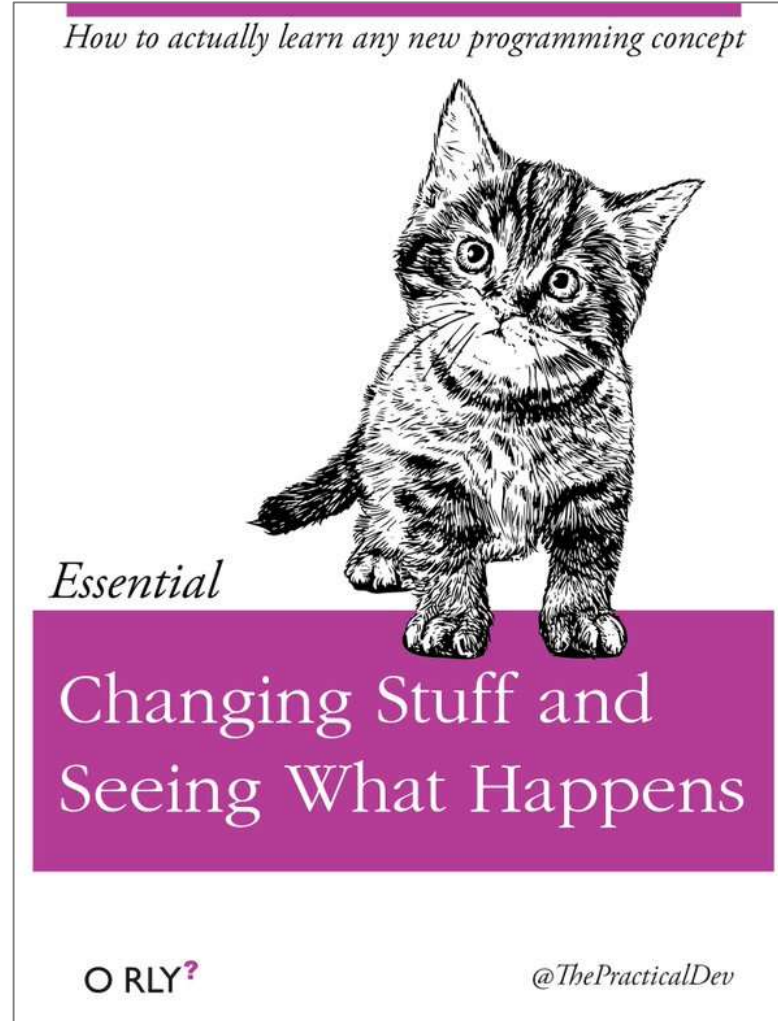


Tehnologii Web



procesarea datelor XML/HTML (II)
SAX, SimpleXML, prelucrarea documentelor HTML

„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/downloads/SAX/

sax: caracterizare

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

“SAX is a **streaming interface** – applications receive information from XML documents in a continuous stream, with no backtracking or navigation allowed”

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 (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 XML în parte

sax: procesare

Programul consumă și tratează evenimente
produse de procesorul SAX

sax: procesare

Minimal, trebuie definite funcțiile/metodele:

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

trateaza_tag_sfarsit (*procesor*, *tag*)

trateaza_date_caracter (*procesor*, *date*)

conține lista atributelor
atașate *tag*-ului de început

sax: procesare

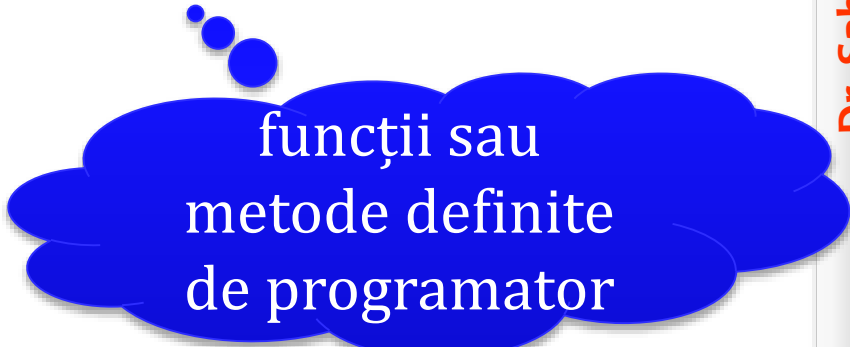
Pentru fiecare eveniment de apariție a *tag*-ului de început, a *tag*-ului de sfârșit și a datelor-conținut, se atașează una din funcțiile de tratare, respectiv:

set_element_handler

(*trateaza_tag_inceput*, *trateaza_tag_sfarsit*)

