Βιομηχανοποίηση Μετασχηματισμών

Μέχρι τώρα, για κάθε μετασχηματισμό απαιτείτο

- το διάβασμα (parsing) του εγγράφου XSL,
- η δημιουργία του σχετικού DOM,
- το διάβασμα (parsing) του εγγράφου εισόδου XML,
- η δημιουργία του σχετικού DOM,

… και όλα αυτά μόνο τα προκαταρκτικά για τον ίδιο τον μετασχηματισμό

Αλλά στην πράξη πολλά από τα παραπάνω είναι

είτε ενδιάμεσα στάδια

(π.χ. το προς μετασχηματισμό έγγραφο να ευρίσκεται ήδη σαν DOM στην μνήμη, σαν κάποιο προηγούμενο αποτέλεσμα)

είτε κοινός παρανομαστής

(π.χ. το ίδιο εγγράφου XSL μπορεί να είναι εφαρμοστέο σε χιλιάδες διαφορετικά XML)

… και πάλι η έξοδος (Result) του μετασχηματισμού μπορεί

να μην είναι αναγκαστικά αρχείο, αλλά (στην περίπτωση εξόδου .xml) να είναι ένα DOM ή μία ακολουθία parse events

Βιομηχανοποίηση Μετασχηματισμών

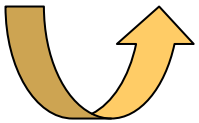
Τα parser events και το DOM δεν πρέπει μόνο να τα βλέπομε ***σαν αποτελέσματα***

... μπορούν να είναι και ***είσοδοι για ένα επόμενο βήμα***

Έτσι έχομε τρεις δυνατές πηγές / προορισμούς XML

DOMSource	**DOMResult**
SAXSource	**SAXResult**
StreamSource	**StreamResult**

πολλές δυνατότητες



… όπου και το .xslt είναι XML

Είναι φανερό ότι το καθένα από την κάθε τριάδα είναι ισοδύναμο όσον αφορά τον μετασχηματισμό

Βιομηχανοποίηση Μετασχηματισμών

Και επιπλέον μας δίδονται πολλές δυνατότητες για **StreamSource**

Μερικές:

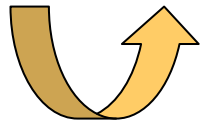
StreamSource(File f**)**	**Construct a StreamSource from a File**
StreamSource(InputStream inputStream**)**	**Construct a StreamSource from a byte stream**
StreamSource(Reader reader**)**	**Construct a StreamSource from a character reader**
StreamSource(String systemId**)**	**Construct a StreamSource from a URL**

