# ΔΙΑΔΙΚΤΥΟ ΚΑΙ ΕΦΑΡΜΟΓΕΣ Εαρινό εξάμηνο 2013

**Άσκηση 2η :  XML και XSL Transformations**

**Eυρίσκεται και στο site του μαθήματος** [www.cn.ntua.gr](http://www.cn.ntua.gr)

**Προθεσμία παράδοσης σε γραπτή μορφή Παρασκευή 17.05.13**

Να παραδοθούν σε έντυπη μορφή τα σχετικά προγράμματα και τα αποτελέσματά τους. Ίδιες απαντήσεις από διαφορετικούς σπουδαστές δεν θα λαμβάνονται υπ’ όψιν.

###### Ι. XSL

Επιλέξτε μία περιοχή ενδιαφέροντός σας (π.χ. αυτοκίνητα, τρόφιμα, διακοπές, κλπ) και γράψτε ένα σχετικό κείμενο .xml που να περιέχει το root, αντικείμενα με attributes και ένα επιπλέον παρακάτω επίπεδο χωρίς attributes. ‘Όλα τα αντικείμενα πρέπει να έχουν κοινούς τύπους χαρακτηριστικών. Μερικά χαρακτηριστικά εμφανίζονται σαν παιδιά του αντικειμένου και μερικά σαν attributes. Ένα τέτοιο παράδειγμα είναι το Παράδειγμα 7.1 των σημειώσεων.

Μετά δημιουργήστε με δύο τρόπους σε μια σελίδα html ένα πίνακα, ο οποίος θα έχει μία σειρά για κάθε αντικείμενο και μία στήλη για την τιμή του κάθε χαρακτηριστικού. Στην πρώτη σειρά θα φαίνεται ο τύπος (ονομασία, π.χ. color) του χαρακτηριστικού, δηλ ο χαρακτηρισμός της στήλης. Στον χαρακτηρισμό θα επιπροστίθεται το string “ (a)” ή “ (c)”, όταν αυτός προέρχεται από attribute ή από παιδί του αντικειμένου Οι δύο τρόποι είναι:

* **Α.(ΥΠΟΧΡΕΩΤΙΚΟ-χρειάζεται για το ΙΙ κάτω)** Με μετασχηματισμό .xslt
* **Β.(ΠΡΟΑΙΡΕΤΙΚΟ)** Με πρόγραμμα κατά το πρότυπο του Κωδ. 6.5 των σημειώσεων.

Φυσικά, και οι δύο τρόποι θα πρέπει να είναι τελείως γενικοί και να παρουσιάζουν χωρίς καμία αλλαγή κάθε κείμενο .xml με τις παραπάνω προδιαγραφές.

Υπόδειξη για το Α. : Η χρήση του mode attribute μπορεί να βοηθήσει στον διαχωρισμό της εντόπισης ενός στοιχείου είτε για να γράψομε τον όνομά του (μία φορα) σαν τίτλο της κάθε στήλης, είτε για να γράψομε την τιμή (πολλές φορές για κάθε σοτιχείο.

**ΙΙ. Μετασχηματισμός XSL μέσα στον Tomcat**

Δίδεται o πηγαίος κώδικας του servlet **XMLTransformerAskhsh**.java, το οποίο ‘μορφώνεται’ με ένα μετασχηματισμό κατά την αρχικοποίησή του (init). Όταν κληθή, σύμφωνα με τις επιθυμίες του χρήστη ως προς την παρουσίαση (βλ. παρακάτω το doGet), μετασχηματίζει μέσω του CarPresentor2.xsl το Cars.xml (ευρίσκονται και τα δύο στο WEB-INF) εμφανίζοντας το τελευταίο στον browser σε μορφή πίνακα. Όπως δίνεται στο πρόγραμμα, το xsltDoc φτιάχνεται (από αρχείο) σαν SAXSource, δηλ. σαν μία ακολουθία συμβάντων, η οποία ‘μορφώνει’ το myTransformer της Transformer, για το πώς θα μετασχηματίζει.

