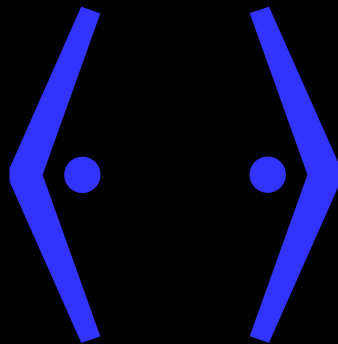


Tehnologii Web

procesarea datelor XML (II)



SAX (Simple API for XML)

prelucrarea simplificată a documentelor XML

“Înainte de a pune noi întrebări,
gândește-te dacă într-adevăr vrei
să cunoști răspunsul la ele.”

Gene Wolfe

Există maniere alternative
pentru procesarea documentelor XML?

sax: intro

Scop:

consultarea documentelor XML/HTML
fără ca în prealabil să fie construit
arborele de noduri-obiect

sax: intro

Scop:

consultarea documentelor XML/HTML
fără ca în prealabil să fie construit
arborele de noduri-obiect

► documentul nu trebuie stocat complet
în memorie înainte de a fi efectiv prelucrat

sax: caracterizare

Oferă o procesare XML secvențială (liniară),
bazată pe evenimente – *event-oriented*

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

sax: caracterizare

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

www.saxproject.org

sax: caracterizare

Larg acceptat ca standard industrial

SAX 1.0 (1998)

implementare de referință în limbajul Java
org.xml.sax

sax: caracterizare

Larg acceptat ca standard industrial

SAX 2.0 (2002—2004)

suport pentru spații de nume,
diverse configurări + extensii

sax: procesare

- Pentru fiecare tip de construcție XML
- început de *tag*, sfârșit de *tag*, date (conținut), instrucțiune de procesare, comentariu,... – va fi emis un eveniment care va fi tratat de o funcție/metodă (*handler*)

sax: procesare

Funcțiile/metodele de tratare se specifică
de către programator, pentru fiecare tip
de construcție în parte

sax: procesare

Programul consumă și tratează evenimente produse de procesorul SAX

sax: procesare

Minimal, pentru SAX 1.0,
trebuie definite funcțiile/metodele:

trateaza_tag_inceput (*procesor, tag, atrib*)

trateaza_tag_sfarsit (*procesor, tag*)

trateaza_date_caract (*procesor, date*)

sax: procesare

Minimal, pentru SAX 1.0,
trebuie definite funcțiile/metodele:

trateaza_tag_inceput (*procesor*, *tag*, *atrib*)

trateaza_tag_sfarsit (*procesor*, *tag*)

trateaza_date_caract (*procesor*, *date*)

conține lista atributelor
atașate *tag*-ului de început

sax: procesare


Pentru fiecare eveniment de apariție a *tag*-ului de început, a *tag*-ului de sfârșit și a datelor-conținut, se atașează una din funcțiile de tratare, respectiv:

set_element_handler

(*trateaza_tag_inceput*, *trateaza_tag_sfarsit*)

