|  |
| --- |
| ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ  ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ  ΑΚΑΔΗΜΑΙΚΟ ΕΤΟΣ 2012 – 2013 Εαρινό εξαμηνο |
| Διαδίκτυο και Εφαρμογές |
|  |
| Άσκηση 2η : XML και XSL Transformations |
|  |
|  |

8Ο ΕΞΑΜΗΝΟ

**ΦΟΙΤΗΤΗΣ:**

ΚΟΓΙΑΣ ΜΑΡΙΟΣ-ΕΥΑΓΓΕΛΟΣ

**Σκοπός**

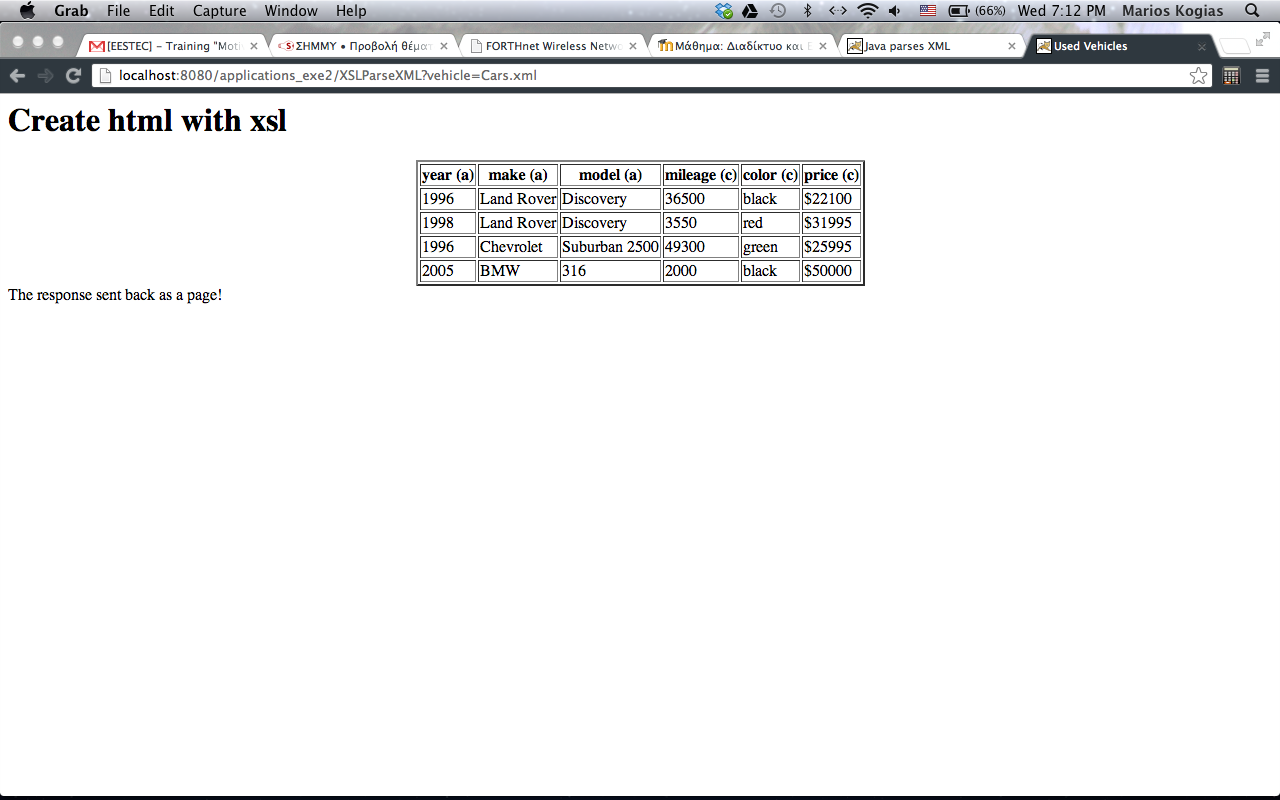
Σκοπός της συγκεκριμένης άσκησης είναι η εξοικείωση με την τεχνολογία του xml και xsl. Ζητούμενο είναι δεδομένου ενός συγκεκριμένου προτύπου xml να δημιουργούμε έναν πίνακα html με τα περιεχόμενα του html. Αυτό το κάναμε με 2 διαφορετικούς τρόπους:

