

Tehnologii Web

I GUOIO SU AAGD

Procesari XML

partea II: *SAX (Simple API for XML)* +
prelucrarea simplificata a documentelor XML

detalii in [TX, 240-268]



“Inainte de a pune noi intrebari,
gindeste-te daca intr-adevar vrei
sa cunosti raspunsul la ele.”

Gene Wolfe

©6U6 MOIIG



Exista maniere alternative
pentru procesarea documentelor XML?



sax: intro

sax: intro

Scop:
manipularea documentelor XML
fara ca in prealabil sa fie construit
arborele de noduri-obiect



sax: intro

sax: intro

Scop:

manipularea documentelor XML
fara ca in prealabil sa fie construit
arborele de noduri-obiect

▶ documentul nu trebuie stocat complet
in memorie inainte de a fi efectiv prelucrat



sax: caracterizare

29X: C9L9Cf6L1Z9L6

Ofera o procesare XML secventiala (liniara),
bazata pe evenimente – *event-oriented*

initiator: David Megginson
www.megginson.com/SAX/



sax: caracterizare

29X: C9L9Cf6L1Z9L6

Efort independent – de cel al Consorțiului Web –
de standardizare a procesării XML
condusă de evenimente

larg acceptat ca standard industrial:
SAX 1.0 + **SAX 2.0** (spatii de nume + extensii)

www.saxproject.org



sax: procesare

29X: b10C62916

Pentru fiecare tip de constructie XML
– inceput de *tag*, sfarsit de *tag*, continut,
instructiune de procesare, comentariu,... –
va fi emis un eveniment care va fi tratat
de o functie/metoda (*handler*)



sax: procesare
29X: b10C62916

Funcțiile/metodele de tratare se specifica
de catre programator, pentru fiecare tip
de constructie in parte



sax: procesare
29X: b10C62916

Programul consuma si trateaza evenimente
produse de procesorul SAX



sax: procesare

29X: b10C62916

Minimal, trebuie definite functiile/metodele:

trateaza_tag_inceput (*procesor, tag, atrib*)

trateaza_tag_sfarsit (*procesor, tag*)

trateaza_date_caract (*procesor, date*)



sax: procesare

29X: b10C62916

Minimal, trebuie definite functiile/metodele:

trateaza_tag_inceput (*procesor, tag, atrib*)

trateaza_tag_sfarsit (*procesor, tag*)

trateaza_date_caract (*procesor, date*)

contine lista atributelor
atasate *tag*-ului de inceput



sax: procesare

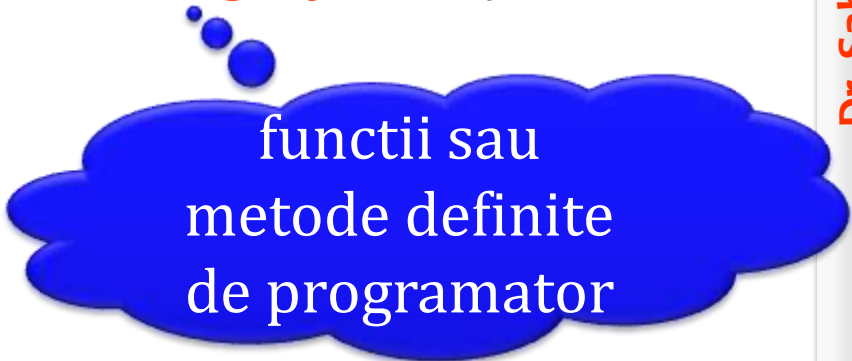
Pentru fiecare eveniment de aparitie a *tag*-ului de inceput, a *tag*-ului de sfarsit si a datelor-continut, se ataseaza una din functiile de tratare, respectiv:

set_element_handler

(*trateaza_tag_inceput, trateaza_tag_sfarsit*)

