XML i tehnologije:

računalna obrada XML dokumenta

SAX i DOM

- 2 osnovne paradigme za izgradnju aplikacijskih programskih sučelja (API) za XML:
  - SAX (Simple API for XML)
  - DOM (Document Object Model)

SAX – Simple API for XML

- sekvencijalno čitanje od početka do kraja dokumenta; segmenti se obrađuju redoslijedom njihovog pojavljivanja

```
<person>
  <name>Nikola Tesla</name>
  <email>tesla@tesla.com</email>
</person>
```

Ovisno o duljini spremnika (*buffer*) koja ovisi o implementaciji, moguće je da SAX registrira više uzastopnih tekstualnih čvorova

SAX:
1. Počinje XML dokument
2. Počinje element **person**
3. Tekst *novi redak*
4. Počinje element **name**
5. Tekst Nikola Tesla
6. Završava element **name**
7. Tekst *novi redak*
8. Počinje element **email**
9. Tekst **tesla@tesla.com**
10. Završava element **email**
11. Tekst *novi redak*
12. Završava element **person**
13. Završava XML dokument

SAX – Simple API for XML

- ne postoji službena specifikacija
- standard *de facto* je izvedba u jeziku Java
- preporučuje se programski paket **Xerces** tvrtke *Apache* (otvoreni kod, javno dostupan, ugrađen u mnogim sustavima za razvoj programa)

Packages

org.apache.html.dom
org.apache.wml
org.apache.wml.dom
org.apache.xerces.dom
org.apache.xerces.dom.events
org.apache.xerces.domx
org.apache.xerces.framework
org.apache.xerces.jaxp
org.apache.xerces.msg
org.apache.xerces.parsers
org.apache.xerces.readers
org.apache.xerces.utils
org.apache.xerces.utils.regex
org.apache.xerces.validators.common
org.apache.xerces.validators.datatype
org.apache.xerces.validators.dtd
org.apache.xerces.validators.schema
org.apache.xerces.validators.schema.identity
org.apache.xml.serialize
org.w3c.dom
org.w3c.dom.events
org.w3c.dom.html
org.w3c.dom.ranges
org.w3c.dom.traversal
org.xml.sax
org.xml.sax.ext
org.xml.sax.helpers

DOM – Document Object Model

- platforma za obradu HTML i XML dokumenata
- standard organizacije W3C, postoje 3 razine
  - Za 3. razinu vrijedi trodijelna specifikacija

1. Document Object Model (DOM) Level 3 Core Specification, Version 1.0 (W3C Recommendation 07 April 2004) http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407
2. Document Object Model (DOM) Level 3 Load and Save Specification, Version 1.0 (W3C Recommendation 07 April 2004) http://www.w3.org/TR/2004/REC-DOM-Level-3-LS-20040407
3. Document Object Model (DOM) Level 3 Validation Specification, Version 1.0 (W3C Recommendation 27 January 2004) http://www.w3.org/TR/2004/REC-DOM-Level-3-Val-20040127

- Najnovije važeće (uključujući i buduće) verzije specifikacija nalaze se na adresama
  http://www.w3.org/TR/DOM-Level-3-Core
  http://www.w3.org/TR/DOM-Level-3-LS
  http://www.w3.org/TR/DOM-Level-3-Val

DOM – Document Object Model

- DOM, slično kao XPath, promatra XML dokument kao stablo čvorova
  - 12 vrsta čvorova:

Naziv vrste čvora	Postoji u XPathu
korijenski čvor (Document)	DA
fragment (DocumentFragment)	NE
tip dokumenta (DocumentType)	NE
entitet (Entity)	NE
poveznica s entitetom (EntityReference)	NE
element (Element)	DA
atribut (Attr)	DA
procesorska instrukcija (ProcessingInstruction)	DA
komentar (Comment)	DA
tekst (Text)	DA
CDATASection	NE
zabilješka (Notation)	NE

- entitet i poveznica s entitetom
  - prije pojave XML Scheme "posebna" slova nacionalnih alfabeta definirala su se kao entiteti, npr. **é** u riječi **Montréal**

```
<?xml version="1.0" ?>
<!DOCTYPE purchaseOrder [
<!ENTITY eacute "&#xE9;">
]>
<purchaseOrder xmlns=http://www.example.com/PO1  orderDate="1999-10-20">
  <!-- etc. -->
    <city>Montr&eacute;al</city>
  <!-- etc. -->
</purchaseOrder>
```

```
XML SCHEMA: <xsd:element name="eacute" type="xsd:token" fixed="&#xE9;"/>

<?xml version="1.0" ?>
<purchaseOrder xmlns="http://www.example.com/PO1"
           xmlns:c="http://www.example.com/characterElements"
           orderDate="1999-10-20">
  <!-- etc. -->
    <city>Montr<c:eacute/>al</city>
  <!-- etc. -->
</purchaseOrder>
```

DOM – programske izvedbe

- preporučuje se ranije spomenuti programski paket **Xerces**

Packages
org.apache.html.dom
org.apache.wml
org.apache.wml.dom
org.apache.xerces.dom
org.apache.xerces.dom.events
org.apache.xerces.domx
org.apache.xerces.framework
org.apache.xerces.jaxp
org.apache.xerces.msg
org.apache.xerces.parsers
org.apache.xerces.readers
org.apache.xerces.utils
org.apache.xerces.utils.regex
org.apache.xerces.validators.common
org.apache.xerces.validators.datatype
org.apache.xerces.validators.dtd
org.apache.xerces.validators.schema
org.apache.xerces.validators.schema.identity
org.apache.xml.serialize
org.w3c.dom
org.w3c.dom.events
org.w3c.dom.html
org.w3c.dom.ranges
org.w3c.dom.traversal
org.xml.sax
org.xml.sax.ext
org.xml.sax.helpers

SAX ili DOM

- SAX je brži i lako se postigne da se ne gubi vrijeme na nevažne dijelove XML dokumenata, DOM zahtijeva da se učita cijelo stablo
- DOM omogućuje kretanje po stablu s čvora na čvor i laganu malipulaciju čvorovima (dodavanje novih čvorova, brisanje čvorova)