* με xsl, η οποία ορίζει ένα πρότυπο παραγωγής html από το δεδομένο xml
* με java η οποία από το xml δημιουργεί την dom αναπαράστασή του και τυπώνει τα περιεχόμενα των κόμβων μέσα στο html

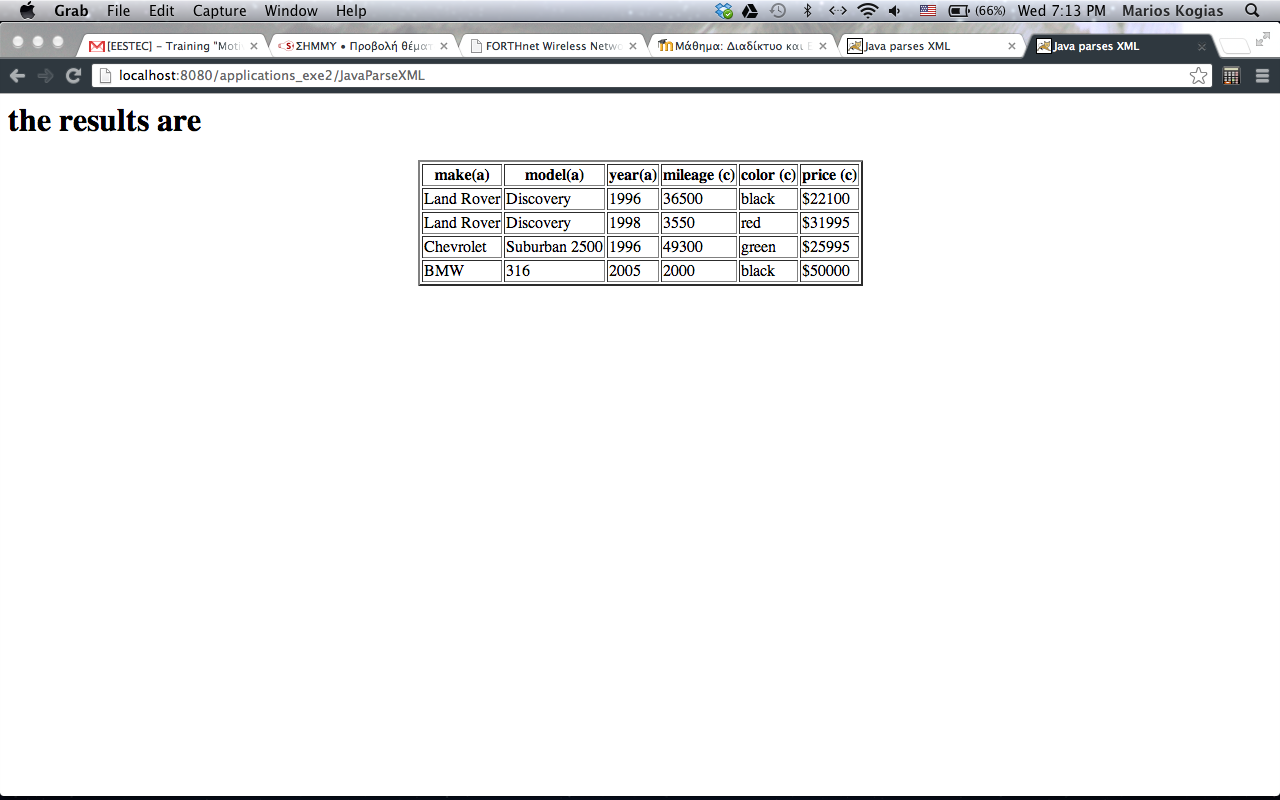
###### Ι. XSL

Στο πρώτο μέρος δημιουργούμε δυο διαφορετικές σελίδες οι οποίες δεδομένου κάποιου xml δημιουργούν έναν πίνακα html. Η πρώτη δημιουργεί το html μέσω xsl ενώ στη δεύτερη το html δημιουργείται από τη java.

Παρακάτω μπορούμε να δούμε 2 screenshots των 2 αυτών σελίδων.



*Δημιουργία html από xml με βοήθεια xsl*

**

*Δημιουργία html μέσω java xml parsing*

Ο κώδικας των 2 servlets μπορεί να βρεθεί στο τέλος της αναφοράς.

