Simple API for XML - SAX

Parsiranje XML dokumenata

- svako bi mogao da napravi parser za XML dokumente, ali...
- bolje da koristimo gotove parsere
  - testirani
    - vođeno je računa o svim detaljima XML specifikacije
  - optimizovani
  - obavljaju i validaciju

Parsiranje XML dokumenata

- postoji više kvalitetnih parsera za XML
- koji koristiti?
  - vezivanje za neki specifičan parser nije poželjno
    - šta ako razvoj tog parsera prestane?
- standardni API za XML parsere
  - možemo koristiti svaki parser koji poštuje ovaj API
  - ako zamenimo parser ne menjamo naš kod koji ga koristi
- → Simple API for XML (SAX)

Simple API for XML

- SAX je definisan za različite jezike
  - Java
  - C++
  - ...
- specifikacije SAX-a za svaki jezik su međusobno vrlo slične

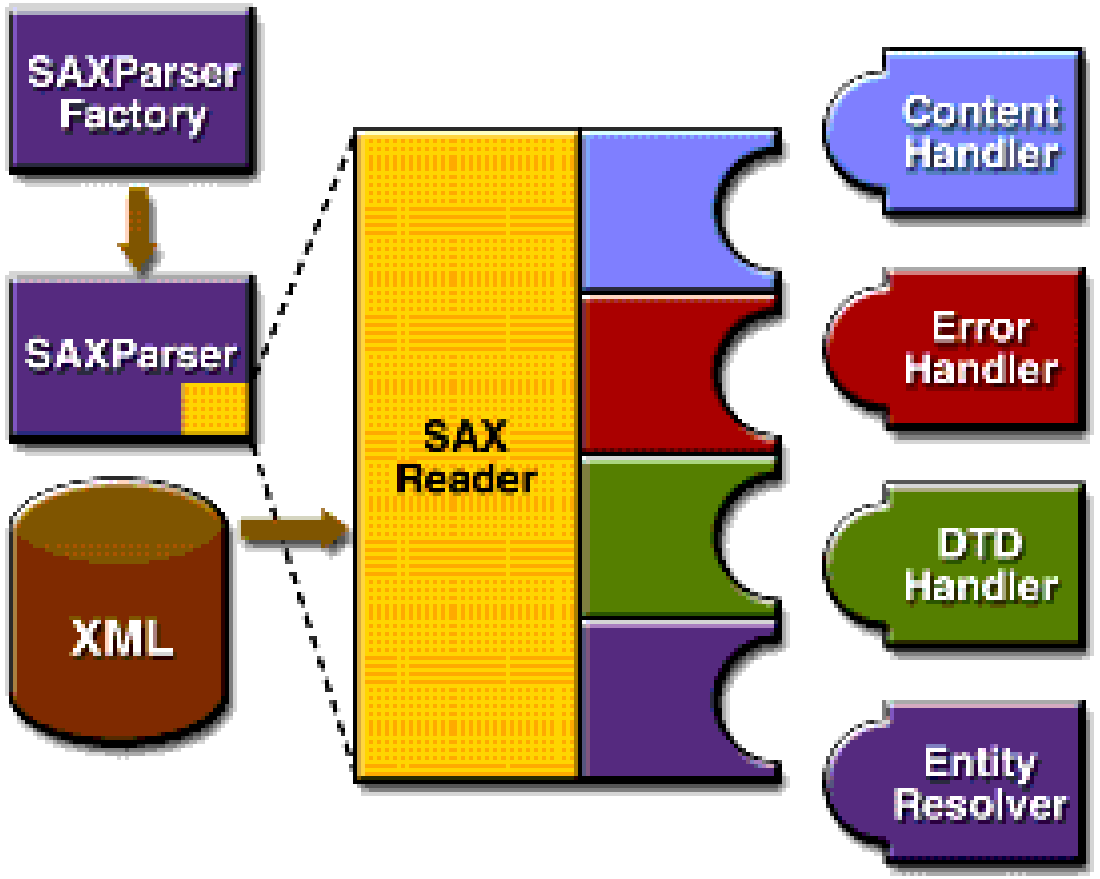
Simple API for XML

- nije W3C standard
  - razvijen kroz saradnju na xml-dev mailing listi
  - jeste de facto standard
- verzije
  - 1998: v 1.0
  - 2000: v 2.0: namespace podrška, property mehanizam