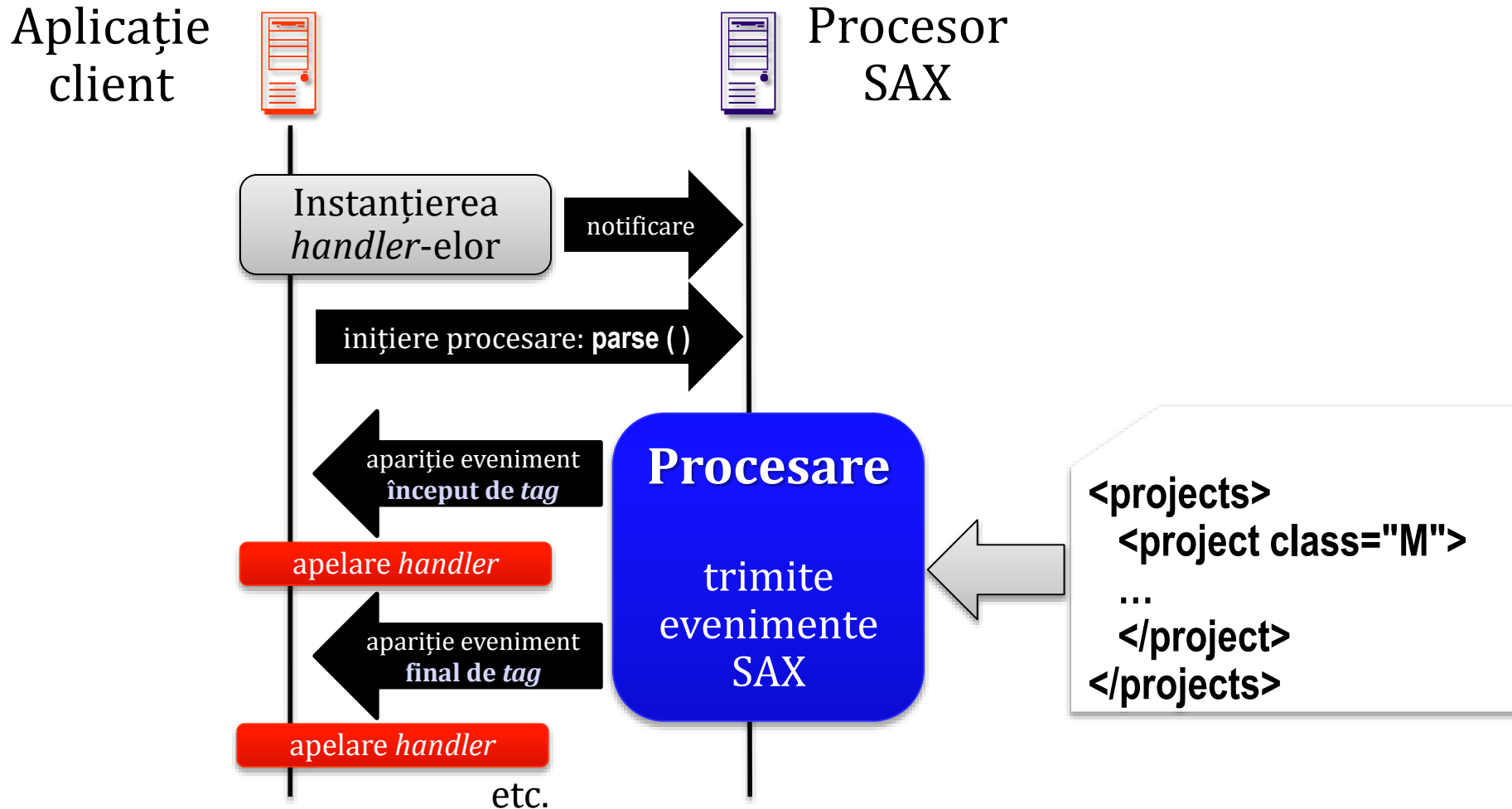
set_character_data_handler

(*trateaza_date_caracter*)



