Khalil el maghraoui

TP2 XML

XQuery

EXERCICE 1 :

<bib>

<book year="1994">

<title>TCP/IP Illustrated</title>

<author><last>Stevens</last><first>W.</first></author>

<publisher>Addison-Wesley</publisher>

<price>65.95</price>

</book>

<book year="1992">

<title>Advanced Programming in the Unix environment</title>

<author><last>Stevens</last><first>W.</first></author>

<publisher>Addison-Wesley</publisher>

<price>65.95</price>

</book>

<book year="2000">

<title>Data on the Web</title>

<author><last>Abiteboul</last><first>Serge</first></author>

<author><last>Buneman</last><first>Peter</first></author>

<author><last>Suciu</last><first>Dan</first></author>

<publisher>Morgan Kaufmann Publishers</publisher>

<price>39.95</price>

</book>

<book year="1999">

<title>The Economics of Technology and Content for Digital TV</title>

<editor>

<last>Gerbarg</last><first>Darcy</first>

<affiliation>CITI</affiliation>

</editor>

<publisher>Kluwer Academic Publishers</publisher>

<price>129.95</price>

</book>

</bib>

Requête 1 :

Sélectionne les livres publiés par Addison-Wisley après 1991. Pour

chacun d'entre eux, retourne un élément book contenant le titre du

livre et possédant un attribut year contenant l'année de publication.

Notez l'utilisation du where pour ajouter une contrainte sur les

éléments sélectionnés par la boucle for.

L'ensemble de ces éléments book est placé dans un élément bib.

Requête 2 :

===========

Construit à partir des données bibliographiques une liste d'éléments

result contenant les associations titre de livre - auteur (jointure).

Ces éléments sont placés au sein d'une balise results.

Attention, l'enchaînement de variables avec des virgules dans un for

correspond en fait à plusieurs boucles for imbriquées.

Requête 3 :

===========

Construit pour chaque auteur la liste des livres dont il est un des

auteurs. Pour chaque auteur, ces informations sont placées dans une

balise result. L'ensemble des ces balises est placé dans un élément

results. Une double boucle est nécessaire pour 1/ parcourir la liste

des auteurs et 2/ pour chaque auteur, parcourir la liste des livres

pour recenser les livres dont il est auteur.

EXERCICE 2 :

<collection>

<serie nom="Lanfeust de Troy">

<tome numero="1">

<scenariste>Arleston</scenariste>

<dessinateur>Tarquin</dessinateur>

<titre>L&apos;ivoire du Magohamoth</titre>

</tome>

<tome numero="2">

<dessinateur>Tarquin</dessinateur>

<titre>Thanos l&apos;incongru</titre>

</tome>

<editeur nom="Soleil production"/>

</serie>

<serie nom="Calvin &amp; Hobbes">

<tome numero="1">

<titre>Adieu, monde cruel</titre>

</tome>

</serie>

<serie nom="Léonard">

<tome numero="23">

<scenariste>De groot</scenariste>

<dessinateur>Turk</dessinateur>

<titre>Poil au génie</titre>

</tome>

<editeur nom="Le lombard"/>

</serie>

</collection>

1.

for $serie in doc("collection.xml")//serie

return

<serie

titre="{$serie/@nom}"

editeur="{$serie/editeur/@nom}"

/>

2.

for $tome in doc("collection.xml")//tome

let $serie := doc("collection.xml")//serie[tome=$tome]

return

<tome numero="{$tome/@numero}" serie="{$serie/@nom}" titre="{$tome/titre/text()}"/>

3.

for $tome in doc("collection.xml")//tome

let $serie := doc("collection.xml")//serie[tome=$tome]

let $titre := $tome/titre/text()

order by $titre

return

<tome numero="{$tome/@numero}" serie="{$serie/@nom}" titre="{$titre}"/>

Exercice 3 :

import java.io.File;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.transform.OutputKeys;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerConfigurationException;

import javax.xml.transform.TransformerException;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.Comment;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

public class ReadXMLFile

{

public static void main(final String[] args) {

/\*

\* Etape 1 : récupération d'une instance de la classe "DocumentBuilderFactory"

\*/

final DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

try

{

/\*

\* Etape 2 : création d'un parseur

\*/

final DocumentBuilder builder = factory.newDocumentBuilder();

/\*

\* Etape 3 : création d'un Document

\*/

final Document document= builder.newDocument();

/\*

\* Etape 4 : création de l'Element racine

\*/

final Element racine = document.createElement("repertoire");

document.appendChild(racine);

/\*

\* Etape 5 : création d'une personne

\*/

final Comment commentaire = document.createComment("John DOE");

racine.appendChild(commentaire);

final Element personne = document.createElement("personne");

personne.setAttribute("sexe", "masculin");

racine.appendChild(personne);

/\*

\* Etape 6 : création du nom et du prénom

\*/

final Element nom = document.createElement("nom");

nom.appendChild(document.createTextNode("DOE"));

final Element prenom = document.createElement("prenom");

prenom.appendChild(document.createTextNode("John"));

personne.appendChild(nom);

personne.appendChild(prenom);

/\*

\* Etape 7 : création des numéros de téléphone

\*/

final Element telephones = document.createElement("telephones");

final Element fixe = document.createElement("telephone");

fixe.appendChild(document.createTextNode("01 02 03 04 05"));

fixe.setAttribute("type", "fixe");

final Element portable = document.createElement("telephone");

portable.appendChild(document.createTextNode("06 07 08 09 10"));

portable.setAttribute("type", "portable");

telephones.appendChild(fixe);

telephones.appendChild(portable);

personne.appendChild(telephones);

/\*

\* Etape 8 : affichage

\*/

final TransformerFactory transformerFactory = TransformerFactory.newInstance();

final Transformer transformer = transformerFactory.newTransformer();

final DOMSource source = new DOMSource(document);

final StreamResult sortie = new StreamResult(new File("F:\\file.xml"));

//final StreamResult result = new StreamResult(System.out);

//prologue

transformer.setOutputProperty(OutputKeys.VERSION, "1.0");

transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");

transformer.setOutputProperty(OutputKeys.STANDALONE, "yes");

//formatage

transformer.setOutputProperty(OutputKeys.INDENT, "yes");

transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "2");

//sortie

transformer.transform(source, sortie);

}

catch (final ParserConfigurationException e)

{

e.printStackTrace();

}

catch (TransformerConfigurationException e)

{

e.printStackTrace();

}

catch (TransformerException e)

{

e.printStackTrace(); }} }

declare function local:sortie\_cat($cat as element(categorie)) as element(div) {

{

**{ (: element h3 avec le nom de la categorie :) $cat/@nom/data() }**

, (: paragraphe avec la description si elle existe :) if ($cat/description) then

{ $cat/description/text() }

else (), (: puis on fait la liste des bookmarks - avec hiérarchie des sous-categories :)

{ for $elt in $cat/element() (: $elt represente un element fils de l'element $cat :) return if ($elt instance of element(categorie)) then (: $elt est une sous categorie, donc appel récursif :)

* {local:sortie\_cat($elt)}

else (if ($elt instance of element(bookmark)) then (: $elt est un bookmark, on appelle la fonction de la question 1 :)

* {local:sortie\_bookmark($elt)}

else ()) (: on ne traite pas les autres types d'elements, s'il y en a :) }

}

}; (: pour tester la question : for $b in //categorie return local: sortie\_cat($b) :)