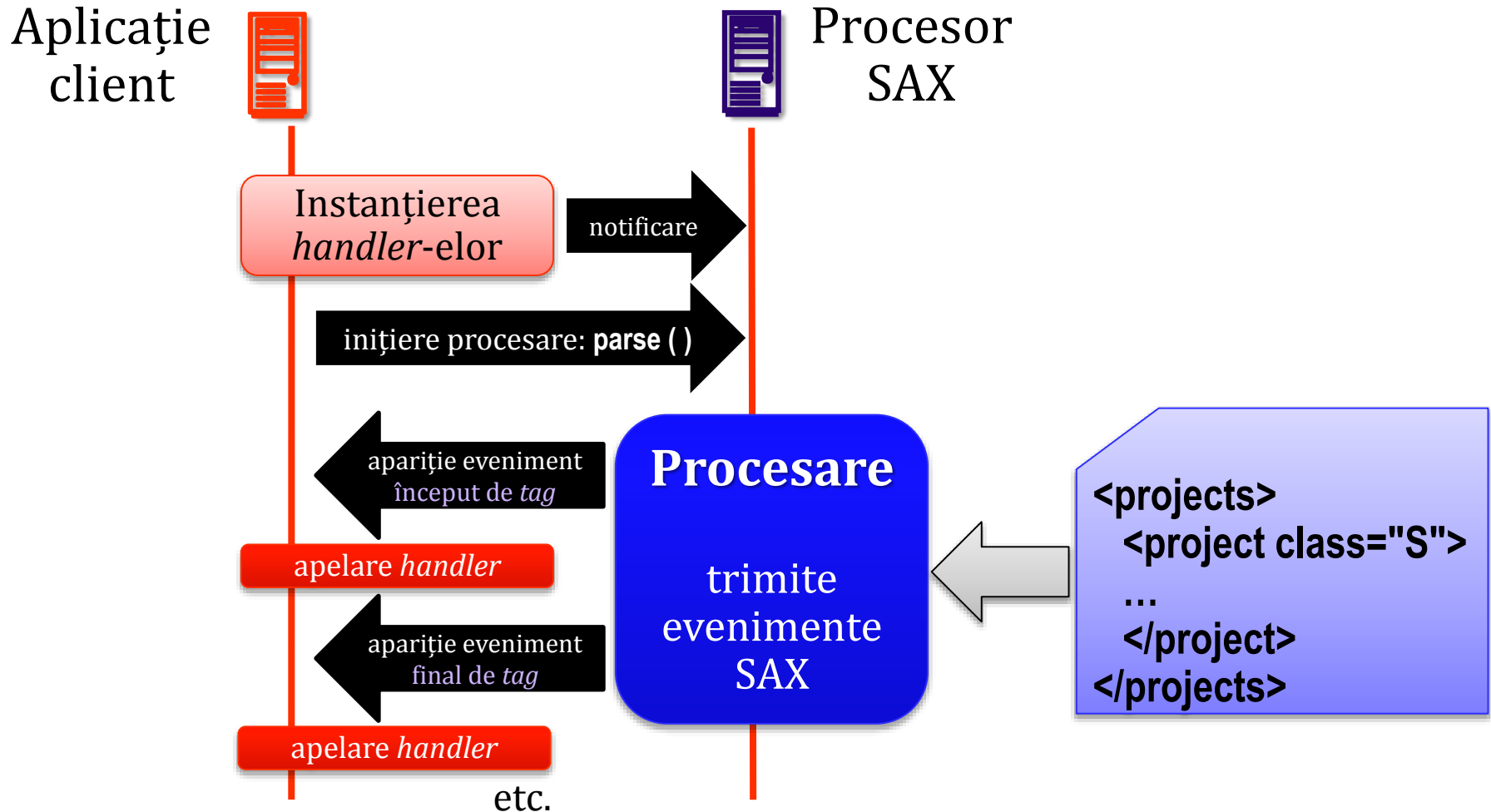
set_character_data_handler

(*trateaza_date_caract*)



funcții sau
metode definite
de programator

sax: procesare



sax: procesare

Implementarea de referință (Java): **org.xml.sax**
interfețe implementate de procesorul XML (*SAX Driver*)
interfețe implementate de aplicație: **DocumentHandler**,
ErrorHandler, **DTDHandler**, **EntityResolver** (opționale)
clase SAX standard: **InputSource**, **SAXException**,
SAXParseException, **HandlerBase**
clase adiționale:
ParserFactory, **AttributeListImpl**, **LocatorImpl**

Exemplificare: interfața `XMLReader` (Apache Xerces)