import java.io.\*;

import javax.xml.transform.stream.\*;

import javax.xml.transform.\*;

import javax.xml.transform.sax.SAXSource;

import javax.xml.transform.stream.StreamSource;

import javax.xml.transform.stream.StreamResult;

import org.xml.sax.\*;

// for servlet:

import javax.servlet.\*;

import javax.servlet.http.\*;

import org.w3c.dom.\*; // for DOM (Java DOM)

import javax.xml.parsers.\*;

import javax.xml.transform.dom.\*;// for transformations

public class **XMLTransformerAskhsh** extends HttpServlet {

ServletContext ctx;

String absPath; //absolute path to our files - see below

SAXSource xsltDoc; TransformerFactory tF;

Transformer myTransformer;// will be built at init, will be used by doGet

Document doc;

public void init(ServletConfig config) throws UnavailableException {

System.out.println("Init start");

try {

ctx = config.getServletContext(); // we will use the 'contex' below

absPath = ctx.getRealPath("/WEB-INF/CarPresentor2.xsl");

xsltDoc = new SAXSource(new InputSource(absPath));

tF = TransformerFactory.newInstance();

DocumentBuilderFactory fact = DocumentBuilderFactory.newInstance();

// absolutely important, to understand the meaning of prefix 'xslt' !!!!

fact.setNamespaceAware(true);

DocumentBuilder builder = fact.newDocumentBuilder();

doc = builder.parse(absPath);

System.out.println("Name of document element is " + doc.getDocumentElement().getNodeName());

} catch (Exception e) { e.printStackTrace(); }

System.out.println("Init end");

}

private void changeDomByColor(Document doc, String color) {

NodeList nl = doc.getElementsByTagName("h1");

Attr a = doc.createAttribute("style");

a.setValue("background-color: "+color);

nl.item(0).getAttributes().setNamedItem(a);

}

public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

System.out.println("dopost start");

System.out.println("Name of document element (at the post) is " + doc.getDocumentElement().getNodeName());

String color = request.getParameter("color");

System.out.println("You selected the color " + color);

System.out.println(doc.getElementsByTagName("h1").item(0).getAttributes().getNamedItem("style").getNodeValue());

changeDomByColor(doc, color);

System.out.println(doc.getElementsByTagName("h1").item(0).getAttributes().getNamedItem("style").getNodeValue());

PrintWriter pwr = response.getWriter();

try {

DOMSource ds = new DOMSource(doc) ;

System.out.println( ((Document)ds.getNode()).getDocumentElement().getNodeName() +" " +((Document)ds.getNode()).getDocumentElement().getNodeValue() ) ;

// myTransformer = tF.newTransformer(new DOMSource(doc));

// myTransformer = tF.newTransformer(xsltDoc);

myTransformer = tF.newTransformer(ds);

StreamSource xmlSource = new StreamSource(ctx.getResourceAsStream("/WEB-INF/Cars.xml"));

System.out.println("Sending back the xml tranformed into html");

response.setContentType("text/html"); //in order to put in http body

myTransformer.transform(xmlSource, new StreamResult(pwr));

pwr.println("The response sent back as a page!");

pwr.close();

}

catch (Exception e)

{

e.printStackTrace();

}

System.out.println("dopost stop");

}

public void doGet( HttpServletRequest request, HttpServletResponse response ) throws ServletException, IOException {

System.out.println("doget start");

PrintWriter output;

response.setContentType( "text/html" );

output = response.getWriter();

output.println( "<HTML><HEAD><TITLE>" );

output.println( "test" );

output.println( "</TITLE></HEAD><BODY>" );

output.println( "<FORM ACTION='http://localhost:8080/TransformerDir/TransformerAsk' METHOD='POST'>");

output.println( "<STRONG>Please select:<br> </STRONG><PRE>");

output.println( "<INPUT TYPE='radio' NAME='color' VALUE='red'>RED<BR>");

output.println( "<INPUT TYPE='radio' NAME='color' VALUE='green'>GREEN<BR>");

output.println( "<INPUT TYPE='radio' NAME='color' VALUE='blue' CHECKED>BLUE<BR>");

output.println( "</PRE><INPUT TYPE='submit' VALUE='OK'>");

output.println( "</FORM>");

output.println( "</BODY></HTML>" );

output.close(); // close stream

System.out.println("doget stop");