Primjer 1 – SAX

- Za dokumente čija je XML Schema zadana dolje načiniti program koji će ispisivati imena (*name*) svih kupaca (*customer*)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="order" type="OrderType"/>
   <xsd:complexType name="OrderType">
     <xsd:sequence>
       <xsd:element name="customer" type="CustomerType"
                                          maxOccurs="unbounded"/>
     </xsd:sequence>
     <xsd:attribute name="orderID" type="xsd:integer"
                                                 use="required"/>
     <xsd:attribute name="orderDate" type="xsd:date"/>
   </xsd:complexType>
   <xsd:complexType name="CustomerType">
     <xsd:sequence>
       <xsd:element name="name" type="xsd:string"/>
       <xsd:element name="address" type="xsd:string"/>
       <xsd:element name="zip" type="xsd:string"/>
       <xsd:element name="city" type="xsd:string"/>
       <xsd:element name="country" type="xsd:string"/>
       <xsd:element name="email" type="xsd:string"
              minOccurs="0" maxOccurs="unbounded"/>
     </xsd:sequence>
     <xsd:attribute name="ID" type="xsd:integer" use="required"/>
   </xsd:complexType>
</xsd:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<order orderID="234" orderDate="2007-05-31"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:noNamespaceSchemaLocation
        ="C:\JDeveloper\jdev\mywork\TEP\Project1\schemaA.xsd">
  <customer ID="1">
    <name>Nikola Tesla</name>
    <address>1234 Broadway</address>
    <zip>NY1234</zip>
    <city>New York</city>
    <country>USA</country>
  </customer>
  <customer ID="2">
    <name>Michael Faraday</name>
    <address>12 Trafalgar Square</address>
    <zip>L5Z67</zip>
    <city>London</city>
    <country>UK</country>
    <email>michael.faraday@gmail.com</email>
    <email>mick@hotmail.com</email>
  </customer>
</order>
```


Primjer 1 – SAX

```
import org.xml.sax.SAXException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.XMLReaderFactory;
import java.io.IOException;

public class SAXApplication {

public static String[] process(String sourceUri){
      try {
          XMLReader reader=XMLReaderFactory.createXMLReader();
          MySAXHandler handler=new MySAXHandler();
          reader.setContentHandler(handler);
          reader.setErrorHandler(handler);
          reader.parse(sourceUri);
          return handler.getNames();
      } catch (SAXException saxe) {
          saxe.printStackTrace();
      } catch(IOException ioe){
          ioe.printStackTrace();
      }
      return new String[0];
   }

   public static void main(String[] args){
      String path="C:/JDeveloper/jdev/mywork/TEP/Project1/fileA-1.xml";
      java.io.File file=new java.io.File(path);
      java.net.URI uri=file.toURI();
      String[] names=process(uri.toString());
      System.out.println("\nIspis dohvacenih imena:");
      for(int j=0; j<names.length; j++)
          System.out.println("-->"+names[j]);
   }

}
```

Čitač (*parser*) je instanca klase *XMLReader (konkretna klasa com.sun.org.apache.xerces.internal.parsers.SAXParser)*; stvara se uz pomoć klase *XMLReaderFactory*

Način sekvencijalne obrade XML dokumenta specificira zasebna klasa koja mora implementirati sučelja *org.xml.sax.ContentHandler* i *org.xml.sax.ErrorHandler*; za instancu te klase potrebno je pozvati metode setContentHandler i setErrorHandler;

u praksi, spomenuta klasa nasljeđuje klasu *org.xml.sax.helpers.DefaultHandler* koja implementira gornja dva sučelja

URI SE MOŽE ODNOSITI I NA IZVOR KOJI JE ON-LINE:
java.net.URI fer=new java.net.URI(
"http://en.wikipedia.org/wiki/Special:Export/University_of_Zagreb_Faculty_of_Electrical_Engineering_and_Computing");
U TOM SLUČAJU RAČUNALO ĆE SE (AKO JE MOGUĆE) AUTOMATSKI SPOJITI NA ZADANU ADRESU I DOHVATITI XML DOKUMENT

Primjer 1 – SAX

- Metode sučelja org.xml.sax.ContentHandler i org.xml.sax.ErrorHandler

org.xml.sax.ContentHandler	funkcija
void **startDocument**()	početak XML dokumenta
void **endDocument**()	završetak XML dokumenta
void **startElement**(String namespaceURI, String localName, String qName, Attributes atts)	početak elementa *qname* s listom atributa *atts*
void **endElement**(String namespaceURI, String localName, String qName)	završetak elementa *qname*
void **characters**(char[] ch, int start, int length)	dohvat tekstualnog sadržaja
void ignorableWhitespace(char[] ch, int start, int length)	pojava razmaka koji se mogu ignorirati
void processingInstruction(String target, String data)	pojava procesorske instrukcije
void startPrefixMapping(String prefix, String uri)	početak mapiranja URI-ja *uri* na prefiks *prefix*
void endPrefixMapping(String prefix)	završetak mapiranja na prefiks *prefix*
void skippedEntity(String name)	entitet kojeg se preskače
void setDocumentLocator(Locator locator)	podaci o lokaciji dokumenta na računalu

org.xml.sax.ErrorHandler	funkcija
void fatalError(SAXParseException exception)	greška nakon koje nije moguće nastaviti obradu (npr. <a>)
void error(SAXParseException exception)	greška nakon koje je moguće nastaviti obradu
void warning(SAXParseException exception)	upozorenje

Primjer 1 – SAX

```
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

public class MySAXHandler extends DefaultHandler {

   boolean nameFlag=false;
   String nameValue="";
   java.util.List nameList=new java.util.ArrayList();

   public void startElement(String namespaceUri,
                            String localName,
                            String qualifiedName,
                            Attributes atts) throws SAXException {
      System.out.println("Pocetak elementa: "+qualifiedName);
      if(qualifiedName.compareTo("name")==0){
        nameFlag=true;
      }
   }

   public void characters(char[] chars,
                          int startIndex,
                          int length) {
      boolean alphaOrNumeric=false;
      for(int j=startIndex; j<startIndex+length; j++){
        if(Character.isLetterOrDigit(chars[j])){
           alphaOrNumeric=true;
           break;
        }
      }
      if(!alphaOrNumeric)
        return;
      String realContent=new String(chars, startIndex, length);
      System.out.println("sadrzaj:*"+realContent+"*");
      if(nameFlag){
        nameValue+=realContent;
      }
   }