set_character_data_handler

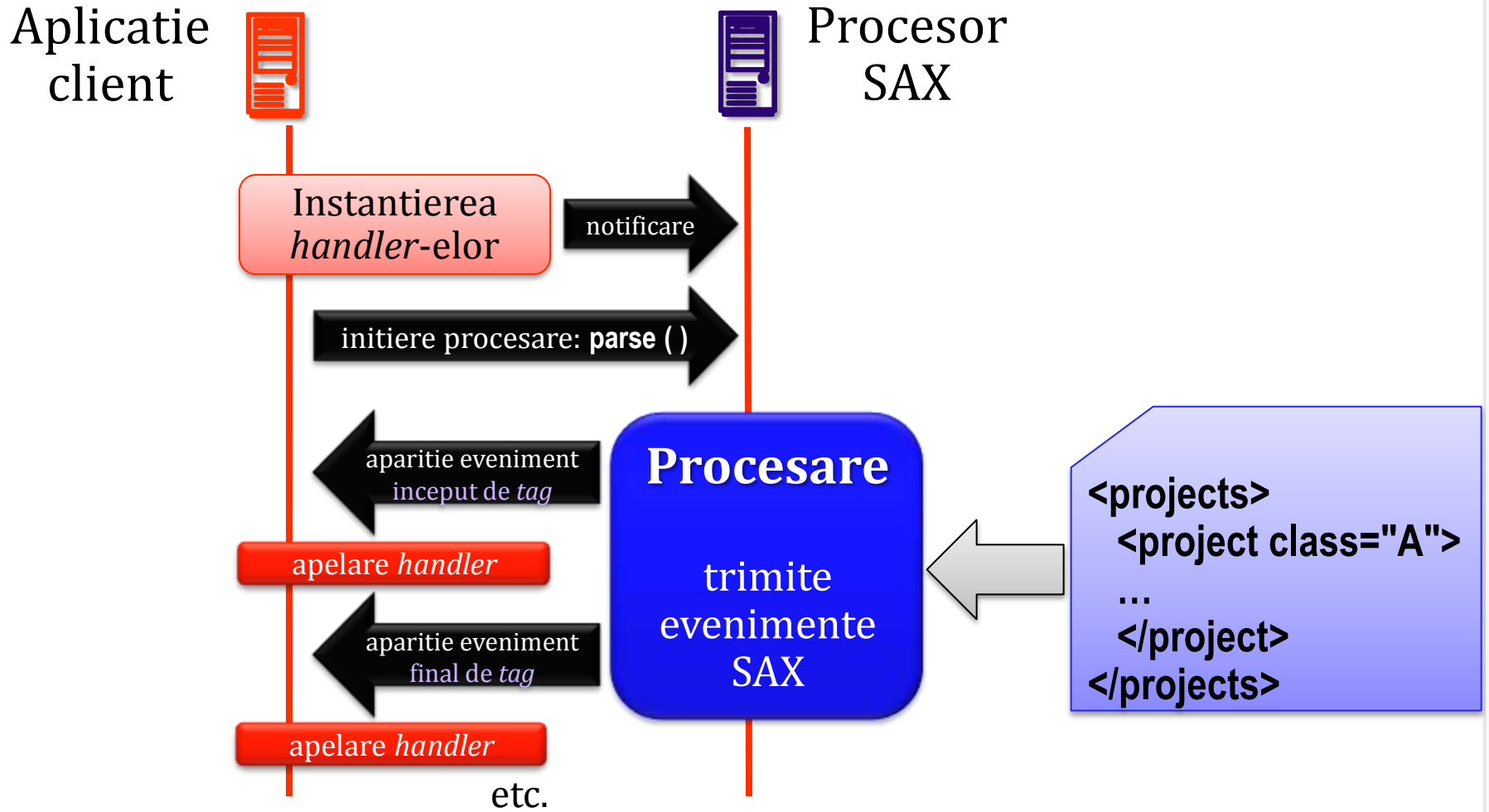
(*trateaza_date_caract*)



functii sau
metode definite
de programator



sax: procesare





sax: procesare

Implementarea de referinta (Java): **org.xml.sax**
interfete implementate de procesorul XML (*SAX Driver*)
interfete implementate de aplicatie: **DocumentHandler**,
ErrorHandler, **DTDHandler**, **EntityResolver** (optionale)
clase SAX standard: **InputSource**, **SAXException**,
SAXParseException, **HandlerBase**
clase aditionale:
ParserFactory, **AttributeListImpl**, **LocatorImpl**