SAX koncept

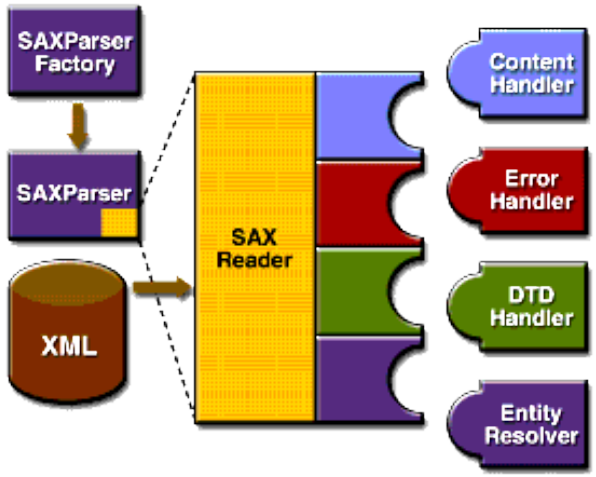
- parsiranje pomoću SAX-a je *event-driven*
- parser tokom parsiranja „generiše događaje“
  - npr. počeo dokument, počeo element, završio se element, ...
- naš kôd je zadužen da obradi „događaj“
  - pišemo tzv. *handlere*
  - njih poziva parser (*callback*)

SAX koncept


SAXParser Factory
SAXParser
XML
SAX Reader
Content Handler
Error Handler
DTD Handler
Entity Resolver

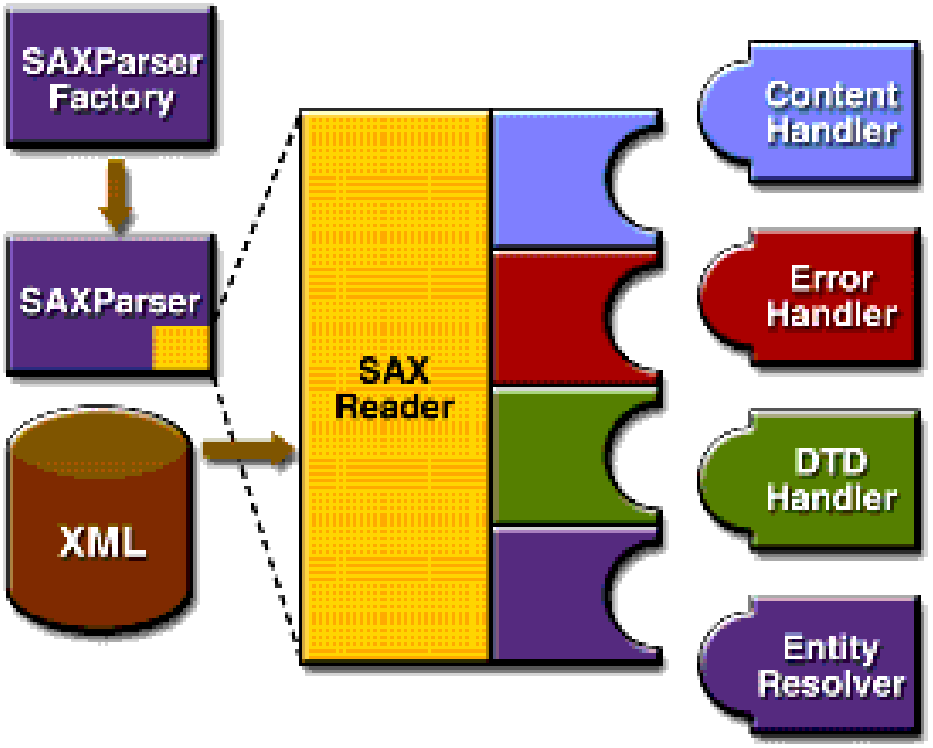
SAX API

- SAXParserFactory – kreira instance parsera
  - određene sistemskim promenjivama
- SAXParser – interfejs sa nekoliko parse() metoda
  - tipični parametri: izvor XML podataka i DefaultHandler objekat obrađuje događaje
- SAXReader – nalazi se unutar SAXParser-a
  - ukoliko je potrebno da se preciznije konfiguriše pozivamo metod getXMLReader(). SAXReader poziva event handlere



SAX handleri

- ContentHandler - interfejs sa *callback* metodama za obradu događaja vezanih za sadržaj dokumenta
  - startDocument(), endDocument(), startElement(), endElement(), characters(), processingInstructions()



SAX handleri

- ErrorHandler - obrada događaja vezanih za greške tokom parsiranja
  - error(), fatalError(), warning()
- DTDHandler - obrada događaja vezanih za parsiranje DTD-a
  - notationDecl(), unparsedEntityDecl()
- EntityResolver - pribavljanje eksternih entiteta
  - resolveEntity()