// prelucrare XML via evenimente (consultarea datelor)

```
public interface XMLReader {  
    // furnizarea de informații despre document  
    public ContentHandler getContentHandler ();  
    public DTDHandler getDTDHandler ();  
    public EntityResolver getEntityResolver ();  
    public ErrorHandler getErrorHandler ();  
    // stabilirea diverselor funcționalități  
    public void setContentHandler (ContentHandler contentHandler);  
    public void setDTDHandler (DTDHandler dtdHandler);  
    public void setEntityResolver (EntityResolver resolver);  
    public void setErrorHandler (ErrorHandler errHandler);  
    // procesarea propriu-zisă  
    public void parse (InputSource in)  
        throws java.io.IOException, SAXException;  
    public void parse (String uri)  
        throws java.io.IOException, SAXException;  
}
```

Exemplificare: interfața **ContentHandler** (Apache Xerces)

```
// utilizată pentru manipularea conținuturilor 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;
    // informații 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: interfața **Attributes** (Apache Xerces)

// specifică attributele asociate unui element

```
public interface Attributes {  
    public int getLength ();  
    public String getType (int index);  
    public String getValue (int index);  
    // acces la informațiile privitoare la numele atributului  
    public String getQName (int index);  
    public String getLocalName (int index);  
    public String getURI (int index);  
    // acces via spații 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: implementări

libxml – API *open source*: C, C++, Haskell, Scala,...

MSSAX – procesări SAX în C, C++, JavaScript;
inclus în MSXML SDK (*Software Development Kit*)

NSXMLParser – implementare Objective-C (Apple)

org.xml.sax – API de referință pentru Java

REXML – procesor XML pentru Ruby

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

sax: implementări

sax – modul Node.js

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

XML::Parser – modul Perl bazat pe procesorul Expat

xml_*() – funcții PHP

xml.sax – suită de module Python

sax: demo



sax vs. dom

Când trebuie folosit SAX?

procesarea unor documente de mari dimensiuni

necesitatea abandonării procesării
(procesorul SAX poate fi oprit oricând)

extragerea unor informații de mici dimensiuni

sax vs. dom

Când trebuie folosit SAX?

crearea unei structuri noi de document XML

utilizarea în contextul unor resurse de calcul reduse
(memorie scăzută, lărgime de bandă îngustă,...)

exemplificare pentru **Android**:

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

sax vs. dom

Când trebuie utilizat DOM?

accesul direct la datele dintr-un document XML

procesări sofisticate

filtrarea complexă a datelor via XPath

efectuarea de transformări XSL

validarea datelor XML prin DTD, XML Schema etc.