funcții sau
metode definite
de programator

sax: procesare



sax: procesare

Implementarea de referință (Java): **org.xml.sax**

detalii la

www.saxproject.org/apidoc/org/xml/sax/package-summary.html

sax: procesare

Interfețe ce pot fi implementate de aplicația noastră:

ContentHandler

rezolvă notificări de evenimente
vizând tipul construcțiilor XML:

început de document, *tag* de început, date textuale,
tag de sfârșit, sfârșit de document etc.

sax: procesare

Interfețe ce pot fi implementate de aplicația noastră:

Attributes

conține lista atributelor
specificate în cadrul unui *tag* de început

sax: procesare

Interfețe ce pot fi implementate de aplicația noastră:

XMLReader

specifică maniera de citire a datelor XML folosind metode de tratare a evenimentelor (*callback-uri*)

sax: procesare

Interfețe ce pot fi implementate de aplicația noastră:

ErrorHandler

specifică maniera de tratare
a erorilor (fatale) și avertismentelor

pot fi emise excepții precum
SAXException și SAXParseException

sax: procesare

Clasa SAX oferită:

InputSource

încapsulează informații despre o sursă de intrare de unde se preiau datele XML (*e.g.*, flux de caractere)

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 procesarea construcțiilor 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 XML
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 (prefix:nume)
    public int getIndex (String qName);
    public String getType (String qName);
    public String getValue (String qName);
}
```

sax: implementări

Expat – procesor XML cu suport pentru diverse metode de prelucrare (DOM, SAX etc.): libexpat.github.io
(C, C++, Lua, .NET, Objective-C, Perl, Python, Ruby, Tcl)

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

lxml – bibliotecă Python: launchpad.net/lxml

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

sax: implementări

NSXMLParser – implementare Objective-C + Swift (Apple)
developer.apple.com/documentation/foundation/xmlparser

org.xml.sax – API de referință pentru Java

REXML – procesor XML pentru Ruby

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

sax-js – modul Node.js

sax: implementări

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

erlsom, xmerl_eventp – module Erlang

xml – pachet Go: golang.org/pkg/encoding/xml/

XML::Parser – modul Perl bazat pe procesorul Expat

xml_*() – funcții PHP: php.net/manual/en/book.xml.php

xml.sax – Python: docs.python.org/3/library/xml.sax.html


sax: implementări – exemplificare

Procesarea în PHP unui document XML via SAX pentru a genera o reprezentare HTML a datelor

web-test.xml – document XML modelând date de intrare

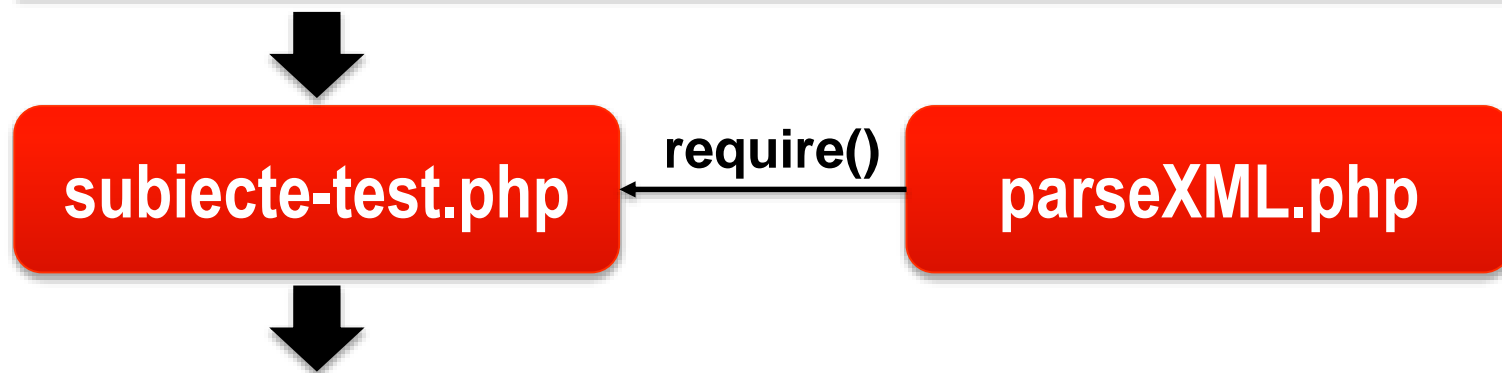
parseXML.php – clasa pentru prelucrarea documentelor XML

subiecte-test.php – program afișând subiectele unor teste



de studiat exemplele
din arhivă