SAX handleri

- da bismo upotrebili parser trebalo bi implementirati 4 interfejsa!
  - naporno
  - nisu nam potrebne sve funkcije (DTD, eksterni entiteti...)
- pomoćna klasa: DefaultHandler
  - implementira sva 4 interfejsa praznim metodama
  - dovoljno je naslediti DefaultHandler i redefinisati samo one *callback* metode koje su nam stvarno potrebne

SAX API i Java

- javax.xml.parsers
  - SAXParserFactory
  - SAXParser
- org.xml.sax
  - ContentHandler
  - ErrorHandler
  - DTDHandler
  - EntityResolver
- org.xml.sax.helpers
  - DefaultHandler

Echo.java

- napravićemo program koji čita XML dokument i ispisuje ga na konzolu
- klasa Echo nasledиće DefaultHandler
  - kao reakciju na svaki događaj ispisaćemo ono što nam parser isporuči
- dodaćemo main()
  - inicijalizacija parsera
  - pokretanje parsera

Echo.java

- main() - inicijalizacija i pokretanje parsera

```
//instanca nase klase - event handler
DefaultHandler handler = new Echo();

// factory kreira parsere
SAXParserFactory factory = SAXParserFactory.newInstance();

// instanciramo jedan parser
SAXParser saxParser = factory.newSAXParser();

// pokrenemo parsiranje
saxParser.parse(new File(args[0]), handler);
```

Echo.java

- dve utility metode
  - nl() - ispisuje znak za novi red u skladu sa operativnim sistemom

```
private void nl() throws SAXException {
  String lineEnd = System.getProperty("line.separator");
  try {
    out.write(lineEnd);
  } catch (IOException e) {
    throw new SAXException("I/O error", e);
  }
}
```

Echo.java

- dve utility metode
  - emit() - ispisuje string na konzolu

```
private void emit(String s) throws SAXException {
  try {
    out.write(s);
    out.flush();
  } catch (IOException e) {
    throw new SAXException("I/O error", e);
  }
}
```

startDocument() i endDocument()

```
public void startDocument() throws SAXException {
  emit("<?xml version='1.0' encoding='UTF-8'?>");
  nl();
}


public void endDocument() throws SAXException {
  try {
    nl();
    out.flush();
  } catch (IOException e) {
    throw new SAXException("I/O error", e);
  }
}
```

startElement()

```
public void startElement(String namespaceURI,
                         String sName,         ...bez prefiksa
                         String qName,         ...sa prefiksom
                         Attributes attrs)
    throws SAXException {

  if ("".equals(sName))
    sName = qName;
  emit("<"+sName);
  if (attrs != null) {
    for (int i = 0; i < attrs.getLength(); i++) {
      String aName = attrs.getLocalName(i);
      if ("".equals(aName))
        aName = attrs.getQName(i);
      emit(" ");
      emit(aName+"=\""+attrs.getValue(i)+"\"");
    }
  }
  emit(">");
}
```

endElement()

```
public void endElement(String namespaceURI,
                       String sName,
                       String qName)
    throws SAXException {

  if ("".equals(sName))
    sName = qName;
  emit("</"+sName+">");
}
```

characters()

```
StringBuffer textBuffer;

public void characters(char buf[], int offset, int len) throws SAXException {
  String s = new String(buf, offset, len);
  if (textBuffer == null) {
    textBuffer = new StringBuffer(s);
  } else {
    textBuffer.append(s);
  }
}

private void echoText() throws SAXException {
  if (textBuffer == null)
    return;
  emit(s.toString());
  textBuffer = null;
}
```

characters() može biti pozvana više puta za jedan kontinualni tekstualni sadržaj iz dokumenta!

characters()

- ispisujemo sadržaj svaki put kad počinje i završava element

```
public void startElement(...) throws SAXException {
  echoText();
  ...
}

public void endElement(...) throws SAXException {
  echoText();
  ...
}
```

→primer1

Analiza rezultata

- očuvani whitespace
  - parser nema DTD na raspolaganju - pretpostavlja da svaki element ima mixed content model - pa čuva sve whitespace karaktere u sadržaju elemenata
  - nije očuvan između atributa!
- komentari su ignorisani
  - trebalo bi implementirati LexicalHandler
- prazni elementi
  - <item/> se događajima predstavlja kao <item></item>

Whitespace test

- dodajemo ispis opisa svakog događaja

→primer2

ignorableWhitespace()