Exemplificare: interfata `XMLReader` (Apache Xerces)

// realizarea procesarea XML (consultarea datelor)

```
public interface XMLReader {  
    // furnizeaza informatii despre document  
    public ContentHandler getContentHandler ();  
    public DTDHandler getDTDHandler ();  
    public EntityResolver getEntityResolver ();  
    public ErrorHandler getErrorHandler ();  
    // seteaza diverse functionalitati  
    public void setContentHandler (ContentHandler contentHandler);  
    public void setDTDHandler (DTDHandler dtdHandler);  
    public void setEntityResolver (EntityResolver resolver);  
    public void setErrorHandler (ErrorHandler errHandler);  
    // procesarea propriu-zisa  
    public void parse (InputSource in)  
        throws java.io.IOException, SAXException;  
    public void parse (String uri)  
        throws java.io.IOException, SAXException;  
}
```

Exemplificare: interfata `ContentHandler` (Apache Xerces)

// utilizata pentru manipularea continuturilor XML

```
public interface ContentHandler {  
  public void setDocumentLocator (Locator locator);  
  public void startDocument () throws SAXException;  
  public void endDocument () throws SAXException;  
  // evenimente  
  public void startElement (String uri, String localName, String qName,  
    Attributes attributes) throws SAXException;  
  public void endElement (String uri, String localName, String qName)  
    throws SAXException;  
  public void characters (char buf[], int offset, int length)  
    throws SAXException;  
  // informatii suplimentare  
  public void ignorableWhitespace (char buf[], int offset, int length)  
    throws SAXException;  
  public void startPrefixMapping (String prefix, String uri)  
    throws SAXException;  
  public void endPrefixMapping (String prefix)  
    throws SAXException;  
}
```

Exemplificare: interfata `Attributes` (Apache Xerces)

```
// specifica attributele asociate unui element
public interface Attributes {
    public int getLength ();
    public String getType (int index);
    public String getValue (int index);
    // acces la informatiile privitoare la nume
    public String getQName (int index);
    public String getLocalName (int index);
    public String getURI (int index);
    // acces via spatii de nume XML
    public int getIndex (String uri, String localName);
    public String getType (String uri, String localName);
    public String getValue (String uri, String localName);
    // acces via nume calificate (ns:nume)
    public int getIndex (String qName);
    public String getType (String qName);
    public String getValue (String qName);
}
```



sax: implementari

libxml – API *open source*: C, C++, Perl, Python, Ruby,...

MSSAX – procesari SAX in C/C++, JScript;
inclus in MSXML SDK (*Software Development Kit*)

org.xml.sax – API de referinta pentru Java

REXML – procesor XML pentru Ruby



sax: implementari

QSAX – parte a mediului de dezvoltare Qt (C++)

Xerces SAX API – platforma XML pentru C++ si Java:
<http://xml.apache.org/>

XML::Parser – modul Perl bazat pe procesorul Expat

xml_*() – functii PHP

xml.sax – modul Python, parte a PyXML



sax: demo
29X: q6W0





sax vs. dom

sax vs. dom

Cand trebuie folosit SAX?

procesarea unor documente de mari dimensiuni

necesitatea abandonarii procesarii
(procesorul SAX poate fi oprit oricind)

extragerea unor informatii de mici dimensiuni



sax vs. dom

sax vs. dom

Cand trebuie folosit SAX?

crearea unei structuri noi de document XML

utilizarea in contextul unor resurse de calcul reduse
(memorie scazuta, largime de banda ingusta,...)

context: **Web-ul mobil** – vezi suportul pentru **Android**:

<http://developer.android.com/reference/javax/xml/parsers/SAXParser.html>



sax vs. dom

Cand trebuie folosit DOM?

