l'attribut `stable` — la valeur par défaut de cet attribut étant `no` — ne peut s'employer que pour le premier membre d'une suite d'éléments `xsl:sort` consécutifs.

Enfin, rappelons également que dans le cas d'un tri numérique, il n'est pas nécessaire de lier l'attribut `data-type` à la valeur `number` :

— si le résultat retourné par l'attribut `select` porte une annotation de type numérique :

```
<xsl:sort select="L3:little-player-nb(.)"/>
```

avec :

```
<xsl:function name="L3:little-player-nb" as="xsd:integer">
  <xsl:param name="..." ...>
  ...
</xsl:function>
```

où « `xsd:` » est le préfixe utilisé pour la bibliothèque de types simples de XML Schema, défini par `xmlns:xsd="http://www.w3.org/2001/XMLSchema"`

— ou si une conversion vers un type numérique a été utilisée :

```
<xsl:sort select="xsd:integer(nb-of-little-players)"/>
```

3.3 Traquer les doublons

Écrire maintenant deux feuilles de style — distincts-v0.xsl et distincts-v1.xsl — parcourant tous les titres (éléments `title`) et précisant quels sont les titres qui sont *univoques*, c'est-à-dire tels qu'aucune autre œuvre de la bibliographie ne porte ce titre. La propriété pour un titre de n'être répété nulle part dans les autres titres peut être réalisée :

— à l'aide d'expressions XPath adéquates : ce n'est néanmoins qu'une solution de petit joueur, car très inefficace, surtout en présence d'une très grosse bibliographie à tester ;

25. Comme nous vous l'avons expliqué en travaux dirigés, ce point n'est que partiellement implanté actuellement. Quoi qu'il en soit, prenez `lang="fr"`.

26. En pratique, c'est très souvent `upper-first`.

— à l'aide de *clés* de XSLT, calculées au préalable par l'élément `xsl:key`, dont nous rappelons les attributs :

```
<xsl:key name="..." match="..." use="..."/>
```

il suffit dès lors de vérifier, en parcourant tous les ouvrages, que la clé de XSLT relative au titre de l'entrée bibliographique ne contient pas d'autres éléments que l'élément présent ; en XSLT, l'image réciproque d'une valeur `v` de la clé `k` est donnée par l'expression `key(k,v)`, qui retourne une séquence de nœuds, ordonnée dans le sens du document.