- služi da parser „javi“ postojanje nebitnog whitespace sadržaja
  - (parser mora da zna koji whitespace je nebitan, treba mu DTD)

setDocumentLocator(Locator loc)

- Locator - objekat koji sadrži podatke o lokaciji na kojoj se desio događaj
  - Locator je validan samo u trenucima poziva event-handling metoda

processingInstruction()

- parametri
  - target - aplikacija koja treba da procesira instrukciju
  - data - podaci za obradu

```
public void processingInstruction(String target, String data)
    throws SAXException {
  nl();
  emit("PROCESS: ");
  emit("<?"+target+" "+data+"?>");
}
```

→primer3

Obrada grešaka

- SAXException
  - može da sadrži i izuzetak koji se desio u event-handleru
- SAXParseException
  - nasleđuje SAXException
  - sadrži informacije o redu u kome je greška

Obrada grešaka

- tri nivoa grešaka
  - warning(SAXParseException e)
    - npr. element definisan dva puta u DTD-u (jeste greška, ali ne pravi probleme)
  - error(SAXParseException e)
    - npr. dokument nije validan
  - fatalError(SAXParseException e)
    - npr. dokument nije dobro formiran

→primer4

CDATA sekcije

- reference na entitete parser automatski zamenjuje njihovim vrednostima
- CDATA sekcije parser automatski pretvara u nizove znakova
  - za Echo primer trebalo bi još zameniti znakove &, <, > referencama na entitete & < > u characters()

Parsiranje uz validaciju

- dokument može da poseduje svoj DTD ili referencu na spoljašnji DTD
- ako DTD postoji
  - nevalidirajući parser ignoriše whitespace tamo gde je to moguće
    - ako nam ipak trebaju, koristimo ignorableWhitespace() događaj
  - validirajući radi sve to plus validaciju

Parsiranje uz validaciju

- kreiranje parsera
  - izbor fabrike - pomoću sistemskog property-ja javax.xml.parsers.SAXParserFactory=com.foo.MyFactory
  - da li je parser validirajući factory.setValidating(true)
  - da li parser vodi računa o namespace-ovima factory.setNamespaceAware(true)

→primer5

Validacija pomoću šeme

1) napraviti validirajući parser koji radi sa namespaces
   - factory.setValidating(true)
   - factory.setNamespaceAware(true)

2) definisati koji šema-jezik se koristi tako što se postavi vrednost za sledeći property parsera (mi koristimo XML Schema)
   - property se zove http://java.sun.com/xml/jaxp/properties/schemaLanguage
   - a vrednost mu je http://www.w3.org/2001/XMLSchema
   -
```
saxParser.setProperty(
  "http://java.sun.com/xml/jaxp/properties/schemaLanguage",
  "http://www.w3.org/2001/XMLSchema");
```

ime propertija je u URL formatu

Validacija pomoću šeme

- povezivanje dokumenta sa šemom - na dva načina
  - šema deklaracija u dokumentu
  - programsko povezivanje (bezbednije...), tada se šema deklaracije u dokumentu ne uzimaju u obzir

```
saxParser.setProperty(
  "http://java.sun.com/xml/jaxp/properties/schemaSource",
  new File(schemaSource));
```

Upozorenja

- prilikom rada sa DTD-om
  - duplirane definicije u DTD-u
  - korišćenje elemenata koji nisu definisani u DTD-u
  - deklarisanje atributa za nepostojeće elemente
- prilikom rada parsera
  - parser je validirajući, a dokument nema <!DOCTYPE ...>
  - refenciranje nedefinisanih parametarskih entiteta kada nema validacije (kada ima validacije, to je greška)

Obrada leksičkih događaja

- opciona
- implementiramo LexicalHandler
  - comment()
  - startCDATA()
  - endCDATA()
  - startEntity()
  - endEntity()
  - startDTD()
  - endDTD()
- naglasimo parseru da se koristi i LexicalHandler

```
xmlReader.setProperty(
  "http://xml.org/sax/properties/lexical-handler",
  handler);
```

→primer6

Obrada DTD događaja

- kada se naiđe na neparsirani entitet

```
<!ENTITY myEntity SYSTEM "URL..." NDATA gif>
<!NOTATION gif SYSTEM "URL...">
```

- za obradu ovih događaja: DTDHandler
  - unparsedEntityDecl()
  - notationDecl()

EntityResolver

- za pribavljanje entiteta
  resolveEntity(String publicId, String systemId)

→primer7