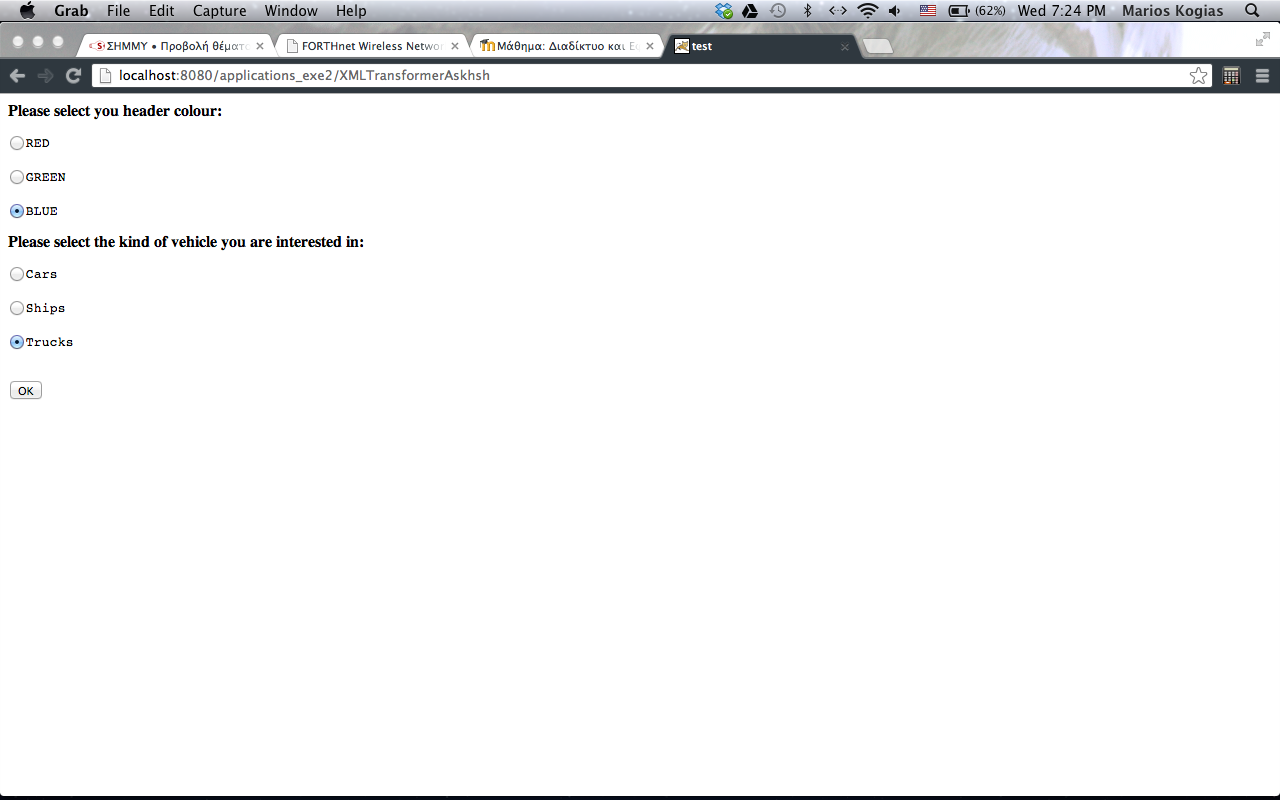
-XSLParseXML.java

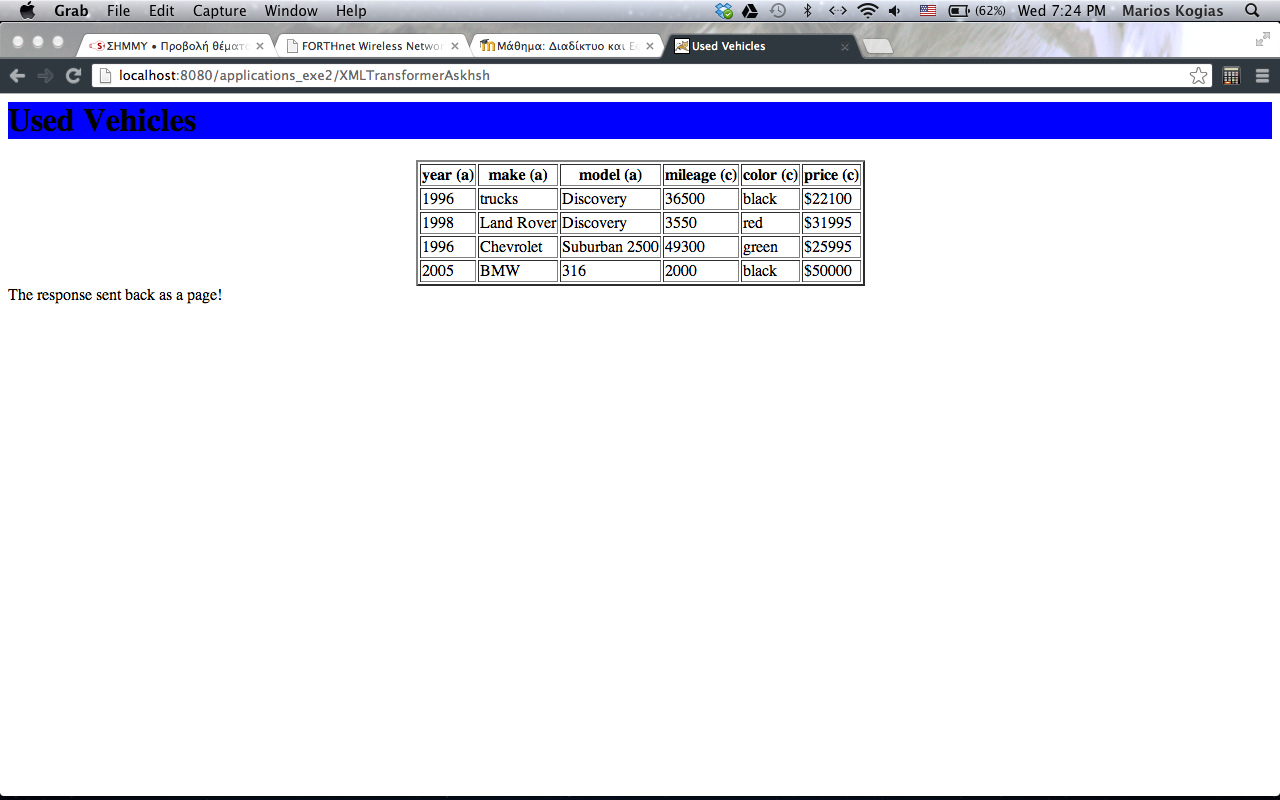
-JavaParseXML.java

**ΙΙ. Μετασχηματισμός XSL μέσα στον Tomcat**

Στο δεύτερο μέρος χρησιμοποιούμε ό,τι φτιάξαμε στο πρώτο μέρος για την xsl και δημιουργούμε μια δυναμική σελίδα. Η σελίδα αφορά μια αντιπροσωπεία μεταχειρισμένων οχημάτων. Στην εφαρμογή ο χρήστης αρχικά δίνει κάποιες παραμέτρους μορφοποίησης (χρώμα επικεφαλίδας) καθώς και το είδος του οχήματος που τον ενδιαφέρει και τελικά παίρνει σε έναν πίνακα τα διαθέσιμα οχήματα της εταιρίας για τη συγκεκριμένη κατηγορία.

Διαδοχικά screenshots της εφαρμογής είναι:





Σε όλα τα παραπάνω παραδείγματα δίπλα από κάθε τίτλο στήλης υπάρχει ένα (a) αν τα στοιχεία αυτά προέρχονται από attributes ενός κόμβου ή ένα (c) αν τα στοιχεία προέρχονται από περιεχόμενο ξεχωριστών κόμβων.

Παρακάτω μπορείτε να βρείτε συνολικά τον κώδικα που γράφτηκε για τη συγκεκριμένη άσκηση. Επίσης τον κώδικα αυτό μπορείτε να τον βρείτε και στο <https://github.com/marioskogias/webApplications/tree/master/applications_exe2>

/\*

\* JavaParseXML

\*/

package servlets;

import java.io.IOException;

import javax.servlet.ServletConfig;

import javax.servlet.ServletContext;

import javax.servlet.ServletException;

import javax.servlet.UnavailableException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.sax.SAXSource;

import org.w3c.dom.Document;

import org.w3c.dom.NamedNodeMap;

import org.w3c.dom.NodeList;

import org.w3c.dom.Node;

import org.w3c.dom.Element;

import java.io.File;

import org.w3c.dom.\*;

import org.xml.sax.InputSource;

import java.io.PrintWriter;