   // na idućoj foliji  public void endElement
   // na idućoj foliji public String[] getNames
}
```

Klasa *org.xml.sax.helpers.DefaultHandler* sve sve metode sučelja *org.xml.sax.ContentHandler* i *org.xml.sax.ErrorHandler* ostavlja prazne (ne rade ništa) osim *fatalError* koja baca iznimku koja ju je i uzrokovala

Zastavica koja se podiže u trenutku kad započne obrada elementa *name*, a spušta završetkom njegove obrade; inicijalno spuštena

Spremnik za vrijednost elementa *name*; inicijalno prazan

Početak obrade svakog elementa; ako ime elementa nije *name*, ne čini ništa; ako ime jest *name*, podigni zastavicu

Obrada svakog tekstualnog sadržaja; ako sadržaj u sebi nema nijedan broj ili slovo (npr. samo novi redak), prekini obradu

Sadržaj koji sadrži bar jedan broj ili slovo spremi u varijablu *realContent*; ako je zastavica podignuta, prilijepi sadržaj na postojeću vrijednost u spremniku *nameValue* (ovisno o duljini svojeg internog spremnika koja ovisi o implementaciji, moguće je da SAX registrira više uzastopnih tekstualnih čvorova)

Primjer 1 – SAX

```
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

public class MySAXHandler extends DefaultHandler {

   // na prethodnoj foliji public void startElement
   // na prethodnoj foliji public void characters

   public void endElement(String namespaceUri,
                          String localName,
                          String qualifiedName)
         throws SAXException {
      System.out.println("Zavrsetak elementa: "+qualifiedName);
      if(nameFlag){
        nameList.add(nameValue);
        nameFlag=false;
        nameValue="";
      }
   }

   public String[] getNames(){
      String[] ret=new String[nameList.size()];
      java.util.Iterator iter=nameList.iterator();
      for(int j=0; iter.hasNext(); j++){
        ret[j]=(String)iter.next();
      }
      return ret;
   }

}
```

Završetak obrade elementa; ako ime elementa nije *name*, ne čini ništa; ako je ime elementa *name*, spremi sadržaj spremnika *nameValue* u odgovarajuću listu; isprazni spremnik *nameValue* i spusti zastavicu

Procedura za dohvat svih imena iz liste; očekivani poziv nakon obrade XML dokumenta

Primjer 1 – SAX

- Ispis programa za dani XML dokument

```
<?xml version="1.0" encoding="UTF-8"?>
<order orderID="234" orderDate="2007-05-31"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:noNamespaceSchemaLocation
        ="C:\JDeveloper\jdev\mywork\TEP\Project1\schemaA.xsd">
  <customer ID="1">
    <name>Nikola Tesla</name>
    <address>1234 Broadway</address>
    <zip>NY1234</zip>
    <city>New York</city>
    <country>USA</country>
  </customer>
  <customer ID="2">
    <name>Michael Faraday</name>
    <address>12 Trafalgar Square</address>
    <zip>L5Z67</zip>
    <city>London</city>
    <country>UK</country>
    <email>michael.faraday@gmail.com</email>
    <email>mick@hotmail.com</email>
  </customer>
</order>
```

```
Pocetak elementa: order
Pocetak elementa: customer
Pocetak elementa: name
sadrzaj:*Nikola Tesla*
Zavrsetak elementa: name
Pocetak elementa: address
sadrzaj:*1234 Broadway*
Zavrsetak elementa: address
Pocetak elementa: zip
sadrzaj:*NY1234*
Zavrsetak elementa: zip
Pocetak elementa: city
sadrzaj:*New York*
Zavrsetak elementa: city
Pocetak elementa: country
sadrzaj:*USA*
Zavrsetak elementa: country
Zavrsetak elementa: customer
Pocetak elementa: customer
Pocetak elementa: name
sadrzaj:*Michael Faraday*
Zavrsetak elementa: name
Pocetak elementa: address
sadrzaj:*12 Trafalgar Square*
Zavrsetak elementa: address
Pocetak elementa: zip
sadrzaj:*L5Z67*
Zavrsetak elementa: zip
Pocetak elementa: city
sadrzaj:*London*
Zavrsetak elementa: city
Pocetak elementa: country
sadrzaj:*UK*
Zavrsetak elementa: country
Pocetak elementa: email
sadrzaj:*michael.faraday@gmail.com*
Zavrsetak elementa: email
Pocetak elementa: email
sadrzaj:*mick@hotmail.com*
Zavrsetak elementa: email
Zavrsetak elementa: customer
Zavrsetak elementa: order

Ispis dohvacenih imena:
-->Nikola Tesla
-->Michael Faraday
```

Primjer 2 – DOM

- Analogno primjeru koji je koristio SAX, za dokumente zadane donjom XML Schema načiniti program koji će ispisivati imena (*name*) svih kupaca (*customer*)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="order" type="OrderType"/>
   <xsd:complexType name="OrderType">
     <xsd:sequence>
       <xsd:element name="customer" type="CustomerType"
                                        maxOccurs="unbounded"/>
     </xsd:sequence>
     <xsd:attribute name="orderID" type="xsd:integer"
                                                 use="required"/>
     <xsd:attribute name="orderDate" type="xsd:date"/>
  </xsd:complexType>
  <xsd:complexType name="CustomerType">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="address" type="xsd:string"/>
      <xsd:element name="zip" type="xsd:string"/>
      <xsd:element name="city" type="xsd:string"/>
      <xsd:element name="country" type="xsd:string"/>
      <xsd:element name="email" type="xsd:string"
            minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="ID" type="xsd:integer" use="required"/>
  </xsd:complexType>
</xsd:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<order orderID="234" orderDate="2007-05-31"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:noNamespaceSchemaLocation
        ="C:\JDeveloper\jdev\mywork\TEP\Project1\schemaA.xsd">
  <customer ID="1">
    <name>Nikola Tesla</name>
    <address>1234 Broadway</address>
    <zip>NY1234</zip>
    <city>New York</city>
    <country>USA</country>
  </customer>
  <customer ID="2">
    <name>Michael Faraday</name>
    <address>12 Trafalgar Square</address>
    <zip>L5Z67</zip>
    <city>London</city>
    <country>UK</country>
    <email>michael.faraday@gmail.com</email>
    <email>mick@hotmail.com</email>
  </customer>
</order>
```