sax vs. dom

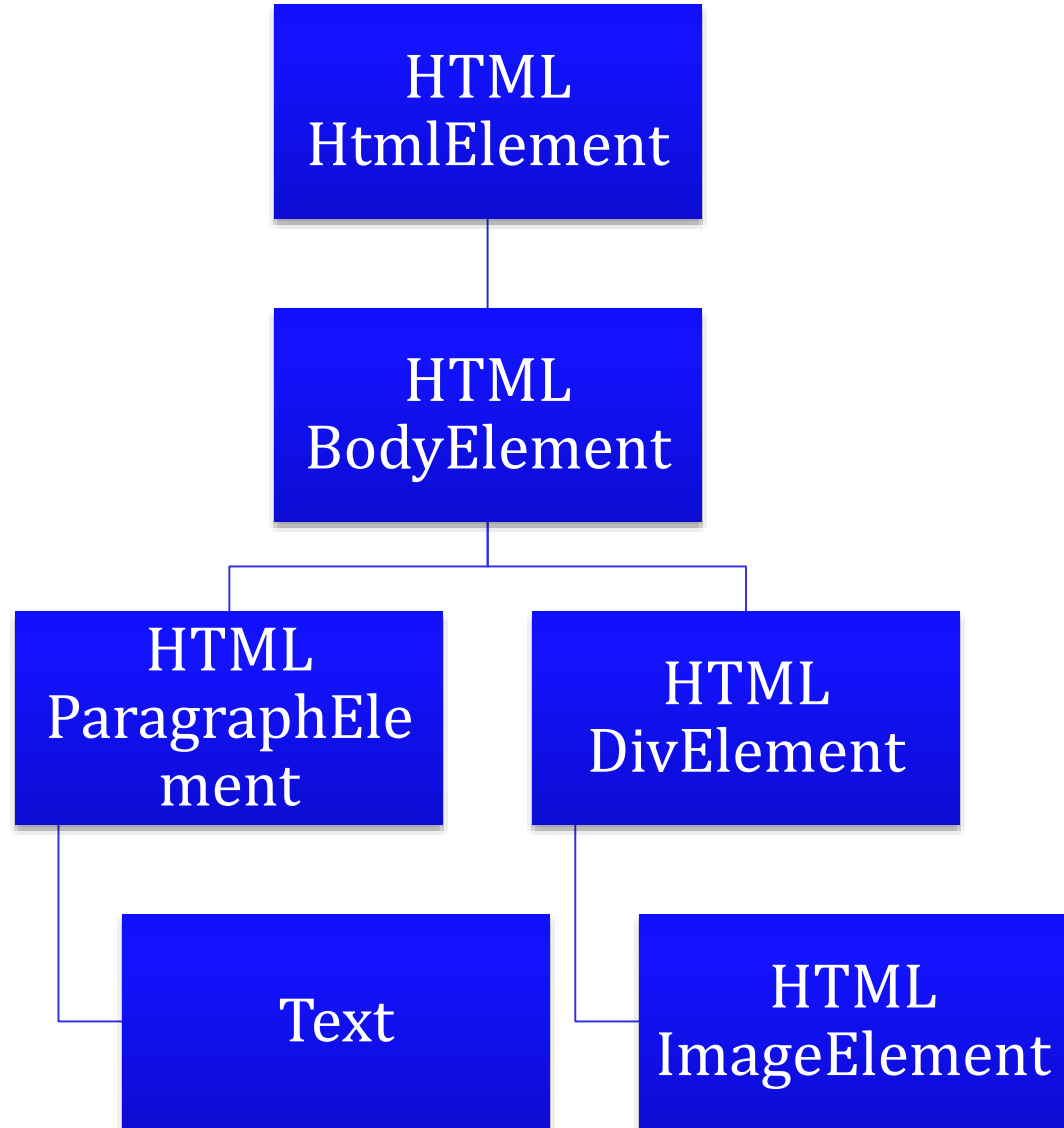
Când trebuie utilizat DOM?

necesitatea modificării și/sau salvării documentelor XML

în contextul procesării datelor XML/HTML direct
în cadrul navigatorului Web, date obținute eventual
via transferuri asincrone prin AJAX