```
<!-- document XML colectând subiecte de teste scrise -->
<subiecte materie="Tehnologii Web" data="2019-04-16">
  <subiect>Ce este Web-ul?</subiect>
  <subiect>Unde e stocat un cookie?</subiect>
  <subiect>La ce este util limbajul XPath?</subiect>
  ...
</subiecte>
```



← → ↻ 🏠 🔒 https://profs.info.uaic.ro/~busaco/php/sax/subiecte-test.php

1. Ce este Web-ul?
2. Unde e stocat un cookie?
3. La ce este util limbajul XPath?

```
class parseXML {  
    private $xml_parser; // instanța analizorului XML  
    private $xml_file;    // numele fișierului XML citit  
    private $html_code; // codul HTML generat  
    private $open_tags; // mulțimea substituțiilor tag-urilor de început  
    private $close_tags; // mulțimea substituțiilor tag-urilor de sfârșit  
  
    public function set_open_tags($tags) {  
        $this->open_tags = $tags;  
    }  
    public function set_close_tags($tags) {  
        $this->close_tags = $tags;  
    }  
    // stabilește numele fișierului XML procesat  
    public function set_xml_file($file) {  
        $this->xml_file = $file;  
    }  
    // furnizează codul HTML generat  
    public function get_html_code() {  
        return $this->html_code;  
    }  
}
```

clasa oferind
prelucrarea
documentelor XML

```
// tratarea evenimentului de apariție a unui tag de început
private function start_element($parser, $name, $attrs) {
    // dacă există în tabloul de substituții, va fi înlocuit cu conținutul precizat
    if ($format = $this->open_tags[$name])
        $this->html_code .= $format;
}

// tratarea evenimentului de apariție a unui tag de sfârșit
private function end_element($parser, $name) {
    if ($format = $this->close_tags[$name])
        $this->html_code .= $format;
}

// tratarea evenimentului de aparitie a unor date de tip caracter
private function character_data($parser, $data) {
    $this->html_code .= $data;
}
```

clasa oferind
prelucrarea
documentelor XML


```
public function parse() { // analiza propriu-zisă condusă de evenimente
    $this->xml_parser = xml_parser_create(); // instanțiază procesorul XML
    // înregistrează funcțiile de analiză ca fiind metode ale obiectului curent
    xml_set_object($this->xml_parser, $this);
    // setează opțiuni (tag-urile nu sunt rescrise cu caractere mari)
    xml_parser_set_option($this->xml_parser,
                          XML_OPTION_CASE_FOLDING, false);
    // stabilește funcțiile de procesare a elementelor XML
    xml_set_element_handler($this->xml_parser,
        "start_element", "end_element");
    xml_set_character_data_handler($this->xml_parser, "character_data");
    if (!$fp = fopen($this->xml_file, "r")) // deschide fișierul XML
        die("could not open XML source");
    while ($data = fread($fp, 4096)) { // citește date în „calupuri” de 4KB
        if (!xml_parse($this->xml_parser, $data, feof($fp))) {
            die(sprintf("XML error: %s at line %d", // eroare de procesare
                xml_error_string(xml_get_error_code($this->xml_parser)),
                xml_get_current_line_number($this->xml_parser))); } }
    } // parse
} // class
```