Primjer 2 – DOM

```
import javax.xml.parsers.DocumentBuilder; import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException; import org.w3c.dom.Document; import org.w3c.dom.Element;
import org.w3c.dom.Node; import org.w3c.dom.NodeList; import org.xml.sax.SAXException;

public class DOMApplication {

   public static String[] process(String sourceUri) {
      java.util.List names=new java.util.ArrayList();
      try {
         DocumentBuilderFactory factory=DocumentBuilderFactory.newInstance();
         DocumentBuilder builder=factory.newDocumentBuilder();
         Document doc=builder.parse(sourceUri);
         Element order=doc.getDocumentElement();
         NodeList customerList=order.getElementsByTagName("customer");
         for(int cIndex=0; cIndex<customerList.getLength(); cIndex++){
            Element cust=(Element)customerList.item(cIndex);
            NodeList nameList=cust.getElementsByTagName("name");
            for(int nIndex=0; nIndex<nameList.getLength(); nIndex++){
               Element name=(Element)nameList.item(nIndex);
               Node textNode=name.getFirstChild();
               if(textNode==null)
                  continue;
               if(textNode.getNodeType()!=Node.TEXT_NODE)
                  continue;
               String value=textNode.getNodeValue();
               names.add(value);
            }
         }
         String ret[]=new String[names.size()];
         java.util.Iterator i=names.iterator();
         for(int j=0; i.hasNext(); j++)
            ret[j]=(String)i.next();
         return ret;
      } catch (ParserConfigurationException pce) {
         pce.printStackTrace();
      } catch(java.io.IOException ioe){
         ioe.printStackTrace();
      } catch(SAXException saxe){
         saxe.printStackTrace();
      }
      return new String[0];
   }
```

Instanca apstrakne klase *DocumentBuilder (konkretna klasa com.sun.org.apache.xerces.internal.jaxp.DocumentBuilderImpl)*; stvara se uz pomoć klase *DocumentBuilderFactory*

getDocumentElement vraća korijenski element XML dokumenta (*order*)

lista sve djece elementa (varijable) *order* čije je ime *customer*

lista sve djece elementa (varijable) *cust* čije je ime *name*

budući da element *name* po XML Schemi nema podelemenata nego samo sadržaj, čvor sadržaja je prvo (i jedino) dijete elementa *name*; ipak, najprije provjeri postoji li uopće prvo tj. ijedno dijete elementa *name* te je li ono uistinu tekstualni sadržaj

Primjer 2 – DOM

```
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;


public class DOMApplication {

  //na prethodnoj foliji public static String[] process(String sourceUri)

 public static void main(String[] args){
       // obradjujem  datoteku fileA-1.xml
       String path="C:/JDeveloper/jdev/mywork/TEP/Project1/fileA-1.xml";
       java.io.File file=new java.io.File(path);
       java.net.URI uri=file.toURI();
       String[] names=process(uri.toString());
       System.out.println("Ispis dohvacenih imena:");
       for(int j=0; j<names.length; j++)
         System.out.println("-->"+names[j]);
 }

}
```

```
Ispis dohvacenih imena:
-->Nikola Tesla
-->Michael Faraday
```


Primjer 3 – Provjera valjanosti XML dokumenta spram pripadajuće XML Scheme

- programski paket **Xerces** nudi i podršku za provjeru valjanosti XML dokumenta
- U nadolazećem primjeru za objekt klase *javax.xml.validation.Validator* poziva se metoda *setErrorHandler(org.xml.sax.ErrorHandler h)*
  - Objekt *h* je instanca klase koja implementira sučelje *org.xml.sax.ErrorHandler*, najčešće nasljeđuje ranije prikazanu klasu *org.xml.sax.helpers.DefaultHandler*

Primjer 3 – Provjera valjanosti XML dokumenta spram pripadajuće XML Scheme

```
import java.net.URISyntaxException; import javax.xml.XMLConstants; import javax.xml.validation.Schema;
import javax.xml.validation.SchemaFactory; import javax.xml.validation.Validator;
import javax.xml.transform.stream.StreamSource; import org.xml.sax.SAXException;

public class ValidationApplication {

   public static boolean validate(String documentUri, String schemaUri)  {
      try {
         java.net.URI schUri=new java.net.URI(schemaUri);
         java.net.URL schUrl=schUri.toURL();
         SchemaFactory schemaFactory = SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
         Schema schema = schemaFactory.newSchema(schUrl);

         java.net.URI docUri=new java.net.URI(documentUri);
         java.net.URL docUrl=docUri.toURL();
         StreamSource ss=new StreamSource(docUrl.toString());

         MyErrorHandler meh=new MyErrorHandler();
         Validator validator = schema.newValidator();
         validator.setErrorHandler(meh);
         validator.validate(ss);
         if(meh.validationSuccessful()){
            System.out.println("VALIDATION successfull for document with URI "
              +documentUri+"\nconcerning XML Schema with URI "+schemaUri);
            meh.printWarnings();
            return true;
         }
         System.err.println("VALIDATION NOT successfull for document with
    URI "+documentUri+"\nconcerning XML Schema with URI "+schemaUri);
         meh.printErrors();
         meh.printWarnings();
      } catch(java.io.IOException ioe){
         ioe.printStackTrace();
      } catch (SAXException saxe) {
         saxe.printStackTrace();
      } catch(URISyntaxException use){
         use.printStackTrace();
      }
      return false;
   }

   // izostavljena je metoda public static void main
}
```

ulazni argumenti su URI dokumenta i URI XML Scheme

za stvaranje objekta apst. klase *javax.xml.validation.Schema* (pomoću objekta klase *javax.xml.validation.SchemaFactory*) treba URL (ne URI)

XML dokument kojemu se provjerava valjanost potrebno je proslijediti kao objekt klase *javax.xml.transform.stream.StreamSource*

Kao i kod SAX-a, obrada grešaka provodi se nakon poziva metode ***validate*** kroz objekt koji implementira sučelje *org.xml.sax.ErrorHandler*

Ovisno o ishodu provjere valjanosti (zapis o tome u objektu koji implementira sučelje *org.xml.sax.ErrorHandler*), načini odgovarajući ispis i vrati TRUE za valjani ili FALSE za nevaljani dokument

Apache: implementacija apst. klase *javax.xml.validation.Schema* je *com.sun.org.apache.xerces.internal.jaxp.validation.xs.SchemaImpl*; apst. klase *javax.xml.validation.Validator* klasa *com.sun.org.apache.xerces.internal.jaxp.validation.ValidatorImpl*