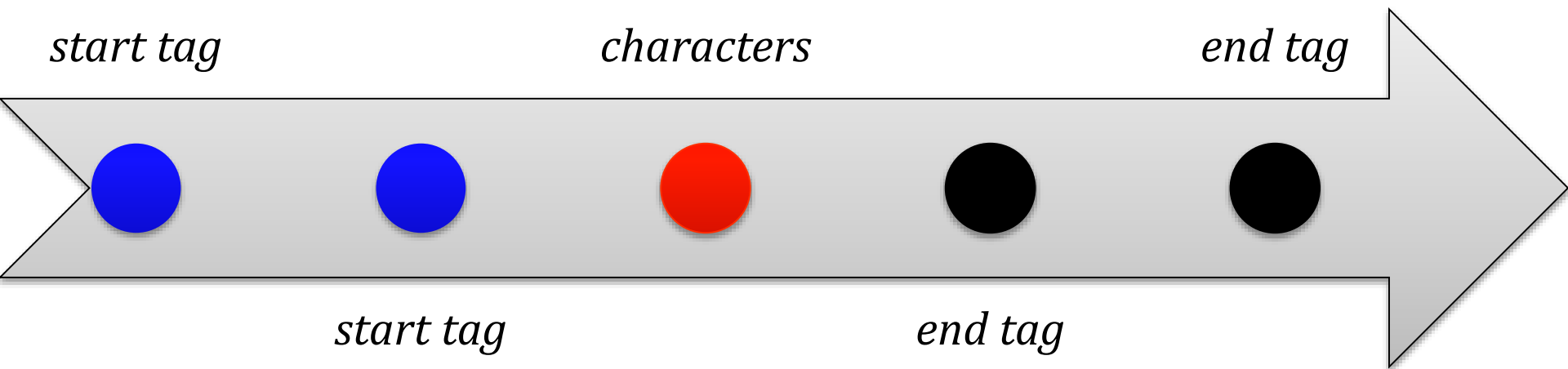
sax vs. dom

DOM necesită
încărcarea completă
a documentului XML
în vederea procesării
ca arbore



sax vs. dom

SAX preia fragmente reduse din document,
efectuându-se o prelucrare liniară
(șir de evenimente)



sax vs. dom

SAX poate fi utilizat
pentru generarea de arbori DOM

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

exemplificări:

modulul **dom-js** (Node.js), biblioteca **lxml** (Python)

sax vs. dom

În cazul unor structuri XML sofisticate,
maniera de procesare SAX poate fi inadecvată

procesările SAX ignoră
contextul apariției unui anumit element

sax vs. dom: exemplificare

Fie structura de document XML,
specificată prin următorul DTD:

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

Ce metodă de procesare s-ar preta, dacă numărul de
elemente **<categ>** ar fi de ordinul milioane?

sax vs. dom

Unele implementări SAX oferă suport
pentru validări și transformări

uzual, se folosesc ambele API-uri

Există și alte metode de procesare XML?

Procesarea documentelor XML

alternative:

XPP – *XML Pull Parsing*

„legarea” datelor XML
procesare simplificată

alternative: xml pull parsing

Stiluri de procesări XML conduse de evenimente:

push versus pull

alternative: xml pull parsing

Stiluri de procesări XML conduse de evenimente:

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

alternative: xml pull parsing

Stiluri de procesări XML conduse de evenimente:

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

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

alternative: xml pull parsing

Stiluri de procesări XML conduse de evenimente:

pull = aplicația controlează maniera de procesare și poate solicita (*pull*) procesorului următorul eveniment XML

XPP – XML Pull Parsing

alternative: xml pull parsing

Stiluri de procesări XML conduse de evenimente:

pull = aplicația controlează maniera de procesare și poate solicita (*pull*) procesorului următorul eveniment XML

XPP – XML Pull Parsing

structura codului-sursă al programului
reflectă structura documentului XML prelucrat

alternative: xml pull parsing

Interfețele <i>push</i>	Interfețele <i>pull</i>
Procesare <i>read-only</i>	Moștenesc avantajele interfețelor <i>push</i>
Prelucrare rapidă, via fluxuri de date (<i>streams</i>)	Evenimentele se consumă conform necesităților
Codul-sursă poate fi dificil de înțeles	Programele au o structură mai clară

xml pull parsing – implementări

StAX – *Streaming API for XML* (Java) – JSR 173
<http://jcp.org/en/jsr/detail?id=173>

exemple de implementări:

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

Oracle StAX – inclus în XDK

SJSXP – disponibil în Java SDK

Woodstox – oferit în contextul serviciilor Web cu SOAP

xml pull parsing – implementări

irrXML – inițial, parte din Irrlicht 3D Engine (C++)

pull – pachet Scala pe procesare XPP

QXmlStreamReader, QXmlStreamWriter din mediul **Qt** (C++)

saxpath – modul Node.js permițând evaluarea de expresii XPath pentru un flux de evenimente SAX