```
// programul PHP ce afișează propunerile de subiecte de teste scrise
require("parseXML.php");

// substituția via două tablouri asociative a elementelor XML cu cod HTML
// (cheia = element XML de intrare, valoarea cheii = marcaj HTML rezultat)
$open_tags = [
    'subiecte' => "<section><ol>",
    'subiect'  => "<li>"
];
$close_tags = [
    'subiecte' => "</ol></section>",
    'subiect'  => "</li>"
];
// instanțiază și inițializează analizorul
$parser = new parseXML();
$parser->set_xml_file('web-test.xml');
$parser->set_open_tags($open_tags);
$parser->set_close_tags($close_tags);
// rulează analizorul XML
$parser->parse();
echo $parser->get_html_code(); // redarea codului HTML generat
```

```
// programul PHP ce afișează propunerile de subiecte de teste scrise
require("parseXML.php");

// substituția via două tablouri asociative a elementelor XML cu cod HTML
// (cheia = element XML de intrare, valoarea cheii = marcaj HTML rezultat)
$open_tags = [
    'subiecte' => "<section><ol>",
    'subiect'  => "<li>"
];
$close_tags = [
    'subiecte' => "</ol></section>",
    'subiect'  => "</li>"
];
// instanțiază și inițializează analizorul
$parser = new parseXML();
$parser->set_xml_file('web-test.xml');
$parser->set_open_tags($open_tags);
$parser->set_close_tags($close_tags);
// rulează analizorul XML
$parser->parse();
echo $parser->get_html_code(); // redarea codului HTML generat
```

arborele DOM al
reprezentării obținute