/\*\*

\* Servlet implementation class JavaParseXML

\*/

public class JavaParseXML extends HttpServlet {

private static final long serialVersionUID = 1L;

public JavaParseXML() {

super();

// TODO Auto-generated constructor stub

}

String path;

public void init(ServletConfig config) throws UnavailableException {

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

try {

path = config.getServletContext().getRealPath("/WEB-INF/Cars.xml");

} catch (Exception e) {

e.printStackTrace();

}

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

}

protected void doGet(HttpServletRequest request,

HttpServletResponse response) throws ServletException, IOException {

PrintWriter output;

response.setContentType("text/html");

output = response.getWriter();

StringBuffer buf = new StringBuffer();

buf.append("<HTML><HEAD><TITLE>\n"); // write here line-by-line the html

// for the desired page

buf.append("Java parses XML\n");

buf.append("</TITLE></HEAD><BODY>\n");

buf.append("<h1> the results are </h1>");

output.println(buf.toString());

output.println(parseXML().toString());

output.print("</body></html>");

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse

\* response)

\*/

protected void doPost(HttpServletRequest request,

HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

}

public StringBuffer parseXML() {

try {

StringBuffer buf = new StringBuffer();

DocumentBuilderFactory dbFactory = DocumentBuilderFactory

.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(path);

// optional, but recommended

// read this -

// http://stackoverflow.com/questions/13786607/normalization-in-dom-parsing-with-java-how-does-it-work

doc.getDocumentElement().normalize();

System.out.println("Root element :"

+ doc.getDocumentElement().getNodeName());

NodeList nList = doc.getDocumentElement().getChildNodes();

System.out.println("----------------------------");

Node temp = nList.item(1);

System.out.println(temp.getNodeName());

NamedNodeMap map = temp.getAttributes();

buf.append("<table align='center' border='2'><tr>");

for (int j = 0; j < map.getLength(); j++) { // attributes as titles

buf.append("<th>" + map.item(j).getNodeName() + "(a)</th>");

System.out.println(map.item(j).getNodeName());

}

NodeList firstList = temp.getChildNodes();

System.out.println("the titles are");

for (int j = 0; j < firstList.getLength(); j++) { // tags as titles

if (firstList.item(j).getNodeType() == Node.ELEMENT\_NODE) {

buf.append("<th>" + firstList.item(j).getNodeName()

+ " (c) </th>");

System.out.println(firstList.item(j).getNodeName());

}

}

buf.append("</tr>");

for (int i = 0; i < nList.getLength(); i++) {

// System.out.println(i+" " + nList.item(i).getTextContent());

Node nNode = nList.item(i);

if (nNode.getNodeType() == Node.ELEMENT\_NODE) {

NodeList children = nNode.getChildNodes();

buf.append("<tr>");

NamedNodeMap mapList = nNode.getAttributes();

System.out.println("the attribute fields are");

for (int j = 0; j < mapList.getLength(); j++) { // attributes

// as fields

buf.append("<td>" + mapList.item(j).getTextContent()

+ "</td>");

System.out.println(mapList.item(j).getTextContent());

}

System.out.println("the tags fields are");

for (int j = 0; j < children.getLength(); j++) {

if (children.item(j).getNodeType() == Node.ELEMENT\_NODE) { // tags

// as

// fileds

System.out.println(children.item(j)

.getTextContent());

buf.append("<td>"

+ children.item(j).getTextContent()

+ "</td>");

}

}

buf.append("</tr>");

}

}

buf.append("</table>");

return buf;

} catch (Exception e) {

e.printStackTrace();

}

return null;

}

}

/\*

\* XMLTransformerAskhsh

\*/

package servlets;

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/"+request.getParameter("vehicle")));

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='XMLTransformerAskhsh' METHOD='POST'>");

output.println( "<STRONG>Please select you header colour:<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></PRE>");

output.println( "<STRONG>Please select the kind of vehicle you are interested in:<br> </STRONG><PRE>");

output.println( "<INPUT TYPE='radio' NAME='vehicle' VALUE='Cars.xml'>Cars<BR>");

output.println( "<INPUT TYPE='radio' NAME='vehicle' VALUE='ships.xml'>Ships<BR>");

output.println( "<INPUT TYPE='radio' NAME='vehicle' VALUE='trucks.xml' CHECKED>Trucks<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");

}

}

/\*

\* XSLParseXML

\*/

package servlets;

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 XSLParseXML 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/firstpart.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");

}

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

}

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

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

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(ds);