Primjer 3 – Provjera valjanosti XML dokumenta spram pripadajuće XML Scheme

```
import org.xml.sax.ErrorHandler;          import org.xml.sax.SAXException;          import org.xml.sax.SAXParseException;

public class MyErrorHandler implements ErrorHandler{

      private java.util.List fatalErrList=new java.util.ArrayList();
      private java.util.List errList=new java.util.ArrayList();
      private java.util.List warningList=new java.util.ArrayList();

      public void fatalError(SAXParseException e)  {
         this.fatalErrList.add(e.getMessage());
      }
      public void error( SAXParseException e ) throws SAXException {
         this.errList.add(e.getMessage());
      }
      public void warning( SAXParseException e ) throws SAXException {
         this.warningList.add(e.getMessage());
      }
      public void printErrors(){
        java.util.Iterator i=this.fatalErrList.iterator();
        while(i.hasNext()){
          String message=(String)i.next();
          System.err.println("FATAL ERROR:"+message);
        }
        i=this.errList.iterator();
        while(i.hasNext()){
          String message=(String)i.next();
          System.err.println("ERROR:"+message);
        }
      }
      public void printWarnings(){
        java.util.Iterator i=this.warningList.iterator();
        while(i.hasNext()){
          String message=(String)i.next();
          System.err.println("WARNING:"+message);
        }
      }
      public boolean validationSuccessful(){
        if(this.fatalErrList.size()>0  || this.errList.size()>0)
          return false;
        return true;
      }
}
```

Načini tri liste: po jednu za ključne greške, obične greške te upozorenja

Za pojavu pojedine iznimke koja predstavlja grešku ili upozorenje, dodaj poruku iznimke u odgovarajuću listu

Ispiši sve greške, prvo ključne, potom obične

Ispiši sva upozorenja

Dokument je valjan u odnosu na XML Schemu ako ne postoji niti jedna greška (bilo ključna, bilo obična)

Primjer 3

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  <xsd:element name="order" type="OrderType"/>
   <xsd:complexType name="OrderType">
     <xsd:sequence>
       <xsd:element name="customer" type="CustomerType“
                                        maxOccurs="unbounded"/>
     </xsd:sequence>
     <xsd:attribute name="orderID" type="xsd:integer"
                                                use="required"/>
     <xsd:attribute name="orderDate" type="xsd:date"/>
  </xsd:complexType>
  <xsd:complexType name="CustomerType">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="address" type="xsd:string"/>
      <xsd:element name="zip" type="xsd:string"/>
      <xsd:element name="city" type="xsd:string"/>
      <xsd:element name="country" type="xsd:string"/>
      <xsd:element name="email" type="xsd:string"
              minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="ID" type="xsd:integer" use="required"/>
  </xsd:complexType>
</xsd:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<order orderID="234" orderDate="2007-05-31"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:noNamespaceSchemaLocation
        ="C:\JDeveloper\jdev\mywork\TEP\Project1\schemaA.xsd">
  <customer ID="1">
    <name>Nikola Tesla</name>
    <address>1234 Broadway</address>
    <zip>NY1234</zip>
    <city>New York</city>
    <country>USA</country>
  </customer>
  <customer ID="2">
    <name>Michael Faraday</name>
    <address>12 Trafalgar Square</address>
    <zip>L5Z67</zip>
    <city>London</city>
    <country>UK</country>
    <email>michael.faraday@gmail.com</email>
    <email>mick@hotmail.com</email>
  </customer>
</order>
```

```
public static void main(String[] args){
      // obradjujem  datoteku fileA-1.xml
      String docPath="C:/JDeveloper/jdev/mywork/TEP/Project1/fileA-1.xml";
      java.io.File docFile=new java.io.File(docPath);
      java.net.URI docUri=docFile.toURI();
      String schPathA="C:/JDeveloper/jdev/mywork/TEP/Project1/schemaA.xsd";
      java.io.File schFileA=new java.io.File(schPathA);
      java.net.URI schUriA=schFileA.toURI();
      validate(docUri.toString(), schUriA.toString());
}
```

Prikazani dokument odgovara danoj XML Schemi

```
VALIDATION successful for document with URI file:/C:/JDeveloper/jdev/mywork/TEP/Project1/fileA-1.xml
concerning XML Schema with URI file:/C:/JDeveloper/jdev/mywork/TEP/Project1/schemaA.xsd
```


Primjer 3

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  <xsd:element name="order" type="OrderType"/>
   <xsd:complexType name="OrderType">
     <xsd:sequence>
       <xsd:element name="customer" type="CustomerType"/>
     </xsd:sequence>
     <xsd:attribute name="orderID" type="xsd:integer"
                                                use="required"/>
     <xsd:attribute name="orderDate" type="xsd:date"/>
  </xsd:complexType>
  <xsd:complexType name="CustomerType">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="address" type="xsd:string"/>
      <xsd:element name="zip" type="xsd:string"/>
      <xsd:element name="city" type="xsd:string"/>
      <xsd:element name="country" type="xsd:string"/>
      <xsd:element name="email" type="xsd:string"
              minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="ID" type="xsd:integer" use="required"/>
  </xsd:complexType>
</xsd:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<order orderID="234" orderDate="2007-05-31"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:noNamespaceSchemaLocation
        ="C:\JDeveloper\jdev\mywork\TEP\Project1\schemaA.xsd">
  <customer ID="1">
    <name>Nikola Tesla</name>
    <address>1234 Broadway</address>
    <zip>NY1234</zip>
    <city>New York</city>
    <country>USA</country>
  </customer>
  <customer ID="2">
    <name>Michael Faraday</name>
    <address>12 Trafalgar Square</address>
    <zip>L5Z67</zip>
    <city>London</city>
    <country>UK</country>
    <email>michael.faraday@gmail.com</email>
    <email>mick@hotmail.com</email>
  </customer>
</order>
```

```
public static void main(String[] args){
      // obradjujem  datoteku fileA-1.xml
      String docPath="C:/JDeveloper/jdev/mywork/TEP/Project1/fileA-1.xml";
      java.io.File docFile=new java.io.File(docPath);
      java.net.URI docUri=docFile.toURI();
       String schPathB="C:/JDeveloper/jdev/mywork/TEP/Project1/schemaB.xsd";
      java.io.File schFileB=new java.io.File(schPathB);
      java.net.URI schUriB=schFileB.toURI();
      validate(docUri.toString(), schUriB.toString());
}
```