```
<section>
  <ol>
    <li>Ce este Web-ul?</li>
    <li>Unde e stocat un cookie?</li>
    <li>La ce este util limbajul XPath?</li>
  </ol>
</section>
```

(în loc de) pauză



When HTML is life...

www.reddit.com/r/ProgrammerHumor/

sax vs. dom

Când poate fi 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 poate fi folosit SAX?

crearea unei structuri noi de document XML

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

sax vs. dom

Când poate fi 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**:

developer.android.com/reference/javax/xml/parsers/SAXParser.html

cod demonstrativ pentru **iOS** – SeismicXML:

developer.apple.com/library/archive/samplecode/SeismicXML/

sax vs. dom

Când poate fi 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

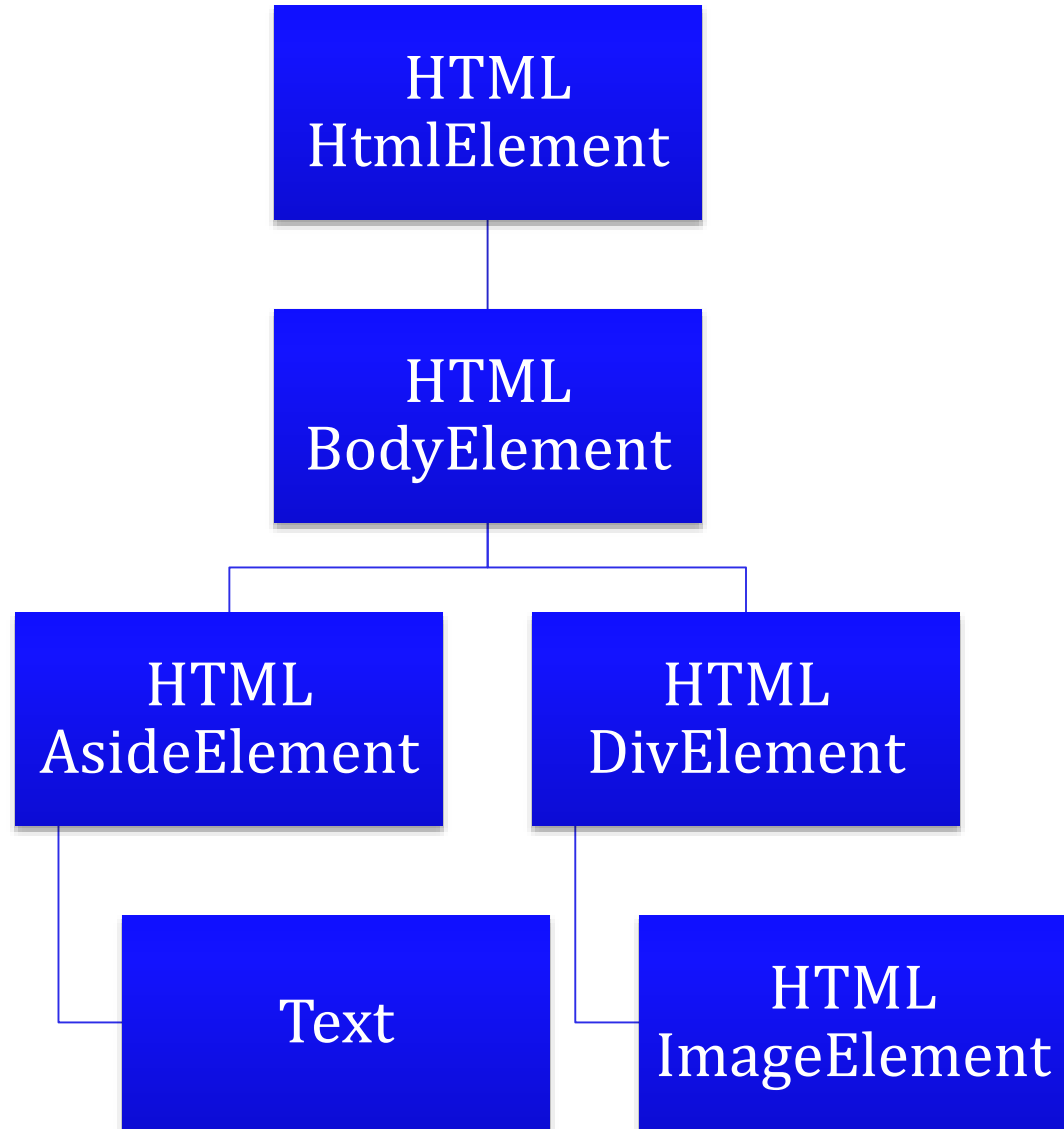
Când poate fi utilizat DOM?

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

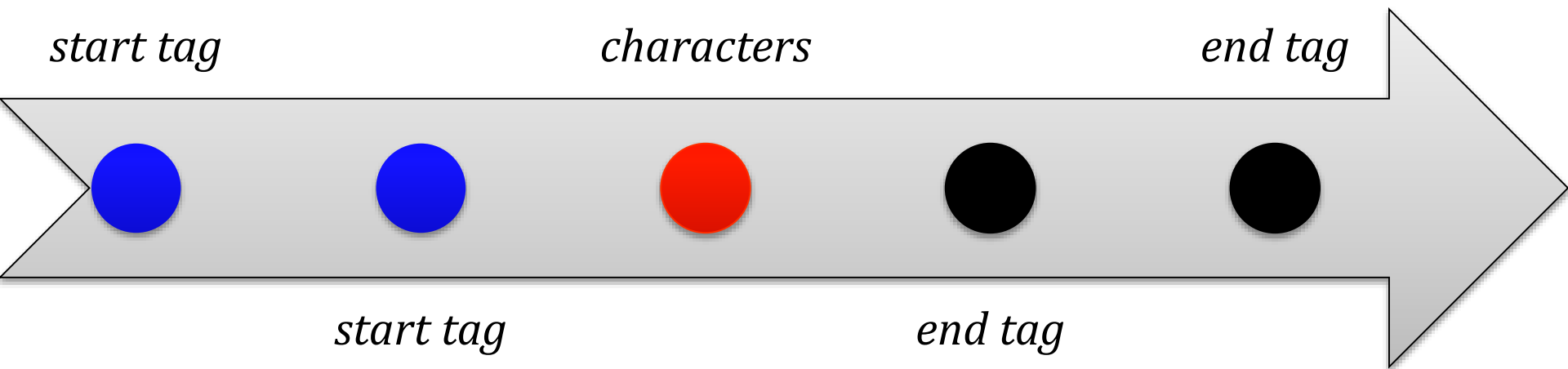
sax vs. dom

DOM necesită încărcarea
completă în memorie
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 foarte mare?

sax vs. dom

Unele implementări SAX oferă suport
pentru validări și transformări

uzual, se folosesc și DOM și SAX

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

www.xmlpull.org

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

xml pull parsing – implementări

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

exemple de implementări:

Aalto – github.com/FasterXML/aalto-xml

Javolution – axat asupra performanței: javolution.org

Oracle StAX – inclus în XDK (*XML Developer's Kit*)
docs.oracle.com/javase/tutorial/jaxp/stax/

Woodstox – github.com/FasterXML/woodstox

xml pull parsing – implementări

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

pull – pachet Scala de procesare XPP

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

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

xml.dom.pulldom – soluție Python

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

acces direct, în stilul *pull*

SAX

acces secvențial, în stilul *push*