XmlPullParser – interfață Java pentru Android

alternative

Clasificare a manierelor de procesare XML

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

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

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

alternative

DOM ofera acces direct, în stilul *pull*

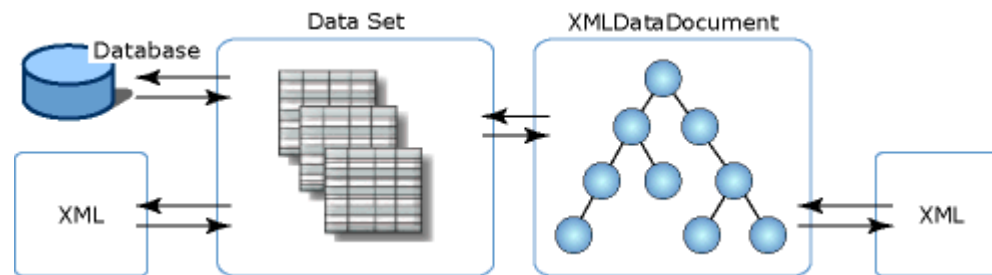
SAX oferă acces secvențial, în stilul *push*

XPP și **.NET XmlTextReader** oferă
acces secvențial, în stilul *pull*

alternative

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

baze de date: *XML info set* \leftrightarrow *dataset*



alternative

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

abordare obiectuală:

date XML \leftrightarrow clase create „din zbor” (C#, Java, Perl, PHP)

alternative

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

interogări asupra datelor XML
direct în limbajul de programare

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

<http://msdn.microsoft.com/en-us/library/vstudio/bb397926.aspx>

XDocument proiecte; // XDocument e o clasă .NET

avansat

```
proiecte = XDocument.Load ("projects.xml");
```

```
var proiecteS =
```

```
    // via o expresie LINQ, preluăm toate proiectele
```

```
    from p in proiecte.Descendants ("project")
```

```
    // din care le alegem pe cele de clasa 'S'
```

```
    where (String) p.Attribute ("class") == "S"
```

```
    // ordonate după numărul de studenți
```

```
    orderby (String) p.Element ("stud")
```

```
    // selectând doar titlul acestora
```

```
    select (String) p.Element ("title");
```

```
// afișăm titlul proiectelor de clasa 'S'
```

```
foreach (var proiect in proiecteS) {
```

```
    Console.WriteLine (proiect);
```

```
}
```

```
// același rezultat, recurgând la XPath
```

```
var proiecteS2 = (IEnumerable)
```

```
    proiecte.XPathEvaluate ("//project[@class='S']/title");
```

alternative

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

JAXB – *Java Architecture for XML Binding* (JSR-222)
<https://jcp.org/en/jsr/detail?id=222>

implementarea de referință: <https://jaxb.java.net/>

diverse *framework*-uri Java: Castor, Zeus

alternative

Procesarea XML simplificată

scop:

procesarea unui document XML (de mici dimensiuni)
direct în memorie,
în manieră obiectuală,
diferită de DOM

alternative

Procesarea XML simplificată

uzual, adoptă maniera de prelucrare XPP
(*XML Pull Parsing*)

alternative

Procesarea XML simplificată

fiecărui element XML îi poate corespunde
o proprietate a unui obiect

atributele asociate elementelor XML pot fi modelate
via o structură de date – *e.g.*, tablou asociativ

alternative

Procesarea XML simplificată

exemplificări diverse:

E4X – ECMAScript for XML (JavaScript)

libxml (C, C++ si alte limbaje)

SimpleXML (PHP)