Dokument ne odgovara danoj XML Schemi (element *order* mora imati točno jedan podelement *customer*)

```
VALIDATION NOT successful for document with URI file:/C:/JDeveloper/jdev/mywork/TEP/Project1/fileA-1.xml
concerning XML Schema with URI file:/C:/JDeveloper/jdev/mywork/TEP/Project1/schemaB.xsd
ERROR:cvc-complex-type.2.4.d: Invalid content was found starting with element 'customer'.
No child element is expected at this point.
```

XSLT

(Extensible Stylesheet Language Transformations)

Obrada XML dokumenata

- Primarna područje upotrebe XML-a
  1. razmjena sadržaja putem mreže
     - početni format →XML → početni ili neki drugi format
       - baza podataka → XML → druga baza podataka
       - baza podataka → XML → ERP
       - ERP → XML → drugi ERP
     - predložak XML dokumenta određen standardom ili bilateralnim/multilateralnim dogovorom
  2. XML kao posrednik pri transformaciji sadržaja u HTML
     - početni format →XML → (X)HTML
       - baza podataka → XML → (X)HTML
     - moguće je pretvaranje izvesti direktno, bez XML-a, XML postaje nužan ako je potrebno u jedinstveni format uklopiti više različitih izvora

Srž ideje

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<katalog>
  <proizvod>
    <naziv>Mortadela</naziv>
    <proizvodjac>PIK Vrbovec</proizvodjac>
    <drzava>Hrvatska</drzava>
    <cijena jedinica="kg">61.78</cijena>
  </proizvod>
  <proizvod>
    <naziv>Sir Tilzit</naziv>
    <proizvodjac>Vindija</proizvodjac>
    <drzava>Hrvatska</drzava>
    <cijena jedinica="kg">53.99</cijena>
  </proizvod>
  <proizvod>
    <naziv>Ulje suncokretovo</naziv>
    <proizvodjac>IPK Cepin</proizvodjac>
    <drzava>Hrvatska</drzava>
    <cijena jedinica="komad">11.99</cijena>
  </proizvod>
  <proizvod>
    <naziv>Cockta</naziv>
    <proizvodjac>Droga Kolinska</proizvodjac>
    <drzava>Slovenija</drzava>
    <cijena jedinica="komad">13.49</cijena>
  </proizvod>
</katalog>
```

```
<html>
 <body>
 <h2>Proizvodi koje Vam nudimo</h2>
 <table border="1">
   <tr bgcolor="#9acd32">
     <th>Proizvod</th>
     <th>Proizvodjac</th>
   </tr>
   <tr>
     <td>Mortadela</td>
     <td>Pik Vrbovec</td>
   </tr>
   <tr>
     <td>Sir Tilzit</td>
     <td>Vindija</td>
   </tr>
   <tr>
     <td>Ulje suncokretovo</td>
     <td>IPK Cepin</td>
   </tr>
   <tr>
     <td>Cockta</td>
     <td>Droga Kolinska</td>
   </tr>
 </table>
 </body>
</html>
```

Proizvodi koje Vam nudimo

Proizvod	Proizvodjac
Mortadela	Pik Vrbovec
Sir Tilzit	Vindija
Ulje suncokretovo	IPK Cepin
Cockta	Droga Kolinska

Srž ideje

- načiniti transformacijski predložak u posebnoj datoteci
- ako se promijeni sadržaj izvorne datoteke (npr. dodavanje novog proizvoda), potrebno je da se promjena sadržaja odrazi i na izlaznu XML (XHTML) datoteku bez promjene sadžaja samog predloška

Skupina jezika XSL

- XSL (Extensible Stylesheet Language) – skupina jezika za rad s XML dokumentima, razvoj pod okriljem W3C
  - **XSL Transformations (XSLT)**
    - jezik zasnovan na XML-u za transformiranje XML dokumenata
    - XML u XML (XHTML)
  - **XSL Formatting Objects (XSL-FO)**
    - jezik zasnovan na XML-u za specificiranje vizualnog formata (XML) dokumenata
    - XML u PDF/PS
  - **XML Path Language (XPath)**
    - jezik koji NIJE zasnovan na XML-u
    - za dohvat pojedinih dijelova XML dokumenta
    - koristi ga XSLT

XSLT - specifikacija

- 3 verzije: 1.0, 1.1 i 2.0
  - XSL Transformations (XSLT) Version 1.0, W3C Recommendation 16 November 1999 http://www.w3.org/TR/1999/REC-xslt-19991116
  - XSL Transformations (XSLT) Version 1.1, W3C Working Draft 24 August 2001 http://www.w3.org/TR/2001/WD-xslt11-20010824/
  - XSL Transformations (XSLT) Version 2.0, W3C Recommendation 23 January 2007 http://www.w3.org/TR/2007/REC-xslt20-20070123/
- verzija 1.1 ostala *working draft*
- verzija 2.0 jest preporuka, ali je ne podržavaju preglednici Interneta
- u praksi se koristi samo verzija 1.0
  - Mozilla Firefox: uključuje XSLT od verzije 3
  - MS internet Explorer: uključuje XSLT od verzije 6
  - Opera: uključuje XSLT od verzije 9
  - Google Chrome: uključuje XSLT od verzije 1
  - Apple Safari: uključuje XSLT od verzije 3

Povratak na primjer

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<katalog>
  <proizvod>
    <naziv>Mortadela</naziv>
    <proizvodjac>PIK Vrbovec</proizvodjac>
    <drzava>Hrvatska</drzava>
    <cijena jedinica="kg">61.78</cijena>
  </proizvod>
  <proizvod>
    <naziv>Sir Tilzit</naziv>
    <proizvodjac>Vindija</proizvodjac>
    <drzava>Hrvatska</drzava>
    <cijena jedinica="kg">53.99</cijena>
  </proizvod>
  <proizvod>
    <naziv>Ulje suncokretovo</naziv>
    <proizvodjac>IPK Cepin</proizvodjac>
    <drzava>Hrvatska</drzava>
    <cijena jedinica="komad">11.99</cijena>
  </proizvod>
  <proizvod>
    <naziv>Cockta</naziv>
    <proizvodjac>Droga Kolinska</proizvodjac>
    <drzava>Slovenija</drzava>
    <cijena jedinica="komad">13.49</cijena>
  </proizvod>
</katalog>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version=
  "1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
 <html>
 <body>
 <h2>Proizvodi koje Vam nudimo</h2>
 <table border="1">
  <tr bgcolor="#9acd32">
   <th>Proizvod</th>
   <th>Proizvodjac</th>
  </tr>
  <xsl:for-each select="katalog/proizvod">
  <tr>
   <td><xsl:value-of select="naziv"/></td>
   <td><xsl:value-of select="proizvodjac"/></td>
  </tr>
  </xsl:for-each>
 </table>
 </body>
 </html>
</xsl:template>

</xsl:stylesheet>
```