... όπως και για **StreamResult**

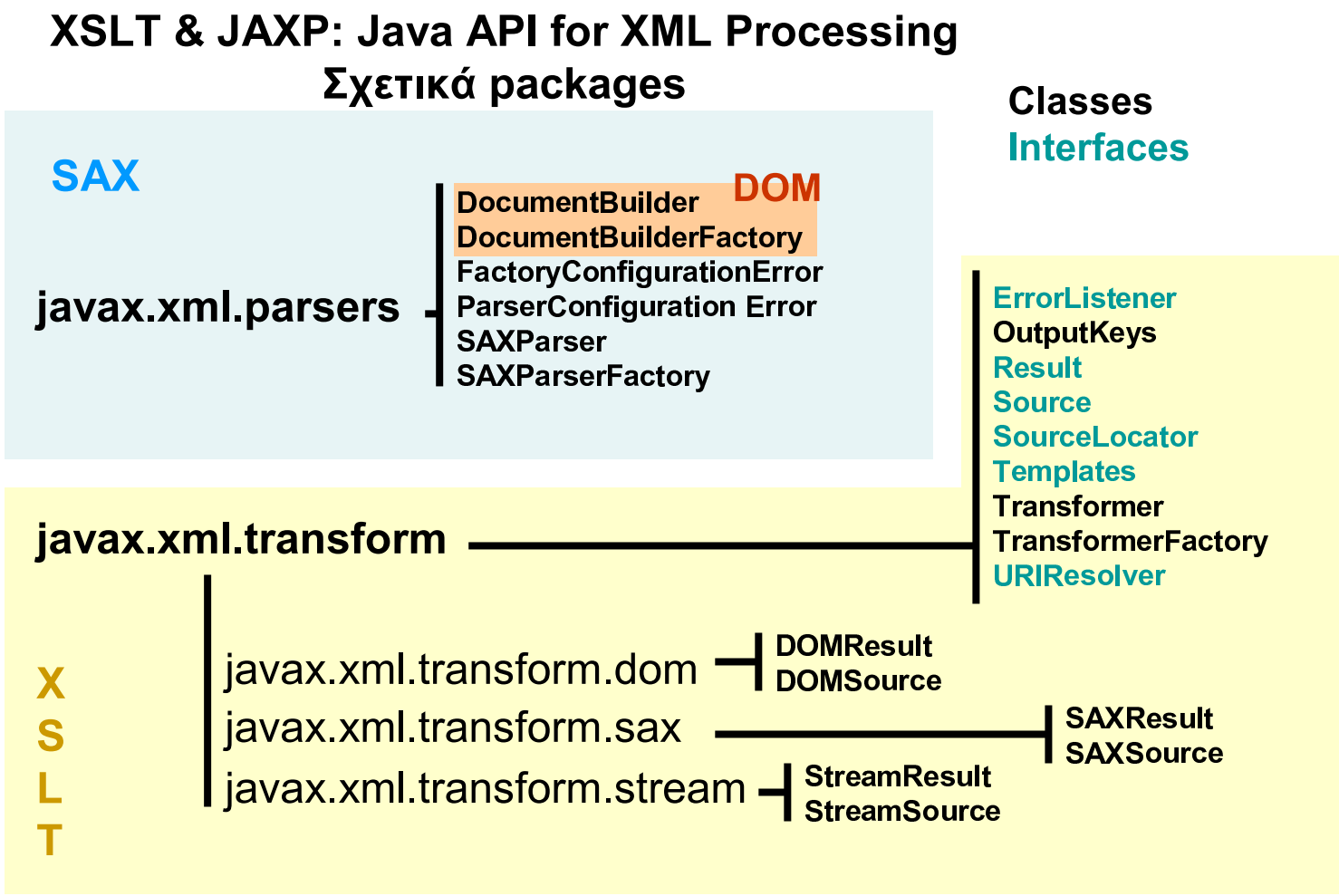
StreamResult(File f**)**	**Construct a StreamResult as a File**
StreamResult(OutputStream outputStream**)**	**Construct a StreamResult as a byte stream**
StreamResult(Writer writer**)**	**Construct a StreamResult as a character writer**
StreamResult(String systemId**)**	**Construct a StreamResult as a URL**

Άρα εξετάζομε τι μας προσφέρει το

Java API for XML Processing



Βιομηχανοποίηση Μετασχηματισμών



… φυσικά όλα τα ανωτέρω εμπεριέχονται στην java

Βιομηχανοποίηση Μετασχηματισμών

Δυο διακριτά βήματα

Δημιουργία μετασχηματιστή

newTransformer(Source source)

- **DOMSource**
- **SAXSource**
- **StreamSource**

Εφαρμογή μετασχηματιστή

transform(**Source** xmlSource, **Result** outputTarget)

- **DOMSource**
- **SAXSource**
- **StreamSource**

- **DOMResult**
- **SAXResult**
- **StreamResult**

... όπου είδαμε την πληθώρα των **StreamSource**

Βιομηχανοποίηση Μετασχηματισμών

Παράδειγμα: Ο μετασχηματισμός χαρακτηρίζεται από μία **SAXSource**

```
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 org.xml.sax.*;

public class XMLIndustrialTransformer {
   public static void main(String args[]) throws Exception
     { TransformerFactory myTransformerFactory = TransformerFactory.newInstance();
       SAXSource myTransformer_s_Source
                = new SAXSource(new InputSource(new FileReader("CarPresentor2.xsl")));
       Transformer myTransformer
                = myTransformerFactory.newTransformer(myTransformer_s_Source);

       StreamResult transformResult = new StreamResult(new FileOutputStream("Cars.html"));
       StreamSource xmlSource=  new StreamSource(new FileInputStream("Cars.xml"));
       myTransformer.transform(xmlSource, transformResult);
     }
}
```

φτιάχνεται **μία φορά** ένας **αμετάβλητος μετασχηματιστής**

που έχει μέσα του **ενσωματωμένο** ότι ζητείται από **το .xsl**

Αυτό μπορεί να εφαρμόζεται σε επαναληπτικά σε πολλά .xml εισόδου παράγοντας αντίστοιχα αποτελέσματα

Βιομηχανοποίηση Μετασχηματισμών

Το προηγούμενο παράδειγμα '**παράλληλη περίπτωση**'

(ένας κοινός μετασχηματισμός για πολλά .xml εισόδου)

Πολλά άλλα πιθανά σενάρια

Ο μετασχηματισμός αντί StreamResult να παράγει SAXResult, το οποίο εκλαμβάνεται σαν SAXSource για ένα επόμενο '**σειριακή περίπτωση**'

DOMSource αντί για SAXSource, κλπ

κλπ