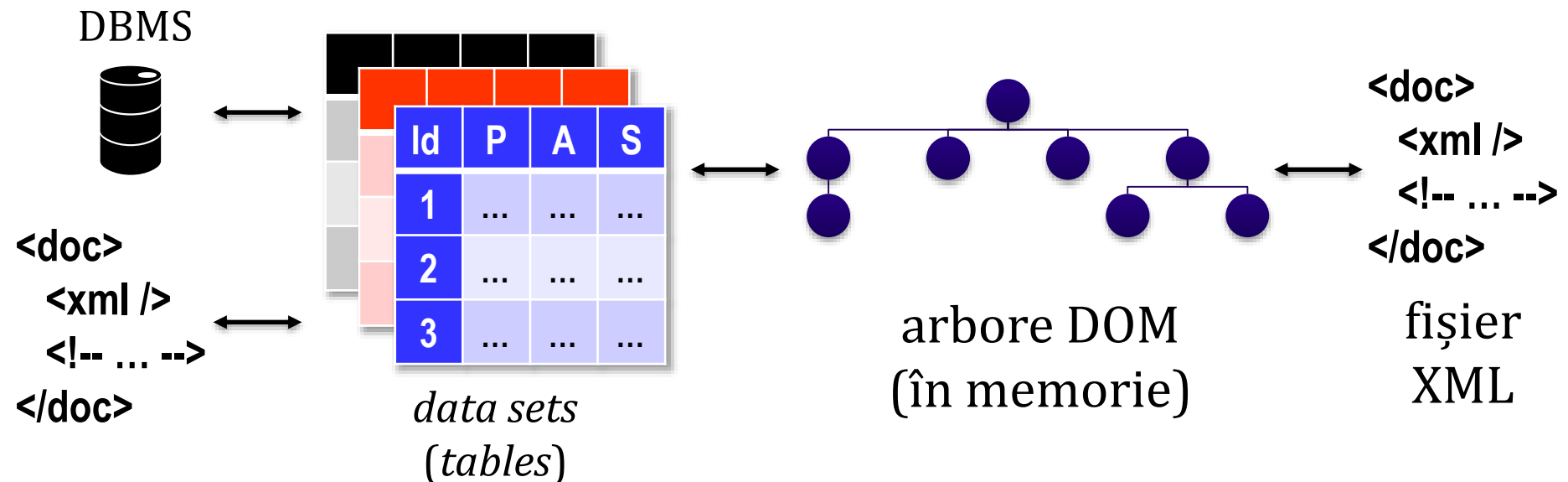
XPP și .NET XmlTextReader

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

XML binding

baze de date: *XML infoset* \leftrightarrow *dataset*

specificația **SQL/XML** – vezi standardul SQL:2016-14

aspecte de interes:

tipul XML pentru valori ale câmpurilor tabelelor
recurgerea la predicate + funcții specifice XML

baze de date: *XML infoset* \leftrightarrow *dataset*

implementări concrete:

Oracle XML DB

docs.oracle.com/cd/B28359_01/appdev.111/b28369/xdb01int.htm

a se consulta și SQLXML (J2SE 8+)

docs.oracle.com/javase/tutorial/jdbc/basics/sqlxml.html

pureXML (IBM DB2)

www.ibm.com/developerworks/data/library/techarticle/dm-0603saracco2/

XML Data (Microsoft SQL Server)

docs.microsoft.com/en-us/sql/relational-databases/xml/

a se studia și clasa SqlXml (.NET Framework)

docs.microsoft.com/en-us/dotnet/api/system.data.sqltypes.sqlxml

XML Functions (PostgreSQL)

www.postgresql.org/docs/current/static/functions-xml.html

alternative

XML binding

abordare obiectuală:
date XML \leftrightarrow clase create „din zbor”
(*serialization, marshalling*)

abordare obiectuală:
date XML \leftrightarrow clase create „din zbor”

exemple:

C++ – **cereal**: uscilab.github.io/cereal/

C++, C#, Go, Java, Ruby, Python,... – **Protocol Buffers**
developers.google.com/protocol-buffers/

JS – **node-xml2js**: github.com/Leonidas-from-XIV/node-xml2js

.NET (C# *et al.*) – clasa **XmlSerializer**
docs.microsoft.com/en-us/dotnet/standard/serialization/introducing-xml-serialization

PHP – **SimpleXML**: php.net/manual/en/book.simplexml.php

Python – **Untangle**: github.com/stchris/untangle

Scala – **scalaxb**: scalaxb.org

alternative

XML binding

interogări asupra datelor XML
direct în limbajul de programare

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

docs.microsoft.com/dotnet/articles/csharp/programming-guide/concepts/linq/linq-to-xml

XDocument proiecte; // XDocument e o clasă .NET

avansat

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