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

XmlTextReader + XmlTextWriter (.NET)

alternative

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

exemple:

modulul **xmlreader** pentru Node.js

xmlReader oferit de biblioteca libxml (implementare C)

extensia **XMLReader** pentru PHP

clasa **XmlTextReader** oferită de .NET Framework

alternative

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

exemplificări:

clasa **XmlTextWriter** disponibilă în .NET Framework
xmlWriter din cadrul bibliotecii libxml (C)
extensia **XMLWriter** pentru PHP
modulul **xml-writer** oferit pentru Node.js


```
$xml = @simplexml_load_file ('http://sit.info/atom.xml');
echo '<header><h1>Însemnările de pe <em>blog</em>-ul lui ' .
$xml->author->name . '</h1></header>';
echo '<article><ol>';
// baleiăm însemnările (elementele <entry>)
foreach ($xml->entry as $insemnare) {
    echo '<li><a title="Detalii" href="' . $insemnare->link['href'] . '">'.
    htmlspecialchars ($insemnare->title) . '</a>';
    // dacă există <content type="html">...</content>,
    // atunci afișăm conținutul (date marcate în HTML)
    if ($insemnare->content['type'] == 'html') {
        echo '<section class="stire">' . $insemnare->content . '</section>';
    }
    echo '</li>';
}
echo '</ol></article>';
```

recurgerea la maniera de
procesare simplificată în PHP

Cum pot fi procesate documentele HTML?

procesare html

Aspect de interes:
ignorarea erorilor de sintaxă

documente bine formatate vs. documente valide

procesare html

Aspect de interes:
ignorarea erorilor de sintaxă

malformed markup

sunt relativ rare cazurile în care documentele HTML
sunt scrise/generate corect

procesare html

Tehnica folosită în mod comun – nerecomandată

Web scrapping

extragerea datelor de interes
prin prelucrarea – de obicei, empirică –
a marcajelor HTML

procesare html

Recurgerea la un procesor HTML/XML specific

scopuri importante:

traversarea (procesarea) unei pagini Web – *e.g.*, via DOM

+

detectarea & repararea erorilor sintactice (*HTML clean*)

vezi și cursurile
anterioare

procesare html: instrumente

Beautiful Soup – bibliotecă Python

<http://www.crummy.com/software/BeautifulSoup/>

Gumbo – procesor HTML5 în C oferit de Google

<https://github.com/google/gumbo-parser>

html5lib – procesare + serializare HTML pentru Python

<https://github.com/html5lib>

HTML::Gumbo, HTML::Parser – module Perl

<http://search.cpan.org/dist/HTML-Parser/>

procesare html: instrumente

Html Agility Pack – bibliotecă .NET

<http://htmlagilitypack.codeplex.com/>

Hubbub – prelucrare de marcaje HTML5 în limbajul C

<http://www.netsurf-browser.org/projects/hubbub/>

Jericho HTML Parser – bibliotecă Java de procesare HTML

<http://jericho.htmlparser.net/>

jsoup – bibliotecă Java pentru HTML5

<http://jsoup.org/>

procesare html: instrumente

Masterminds HTML5-PHP – procesor HTML5 în PHP
<http://masterminds.github.io/html5-php/>

Nokogiri – pachet Ruby
<http://www.nokogiri.org/>

Parse5 – modul Node.js
<https://github.com/inikulin/parse5>

Pure JavaScript HTML5 Parser – bibliotecă JS
<https://github.com/blowsie/Pure-JavaScript-HTML5-Parser>

procesare html: instrumente

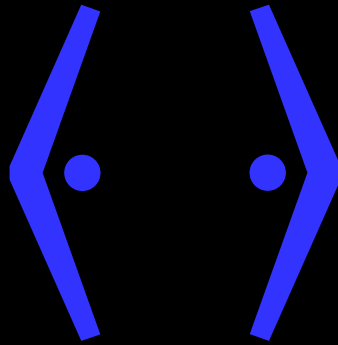
HtmlCleaner – instrument implementat în Java pentru corectarea marcajelor HTML eronate

HTML Purifier – verificare & filtrare a marcajelor HTML (inclusiv vizând atacuri de tip XSS – *Cross Site Scripting*) cu implementări în PHP și Objective-C