}

}

Στον παραπάνω κώδικα του servlet γίνεται αναφορά στα αρχεία Cars.xml και CarPresentor2.xsl τα οποία πρέπει εσεις να καταλάβετε που πρέπει να τοποθετηθούν. Πρέπει επίσης εσείς να βρείτε πως μπορούμε να δοκιμάσομε κατ’ αρχήν τον παραπάνω κώδικα.

Τα αρχεία Cars.xml και CarPresentor.xsl υπάρχουν στις σημειώσεις.

Το αρχείο CarPresentor2.xsl ειναι το ίδιο με το CarPresentor.xsl με τις εξείς μετατροπές:

(α) Αλλάξτε την γραμμή xsl:stylesheet σε

<xsl:stylesheet version = "1.0" xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">

(β) σβήστε τελείως το attribute order-by

Μελετήστε, εγκαταστήστε στον Tomcat και δοκιμάστε το παραπάνω πρόγραμμα. Το πρόγραμμα είναι σε Java DOM. Αν θέλετε παραπάνω λεπτομέρειες για μετασχηματισμούς – Transforms δείτε τις δύο παρουσιάσεις στο DOM\_Transforms από το site.

Με βάση το παραπάνω παράδειγμα φτιάξτε μια εφαρμογή που στον server περιέχεται το XSLT του Ι.Α. και 2-3 XML (αυτοκίνητα, πλοία, αεροπλάνα , η παρόμοια). Ο χρήστης στον browser επιλέγει ποίο από αυτά θέλει να δει σε μορφή πίνακα καθώς και με ποίο εμφανισιακό στιλ (π.χ. κάποιο χρώμα για τους τίτλους των στηλών). Για απλούστευση μπορείτε να καταργήσετε την doGet του παραπάνω παραδείγματος και να εντάξετε την αρχική σελίδα επιλογής των αντικειμένων και της παρουσίασής των σαν index.html.

**Λεπτά Σημεία (μέσα στην XMLTransformerAskhsh)**

**1ον. Πως φτιάχνομε DOMSource;**

ΟΧΙ απευθείας από αρχείο, όπως γίνεται στην περίπτωση της SAXSource,

* αλλά αφού πρώτα διαβάσομε το αρχείο και δημιουργήσομε Document doc (τέτοιο παράδειγμα έχει δοθεί, βλ και παρακάτω)
* και μετά με DOMSource ds = new DOMSource(doc)

Στα παραπάνω προσοχή να βάλετε τα αναγκαία import (φαίνονται στα παραδείγματα που δώσαμε).

**2ον. namespace aware**

Όταν ο builder (βλ. κάτω) επεξεργάζεται ένα αρχείο κειμένου xml και προσπαθεί να διαχωρίζει τα ονόματα των στοιχείων xml, δεν διακρίνει ότι πριν την ‘:’ δεν υπονοείται μέρος του ονόματος αλλά το πρόθεμα ενός namespace (στην συγκεκριμένη περίπτωση το xsl:). Ο builder δεν είναι ‘namespace aware’, δηλ. πνίγεται το namespace, στο οποίο δηλώνεται ότι πρόκειται για xsl πέραν του ότι είναι xml !

**Θεραπεία**:

DocumentBuilderFactory fact = DocumentBuilderFactory.newInstance();

**fact.setNamespaceAware(true); // PREPEI NA SHKVTHEI AYTH H FLAG !!!!!!**

DocumentBuilder builder = fact.newDocumentBuilder();

doc = builder.parse(absPath);