StreamSource xmlSource = new StreamSource(ctx.getResourceAsStream("/WEB-INF/"+request.getParameter("vehicle")));

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");

}

}

Τα 2 αρχεία xsl ειναι

<?xml version="1.0"?>

<xsl:stylesheet version="1.0"

xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">

<html>

<head>

<title>Used Vehicles</title>

</head>

<body background="Money.jpg">

<h1

style="background-color: #446600;

color: #FFFFFF; font-size: 20pt; text-align: center;

letter-spacing: 1.0em">Used Vehicles</h1>

<table align="center" border="2">

<tr>

<xsl:template match="/">

<xsl:apply-templates select="child::\*/child::\*[1]/attribute::\*"/>

<xsl:apply-templates select="child::\*/child::\*[1]/\*"/>

</xsl:template>

</tr>

<!--This template has had so far only presentation elements, -->

<!-- without any reference to the target xml document -->

<!-- We now refer to the the target xml document -->

<xsl:for-each select="child::\*/child::\*">

<tr>

<xsl:for-each select="attribute::\*">

<td>

<xsl:value-of select="current()" />

</td>

</xsl:for-each>

<xsl:for-each select="child::\*">

<td>

<xsl:value-of select="current()" />

</td>

</xsl:for-each>

</tr>

</xsl:for-each>

</table>

</body>

</html>

</xsl:template>

<xsl:template match="child::\*/child::\*[1]/attribute::\*">

<th>

<xsl:value-of select="local-name()"/>

(a)

</th>

</xsl:template>

<xsl:template match="child::\*/child::\*[1]/\*">

<th>

<xsl:value-of select="local-name()"/>

(c)

</th>

</xsl:template>

</xsl:stylesheet>

<?xml version="1.0"?>

**<xsl:stylesheet version**=*"1.0"*

**xmlns:xsl**=*"http://www.w3.org/1999/XSL/Transform"***>**

**<xsl:template match**=*"/"***>**

<html>

<head>

<title>Used Vehicles</title>

</head>

<body >

<h1>Create html with xsl</h1>

<table align="center" border="2">

<tr>

**<xsl:template match**=*"/"***>**

**<xsl:apply-templates select**=*"child::\*/child::\*[1]/attribute::\*"***/>**

**<xsl:apply-templates select**=*"child::\*/child::\*[1]/\*"***/>**

**</xsl:template>**

</tr>

<!--This template has had so far only presentation elements, -->

<!-- without any reference to the target xml document -->

<!-- We now refer to the the target xml document -->

**<xsl:for-each select**=*"child::\*/child::\*"***>**

<tr>

**<xsl:for-each select**=*"attribute::\*"***>**

<td>

**<xsl:value-of select**=*"current()"* **/>**

</td>

**</xsl:for-each>**

**<xsl:for-each select**=*"child::\*"***>**

<td>

**<xsl:value-of select**=*"current()"* **/>**

</td>

**</xsl:for-each>**

</tr>

**</xsl:for-each>**

</table>

</body>

</html>

**</xsl:template>**

**<xsl:template match**=*"child::\*/child::\*[1]/attribute::\*"***>**

<th>

**<xsl:value-of select**=*"local-name()"***/>**

(a)

</th>

**</xsl:template>**

**<xsl:template match**=*"child::\*/child::\*[1]/\*"***>**

<th>

**<xsl:value-of select**=*"local-name()"***/>**

(c)

</th>

**</xsl:template>**

**</xsl:stylesheet>**

Τέλος ένα ενδεικτικό αρχείο xml με δεδομένα είναι το εξής

<?xml version="1.0"?>

<?xml-stylesheet href="CarPresentor2.xsl" type="text/xsl"?>

<vehicles>

<vehicle year="1996" make="Land Rover" model="Discovery">

<mileage>36500</mileage>

<color>black</color>

<price>$22100</price>

</vehicle>

<vehicle year="1998" make="Land Rover" model="Discovery">

<mileage>3550</mileage>

<color>red</color>

<price>$31995</price>

</vehicle>

<vehicle year="1996" make="Chevrolet" model="Suburban 2500">

<mileage>49300</mileage>

<color>green</color>

<price>$25995</price>

</vehicle>

<vehicle year="2005" make="BMW" model="316">

<mileage>2000</mileage>

<color>black</color>

<price>$50000</price>

</vehicle>

</vehicles>