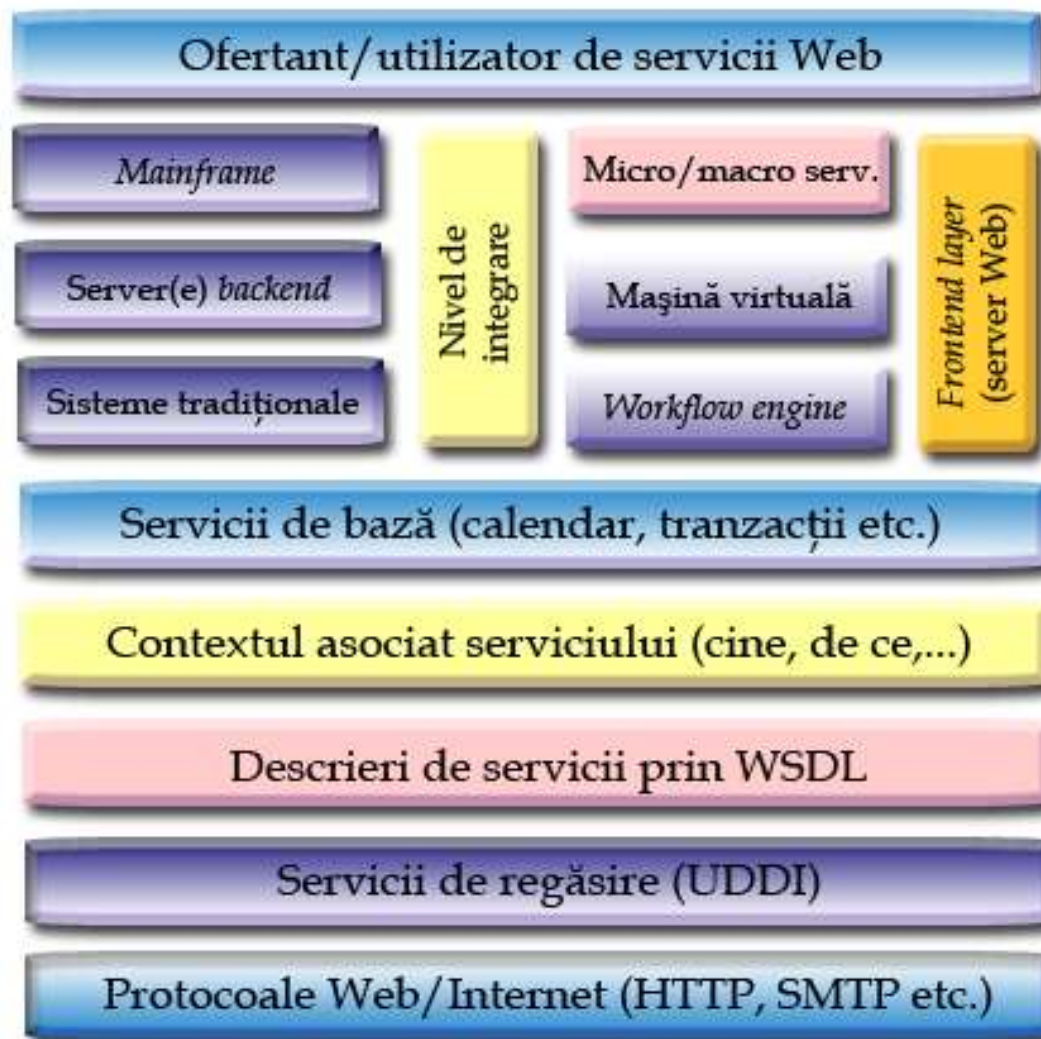
NekoHTML – procesor Java pentru HTML bazat pe Xerces cu suport pentru rezolvarea erorilor sintactice

Validator.nu – procesor Java folosind DOM ori SAX cu semnalarea erorilor de sintaxă HTML

rezumat



procesări XML: de la **SAX** la **XPP** și **Simple XML**
instrumente de prelucrare a documentelor HTML



episodul viitor: **servicii Web** prin SOAP