Γενική Ιδέα 'Επίδειξης' DOM API

Απαιτούμενα packages

μέσα στην java, έχει τις κλάσεις / interfaces που παρουσιάζουν τα αντικείμενα του εγγράφου .xml στην μνήμη

```
import org.w3c.dom.*;
import javax.xml.parsers.*;
```

εκεί μέσα μεταξύ των άλλων και οι
DocumentBuilderFactory
DocumentBuilder

Η Class DocumentBuilder είναι της java και με την μέθοδο της parse("*filename*.xml") επιστρέφει Document, είναι δηλαδή ο parser

Αντικείμενο της DocumentBuilder φτιάχνομε με την DocumentBuilderFactory

Το αντικείμενο Document που επιστρέφεται, πραγματοποιεί (implements) ΠΑΝΤΑ όλα τα interfaces του org.w3c.dom

(τα Node, Element, Attr, NodeList, NamedNodeMap, Document, … που είδαμε είναι όλα interfaces),

⇨ άρα εμείς έχομε την δυνατότητα να γράφομε

```
String nameOfcurrAttribute = currAttribute.getName();
```

… διότι το αντικείμενο **currAttribute** το έφτιαξε (κάπου πιο πριν) ο DOMParser, άρα **implements Attr**, άρα γνωρίζει και εκτελεί την μέθοδο **getName**

Μια ‘Επίδειξη – Read Only’ με DOM API

κώδικας DOMNavigator.java που χρησιμοποιεί myDOMTreeProc.java

Ο κώδικας DOMNavigator.java δημιουργεί από το vehicleTypes.xml το σχετικό DOM.

Επ’ αυτού, με την myDOMTreeProc, δοκιμάζομε τις πιο κοινές μεθόδους που μας διαθέτει το DOM API.

Μεταγλώττιση με **javac DOMNavigator.java οπότε δημιουργούνται DOMNavigator.class και myDOMTreeProc.class**

Τρέξιμο με **java DOMNavigator**

Μέσα στο folder όπου ευρίσκονται οι κώδικες DOMNavigator.java, myDOMTreeProc.java και το vehicleTypes.xml

Βασικές Αρχές

Θα εισάγομε σε DOM οποιοδήποτε έγγραφο .xml (εδώ το vehicleTypes.xml) και θα το περιδιαβούμε κόμβο προς κόμβο, όπου ανάλογα με τον τύπο του κάθε κόμβου θα εξάγομε τις ανάλογες πληροφορίες.

Αντικείμενο της DocumentBuilder θα μας φέρει το έγγραφο στην μνήμη - δηλαδή

η κατάληξη της διαδικασίας parse (κατά DOM) θα είναι ένα αντικείμενο τους κλάσης org.w3c.dom.Document

Το αντικείμενο org.w3c.dom.Document είναι ένα επίπεδο πάνω από το document element και περικλείει όλο το δένδρο των κόμβων

Εφόσον το Node περιλαμβάνει όλους τους τύπους κόμβων μπορούμε σχεδόν αποκλειστικά να δουλεύομε με αυτό και τα NodeList και NamedNodeMap που μπορούμε από αυτό να εξάγομε.

Εδώ μόνον θα διαβάζομε, χωρίς να πειράξομε / μετατρέψομε το έγγραφό μας

Μια 'Επίδειξη – Read Only' με DOM API


```
import org.w3c.dom.*;
import javax.xml.parsers.*;
import java.io.*;

public class DOMNavigator{
  public static void main(String[] args) {
    try{
       DocumentBuilderFactory fact = DocumentBuilderFactory.newInstance();
       DocumentBuilder builder = fact.newDocumentBuilder();
       Document doc = builder.parse("vehicleTypes.xml");

       //Examine document element
       System.out.print("Name of document element is.. ");
       System.out.print(doc.getDocumentElement().getNodeName());
       System.out.print(" ... and its value is ..  "); //this must be NULL !!
       System.out.println(doc.getDocumentElement().getNodeValue());
       System.out.print("No of elements named 'jeep' anywhere in the doc is ..  ");
       System.out.println(doc.getElementsByTagName("jeep").getLength());

       //Iteratively examine every element, starting again from doc element
       myDOMTreeProc dtp = new myDOMTreeProc();
       dtp.processNode(doc.getDocumentElement());
    }
    catch (Exception e){
      e.getMessage();
    }
  }
}
```

Το builder της class DocumentBuilder είναι ο parser της Java

Η εξαγωγή του αντ. org.w3c.dom.Document

(βλ. προηγούμενο slide)