accesul direct la datele dintr-un document XML

cautari complexe

filtrarea complexa a datelor via XPath

efectuarea de transformari XSL



sax vs. dom

Cand trebuie folosit DOM?

necesitatea modificarii & salvarii documentelor XML

in contextul procesarii datelor XML direct
in cadrul navigatorului Web, date obtinute eventual
via transferuri asincrone prin AJAX



sax vs. dom

sax vs. dom

DOM necesita incarcarea completa
a documentului XML
in vederea procesarii ca arbore

SAX necesita pentru procesare existenta
unor fragmente reduse din document,
efectuandu-se o prelucrare liniara
(sir de evenimente)



sax vs. dom

SAX poate fi utilizat
pentru generarea de arbori DOM

Invers, arborii DOM pot fi traversati
pentru a se emite evenimente SAX



sax vs. dom

sax vs. dom

In cazul unor structuri XML sofisticate,
modul de procesare SAX poate fi inadecvat

procesarile SAX ignora
contextul aparitiei unui anumit element



sax vs. dom: exemplificare

Fie structura de document XML,
specificata prin urmatorul DTD:

```
<!DOCTYPE catalog [  
  <!ELEMENT catalog      (categ+)>  
  <!ELEMENT categ        (#PCDATA | categ)*>  
>
```

Ce metoda de procesare s-ar preta, daca numarul de
elemente **<categ>** ar fi de ordinul milioanelelor?



sax vs. dom

Unele implementari SAX ofera suport
pentru validari si transformari

uzual, se folosesc ambele API-uri



Exista si alte metode de procesare XML?



Procesarea documentelor XML

alternative:

XPP – XML Pull Parsing
“legarea” datelor XML
procesare simplificata



alternative

Stiluri de procesari XML conduse de evenimente:

push versus pull



alternative

Stiluri de procesari XML conduse de evenimente:

push = procesorul XML citește date XML și notifică aplicația asupra evenimentelor survenite
(*parsing events*) – **SAX**



alternative

Stiluri de procesari XML conduse de evenimente:

push = procesorul XML citeste date XML si notifica aplicatia asupra evenimentelor survenite
(*parsing events*) – **SAX**

aplicatia nu poate face cereri de evenimente;
ele apar asa cum sunt trimise (*push*) de procesor



alternative

Stiluri de procesari XML conduse de evenimente:

pull = aplicatia controleaza maniera de procesare si poate solicita (*pull*) procesorului urmatorul eveniment XML

XPP – XML Pull Parsing



alternative

Stiluri de procesari XML conduse de evenimente:

pull = aplicatia controleaza maniera de procesare si poate solicita (*pull*) procesorului urmatorul eveniment XML

XPP – XML Pull Parsing

structura codului-sursa al aplicatiei
reflecta structura documentului XML prelucrat

alternative

Interfetele <i>push</i>	Interfetele <i>pull</i>
Procesare <i>read-only</i>	Mostenesc avantajele interfetelor <i>push</i>
Prelucrare rapida, via fluxuri de date (<i>streams</i>)	Evenimentele se consuma conform necesitatilor
Codul-sursa poate fi dificil de inteles	Programele au o structura mai clara

alternative: implementari

alternative: implementari

StAX – *Streaming API for XML* (Java) – JSR 173

<http://jcp.org/en/jsr/detail?id=173>

exemple de implementari:

BEA StAX

Javolution – focalizat pe performanta: <http://javolution.org/>

Oracle StAX – inclus in XDK

SJSXP – disponibil in Java SDK

Woodstox – oferit in contextul SOAP



alternative: implementari

XML Pull – API facilitand procesarile in stilul *pull*
www.xmlpull.org

implementare Java – **org.xmlpull.***

asigura inter-operabilitatea
cu manierele de procesare DOM si SAX



alternative

Clasificare a manierelor de procesare XML

mod de accesare: **secvential** vs. **direct** (*random*)