Proizvodi koje Vam nudimo

Proizvod	Proizvodjac
Mortadela	Pik Vrbovec
Sir Tilzit	Vindija
Ulje suncokretovo	IPK Cepin
Cockta	Droga Kolinska

Element *stylesheet*

- svaki XSLT dokument ima korijenski element *stylesheet*
  - sadrži poziv na standardni imenik za XSLT, http://www.w3.org/1999/XSL/Transform
- element *stylesheet* ima 1 atribut
  - *version* (opisuje verziju predloška; vrijednost "1.0")
- element *stylesheet* ima jedan ili više podelemenata *template*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

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

</xsl:stylesheet>
```

Element *template*

- svaki element *template* odnosi se na dio dokumenta određen XPath izrazom koji je vrijednost pripadnog atributa *match*
  - dijete elementa *template* je korijen rezultata transformacije
  - u pravilu, ako element *stylesheet* sadrži samo jedan podelement *template*, on se odnosi na cijeli dokument (match="/")
  - *stylesheet* može sadržavati više podelemenata *template*; u tom slučaju jedna transformacija u sebi uključuje poziv druge transformacije, analogno pozivanju potprograma u glavnom programu

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>Proizvodi koje Vam nudimo</h2>
  ...
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

Transformacija se odnosi na cijeli XML dokument tj. na korijenski element *katalog*

Rezultat transformacije je XML (XHTML) struktura čiji je korijen element *html*

Proizvodi koje Vam nudimo

Element *value-of*

- na njegovom mjestu se u rezultirajućem dokumentu pojavljuje vrijednost **elementa** ili atributa koji je zadan njegovom atributom *select*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
 <html>
 <body>
   <h2>Proizvodi koje Vam nudimo</h2>
   <table border="1">
     <tr bgcolor="#9acd32">
       <th>Proizvod</th>
       <th>Cijena</th>
       <th>Jedinica</th>
     </tr>
     <tr>
       <td><xsl:value-of select="katalog/proizvod/naziv"/></td>
       <td><xsl:value-of select="katalog/proizvod/cijena"/></td>
       <td><xsl:value-of select="katalog/proizvod/cijena/@jedinica"/></td>
     </tr>
   </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

Tablica s tri stupca: Proizvod, Cijena i jedinica

Dohvat naziva i cijene proizvoda (elementi) te odnosi li se cijena na komad ili kg (atribut *jedinica*)

Ispisuju se samo podaci za prve pojave elemenata naziv i cijena odnosno atributa jedinica.

Potreban je izraz koji će omogućiti dohvat SVIH proizvoda

Proizvodi koje Vam nudimo

Proizvod	Cijena	Jedinica
Mortadela	61.78	kg

Element *for-each*

- Omogućuje dohvat svih elemenata ili atributa određenog naziva (naziv je zadan kao vrijednost atributa *select*)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
 <html>
 <body>
   <h2>Proizvodi koje Vam nudimo</h2>
   <table border="1">
     <tr bgcolor="#9acd32">
       <th>Proizvod</th>
       <th>Cijena</th>
       <th>Jedinica</th>
     </tr>
     <xsl:for-each select="katalog/proizvod">
     <tr>
       <td><xsl:value-of select="naziv"/></td>
       <td><xsl:value-of select="cijena"/></td>
       <td><xsl:value-of select="cijena/@jedinica"/></td>
     </tr>
     </xsl:for-each>
   </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

dohvat svih elemenata *proizvod* koji su podelement elementa *katalog*

nakon pojave izraza **for-each**, atribut *select* ugniježđenog elementa **value-of** navodi samo relativni put u odnosu na element zadan u **for-each**

Proizvodi koje Vam nudimo

Proizvod	Cijena	Jedinica
Mortadela	61.78	kg
Sir Tilzit	53.99	kg
Ulje suncokretovo	11.99	komad
Cockta	13.49	komad

Element *sort*

- Omogućuje sortiranje elemenata dohvaćenih nekim drugim izrazom prema vrijednosti podelementa ili atributa

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
 <html>
 <body>
   <h2>Proizvodi koje Vam nudimo</h2>
   <table border="1">
     <tr bgcolor="#9acd32">
       <th>Proizvod</th>
       <th>Cijena</th>
       <th>Jedinica</th>
     </tr>
     <xsl:for-each select="katalog/proizvod">
       <xsl:sort select="naziv" order="ascending"/>
       <tr>
         <td><xsl:value-of select="naziv"/></td>
         <td><xsl:value-of select="cijena"/></td>
         <td><xsl:value-of select="cijena/@jedinica"/></td>
     </tr>
     </xsl:for-each>
   </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

dohvat svih elemenata *proizvod* koji su podelement elementa *katalog*

sortiranje po nazivu (abecedno) uzlazno (relativni put u odnosu na element zadan u **for-each**)

Proizvodi koje Vam nudimo

Proizvod	Cijena	Jedinica
Cockta	13.49	komad
Mortadela	61.78	kg
Sir Tilzit	53.99	kg
Ulje suncokretovo	11.99	komad

Element *if*

- Ispitivanje nekog uvjeta; ne postoji izraz *else*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
 <html>
 <body>
   <h2>Proizvodi koje Vam nudimo</h2>
   <table border="1">
     <tr bgcolor="#9acd32">
       <th>Proizvod</th>
       <th>Cijena</th>
       <th>Jedinica</th>
     </tr>
     <xsl:for-each select="katalog/proizvod">
       <xsl:if test="cijena &gt; 30">
         <tr>
           <td><xsl:value-of select="naziv"/></td>
           <td><xsl:value-of select="cijena"/></td>
           <td><xsl:value-of select="cijena/@jedinica"/></td>
         </tr>
       </xsl:if>
     </xsl:for-each>
   </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

Uzeti u obzir samo proizvoda čija cijena je VEĆA od 30 (> je u XML-u > < je <)

Proizvodi koje Vam nudimo

Proizvod	Cijena	Jedinica
Mortadela	61.78	kg
Sir Tilzit	53.99	kg

Element *choose*

- naredba analogna **switch-case-default** u drugim programskim jezicima
- *when* se može pojaviti više puta unutar *choose* (analogno **case**)