Μια 'Επίδειξη – Read Only' με DOM API


```
class myDOMTreeProc
{
public void processNode(Node el)
{ System.out.println("\n////////////////////////////// ITERATIVE NODE PROCESSING ////////////////////");
  NodeList mixedContent = el.getChildNodes();
   int noOfChildren = mixedContent.getLength();
  System.out.println("Element named '" + el.getNodeName() +
        "' with parent '" + el.getParentNode().getNodeName() +
        "' has " + noOfChildren + " child nodes (of any kind) and " +
        " has content:");

   for ( int i = 0 ; i < noOfChildren ; i++ )
    { Node currNode = mixedContent.item(i);
      if (currNode.getNodeType() == Node.TEXT_NODE)
      { System.out.println(currNode.getNodeType() + "-Text.............: " + currNode.getNodeValue()); };
      if (currNode.getNodeType() == Node.COMMENT_NODE)
      { System.out.println(currNode.getNodeType() + "-Comment.......: " + currNode.getNodeValue()); };
      if (currNode.getNodeType() == Node.PROCESSING_INSTRUCTION_NODE)
      { System.out.println(currNode.getNodeType() + "-PI.....................: " + currNode.getNodeValue()); };
      if (currNode.getNodeType() == Node.CDATA_SECTION_NODE)
      { System.out.println(currNode.getNodeType() + "-CDATA.............: " + currNode.getNodeValue()); };
      if (currNode.getNodeType() == Node.ELEMENT_NODE)
      {   NamedNodeMap allAttr = currNode.getAttributes();
          System.out.println(currNode.getNodeType() + "-Element '"
                          + ((Element)currNode).getTagName() + "' has "+ allAttr.getLength() +" attribute(s): ");
                          // to use the Element method 'getTagName()', casting was used above
          for ( int k = 0 ; k < allAttr.getLength() ; k++ )
          { System.out.println(allAttr.item(k).getNodeName() + " = " +
                                                        allAttr.item(k).getNodeValue());}
            processNode(currNode); ; // recursive call to this same method !!!!
       }
     }
}
}
```

All cases uniformly handled as 'Node'

Όταν πέφτομε σε Element node καλούμε αναδρομικά (μόνον τότε δυνατόν να υπάρχουν παιδιά !)

Δύο βασικά θέματα

1. Όχι μόνο να διαβάζομε, αλλά και να πειράζομε / μετατρέπουμε το έγγραφό μας

2. Μία δομή κατά DOM που ευρίσκεται στην μνήμη θέλομε να εξαχθεί σε έγγραφο (αρχείο) xml ---- Serialization

Το αντικείμενο Document είναι πάλι πρωταρχικό (όπως και στο διάβασμα)**:**

Για προσθήκη ενός π.χ. υποδένδρου, φτιάχνομε ένα νέο DOM Document, γεμίζομε το περιεχόμενό του με τα διαφορετικά createXXX, μετά πάμε στο υπάρχον αρχικό Document, και καλούμε την importNode αυτού (deep true/false ανάλογα αν το υποδένδρο εις βάθος)

κώδικας CreateDOMandXML.java

Το αντικείμενο Document είναι πάλι πρωταρχικό

(όπως και στο διάβασμα):

Ενώ πριν

```
DocumentBuilderFactory fact = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = fact.newDocumentBuilder();
Document doc = builder.parse("vehicleTypes.xml");
```

Τώρα

```
DocumentBuilderFactory fact = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = fact.newDocumentBuilder();
Document doc = builder.newDocument();
```

κατ' αρχήν κενό αλλά θα το γεμίσομε

Εξαγωγή από έγγραφο

Δημιουργία νέου

αντικειμένου
org.w3c.dom.Document

(ευρίσκεται μέσα στην java)

Το αντικείμενο Document πρέπει τώρα να γίνει αρχείο!
- για να βγει έξω

(εδώ γράφομε!):

Φτιάχνομε Transformer tr

```
TransformerFactory trFac = TransformerFactory.newInstance();
Transformer tr = trFac.newTransformer();
DOMSource source = new DOMSource(doc);
StreamResult result =  new StreamResult(new File("newXmlFile.xml"));
 tr.transform(source, result);
```

...που θέλει source
(το doc που που φτιάχθηκε στην μνήμη)

και result (το νέο αρχείο εγγραφής)

Με javax.xml.transform θα ασχοληθούμε διεξοδικά στο τέλος

Για τα παραπάνω:

```
import javax.xml.transform.*;
import javax.xml.transform.dom.*;
import javax.xml.transform.stream.*;
```

Μια 'Επίδειξη – Write' με DOM API


```
import org.w3c.dom.*;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.*;
import javax.xml.transform.stream.*;
import java.io.*;

public class CreateDOMandXML
{
    public static void main(String[] args)
    {
    System.out.println("Creating DOM, filling DOM, outputting XML file.");
    try{
    DocumentBuilderFactory fact = DocumentBuilderFactory.newInstance();
    DocumentBuilder builder = fact.newDocumentBuilder();
    Document doc = builder.newDocument();
    Element top = doc.createElement("topElement");
    doc.appendChild(top);
    Element flchild = doc.createElement("firstLevelChildElement");
    flchild.setAttribute("att1","some value");
    top.appendChild(flchild);
    TransformerFactory trFac = TransformerFactory.newInstance();
    Transformer tr = trFac.newTransformer();
    DOMSource source = new DOMSource(doc);
    StreamResult result =  new StreamResult(new File("newXmlFile.xml"));
    tr.transform(source, result);

        }catch(Exception e){System.out.println(e.getMessage());}
    System.out.println("Done...");
  }
}
```

κατασκευή doc σαν γενικό αποδοχέα

γέμισμα με περιεχόμενο (κοινό DOM)

μετατροπή σε αρχείο, με μέθοδο transfrom τηςTransformer

Μια 'Επίδειξη – READ / WRITE' με DOM API

Απλός συνδυασμός / συγχώνευση των δύο προηγουμένων περιπτώσεων

DOMNavigator και CreateDOMandXML

Διαβάζομε από αρχείο και δημιουργούμε το Document

```
import org.w3c.dom.*;
import javax.xml.parsers.*;
import java.io.*;
```

Μετατρέπομε σε Αρχείο

```
import javax.xml.transform.*;
import javax.xml.transform.dom.*;
import javax.xml.transform.stream.*;
import java.io.*;
```

Αλλαγές /προσθήκες με DOM

```
import org.w3c.dom.*;
```

Για transformer θα δούμε περισσότερα στο κεφάλαιο περί XSL