controlul fluxului: ***pull*** vs. ***push***

managementul arborelui: **ierarhic** vs. **imbricat**



alternative

DOM ofera acces direct, in stilul *pull*

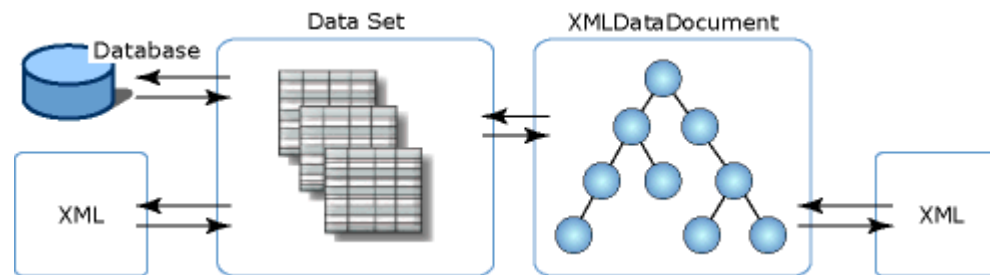
SAX ofera acces secvential, in stilul *push*

XPP si **.NET XmlTextReader** ofera
acces secvential, in stilul *pull*

alternative

“Legarea” datelor XML de alte surse de date
(*XML binding*)

baze de date: *XML info set* ↔ *dataset*





alternative

alternative

“Legarea” datelor XML de alte surse de date
(*XML binding*)

abordare obiectuala:

date XML ↔ clase create “din zbor” (C#, Java, Perl, PHP5)



alternative

“Legarea” datelor XML de alte surse de date
(*XML binding*)

interogari asupra datelor XML
direct in limbajul de programare

LINQ (*Language INtegrated Query*) – .NET Framework



alternative

“Legarea” datelor XML de alte surse de date
(*XML binding*)

exemple de implementari:

Castor (Java)

JAXB – *Java Architecture for XML Binding*

XmlDataDocument (.NET)

Zeus (Java)

etc.



alternative
alternative

Procesarea XML simplificata

scop:

procesarea unui document XML (de mici dimensiuni)
direct in memorie,
in maniera obiectuala,
diferita de DOM



alternative
alternative

Procesarea XML simplificata

fiecarui element XML ii poate corespunde
o proprietate a unui obiect

atributele asociate elementelor XML pot fi memorate
intr-o structura de date – *e.g.*, tablou asociativ



alternative

Procesarea XML simplificata

exemple de implementari:

E4X – *ECMAScript for XML* (JavaScript, ActionScript)

libxml (C, C++ si alte limbaje)

SimpleXML (PHP)

XML::Simple + **XML::Writer** (Perl)

XmlTextReader + **XmlTextWriter** (.NET)



alternative

Procesarea XML simplificata
pentru consultare, se poate folosit
un cititor (*reader*): *XMLReader*

e.g., clasa **XmlTextReader** oferita de .NET Framework



alternative

Procesarea XML simplificata
pentru generare, se poate utiliza
un scriitor (*writer*): *XMLWriter*

e.g., clasa **XmlTextWriter** oferita de .NET Framework



alternative

Procesarea XML simplificata

exemplu recurgind la limbajul PHP

```
$xml = simplexml_load_file ('http://sit.info/sxml/projects.xml');  
// afisam descrierile proiectelor de clasa A via expresii XPath  
foreach ($xml->xpath ("//project[@class='A']") as $proiecte) {  
    echo '<p>' . $proiecte->desc . '</p>';  
}
```



alternative: comparatii

Masurarea performantei procesarilor XML
(*benchmarking*)

exemplu:

XML Benchmark

<http://xmlbench.sourceforge.net/>

A photograph of a forest floor. In the foreground, there are several pine branches with long, thin, brown needles. To the right, there are several logs and pieces of wood, some of which are charred or broken. A green fern is visible in the upper left quadrant. The ground is covered with dry leaves and small twigs. The text "rezumat" is overlaid in white in the upper center, and "procesarea documentelor XML de la SAX la Simple XML & altele" is overlaid in white and red in the lower center.

rezumat

procesarea documentelor XML
de la **SAX** la **Simple XML** & altele