```
<xsl:choose>
  <xsl:when test="expression">
    ... some output ...
  </xsl:when>
  <xsl:otherwise>
    ... some output ....
  </xsl:otherwise>
</xsl:choose>
```

Više predložaka

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>Proizvodi koje Vam nudimo</h2>
  <xsl:apply-templates/>
  </body>
  </html>
</xsl:template>

<xsl:template match="proizvod">
  <p>
  <xsl:apply-templates select="naziv"/>
  <xsl:apply-templates select="proizvodjac"/>
  </p>
</xsl:template>

<xsl:template match="naziv">
  Proizvod: <span style="color:#ff0000">
  <xsl:value-of select="."/></span>
  <br />
</xsl:template>

<xsl:template match="proizvodjac">
  Proizvodjac: <span style="color:#00ff00">
  <xsl:value-of select="."/></span>
  <br />
</xsl:template>

</xsl:stylesheet>
```

Primjeni predložak koji se odnosi na korijenski element (***katalog***) ILI za svaki njegov podelement (***proizvod***) pozovi predložak (**template**) koji u atributu **match** navodi taj podelement; konkretno poziva se *xsl:template match="proizvod"*

Za svaki proizvod otvori novi odlomak ( ). Potom pozovi najprije predložak za element ***naziv*** (*xsl:template match="naziv"*), onda predložak za element ***proizvodjac*** (*xsl:template match="proizvodjac"*) i na kraju zatvori odlomak ( ).

Proizvodi koje Vam nudimo

Proizvod: Mortadela
Proizvodjac: PIK Vrbovec

Proizvod: Sir Tilzit
Proizvodjac: Vindija

Proizvod: Ulje suncokretovo
Proizvodjac: IPK Cepin

Proizvod: Cockta
Proizvodjac: Droga Kolinska

XSLT i preglednik Interneta

- U XML datoteku dodajemo jednu procesorsku instrukciju kojom se pozivamo na predložak
- Preglednik ne prikazuje XML datoteku, nego rezultat XSL transformacije

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="trans4.xsl"?>
<katalog>
  <proizvod>
    <naziv>Mortadela</naziv>
    <proizvodjac>PIK Vrbovec</proizvodjac>
    <drzava>Hrvatska</drzava>
    <cijena jedinica="kg">61.78</cijena>
  </proizvod>
      ...
</katalog>
```

Proizvodi koje Vam nudimo

Proizvod	Cijena	Jedinica
Mortadela	61.78	kg
Sir Tilzit	53.99	kg
Ulje suncokretovo	11.99	komad
Cockta	13.49	komad

Programska izvedba

- preporučuje se programski paket **Xalan** tvrtke *Apache* (otvoreni kod, javno dostupan, ugrađen u mnogim sustavima za razvoj programa)

Primjer 4 – Transformiranje XML datoteke korištenjem XSLT predloška

```
import javax.xml.transform.TransformerFactory; import javax.xml.transform.Transformer; import javax.xml.transform.Templates;
import javax.xml.transform.TransformerConfigurationException; import javax.xml.trans
import javax.xml.transform.stream.StreamResult; import javax.xml.transform.stream.S

public class XSLTTransformation {

   public static void transform(String xmlInputUri, String xmlOutputUri, String transformationUri) {

      try{
        java.net.URI transUri=new java.net.URI(transformationUri);
        java.net.URL transUrl=transUri.toURL();
        StreamSource transSource=new StreamSource(transUrl.toString());
        java.net.URI inputUri=new java.net.URI(xmlInputUri);
        java.net.URL inputUrl=inputUri.toURL();
        StreamSource inputSource=new StreamSource(inputUrl.toString());
        java.net.URI outputUri=new java.net.URI(xmlOutputUri);
        java.net.URL outputUrl=outputUri.toURL();
        StreamResult outputResult=new StreamResult(outputUrl.toString());

        TransformerFactory factory=TransformerFactory.newInstance();
        Templates templates=factory.newTemplates(transSource);
        Transformer transformer=templates.newTransformer();
        transformer.transform(inputSource, outputResult);
      } catch(java.net.URISyntaxException use){
        use.printStackTrace();
      } catch(java.net.MalformedURLException mue){
        mue.printStackTrace();
      } catch(TransformerConfigurationException tce){
        tce.printStackTrace();
      } catch(TransformerException te){
        te.printStackTrace();
      }

   }

   // izostavljena je metoda public static void main
```

ulazni argumenti su URI ulaznog XML-a, URI izlaznog (transformiranog) dokumenta i URI XSLT predloška

ulazni XML i XSLT predložak treba proslijediti kao objekt klase *javax.xml.transform.stream.StreamSource*; adresu izlaznog (transformiranog) dokumenta treba proslijediti kao objekt klase *javax.xml.transform.stream.StreamResult*; u oba slučaja trebamo URL (ne URI)

Najprije se stvara objekt apstraktne klase *javax.xml.transform.TransformerFactory*; on učitava predložak i pritom stvara objekt klase *javax.xml.transform.Templates*; iz njega se stvara objekt klase *javax.xml.transform.Transformer*; Transformer vrši transformaciju uz proslijeđivanje adrese ulaznog i izlaznog dokumenta

Apache: implementacija apst. klasa *TransformerFactory, Transformer* i *Templates* paketa *javax.xml.transform* su *com.sun.org.apache.xalan.internal.xsltc.trax.TransformerFactoryImpl*, *com.sun.org.apache.xalan.internal.xsltc.trax.TransformerImpl* i *com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl*

Primjer 4

```
public static void main(String[] args){
    String inputPath="C:/Documents and Settings/mbanek.ZOEM/My Documents/Nastava/TEP/xslt/katalog.xml";
    java.io.File inputFile=new java.io.File(inputPath);
    java.net.URI inputUri=inputFile.toURI();

    String stylesheetPath="C:/Documents and Settings/mbanek.ZOEM/My Documents/Nastava/TEP/xslt/trans1.xsl";
    java.io.File styleshhetFile=new java.io.File(stylesheetPath);
    java.net.URI stylesheetUri=styleshhetFile.toURI();

    String outputPath="C:/Documents and Settings/mbanek.ZOEM/My Documents/Nastava/TEP/xslt/izlaz.html";
    java.io.File outputFile=new java.io.File(outputPath);
    java.net.URI outputUri=outputFile.toURI();

    transform(inputUri.toString(), outputUri.toString(), stylesheetUri.toString());
}
```