```
var proiecteM =
```

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

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

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

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

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

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

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

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

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

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

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

```
}
```

// același rezultat, recurgând la XPath

```
var proiecteM2 = (IEnumerable)
```

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

alternative

XML binding

JAXB – *Java Architecture for XML Binding* (JSR-222)

jcp.org/en/jsr/detail?id=222

tutorial: docs.oracle.com/javase/tutorial/jaxb/

implementarea de referință: github.com/eclipse-ee4j/jaxb-ri

de experimentat și **EclipseLink**: www.eclipse.org/eclipselink/

alternative

XML binding

interoperabilitate cu alte formate: XML ↔ JSON

nu există o metodă standardizată

exemplificări de instrumente și biblioteci:

Apache Camel (Java), js2xmlparser (Node.js),
JSON-lib (Java), ruby-xml-to-json, x2js (JavaScript),
xml2json (Node.js), xml-to-json (Haskell), xmlutils.py

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:

libxml (C, C++ și alte limbaje)

SimpleXML (PHP) – php.net/manual/en/book.simplexml.php

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

XmlSimple (Ruby)

XmlTextReader + **XmlTextWriter** (.NET)

```
// încarcăm documentul XML
$xml = simplexml_load_file('http://web.info/projects.xml');
// afișăm descrierile proiectelor de clasă M
foreach ($xml->project as $proj) {
    if ($proj['class'] == 'M') {
        echo '<p>' . $proj->desc . '</p>';
    }
}
// similar, dar utilizând XPath
foreach ($xml->xpath("//project[@class='M']") as $proj) {
    echo '<p>' . $proj->desc . '</p>';
}
```

de consultat
exemplele din arhivă

alternative

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

exemple:

modulul **xmlreader** pentru Node.js

xmlReader oferit de biblioteca **libxml** (C *et al.*)

XMLReader (PHP) – php.net/manual/en/book.xmlreader.php

clasa **XmlReader** oferită de .NET (C# *et al.*)

alternative

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

exemplificări:

clasa **XmlWriter** pentru .NET

xmlWriter din cadrul bibliotecii **libxml** (C *et al.*)

XMLWriter (PHP) – php.net/manual/en/book.xmlwriter.php

modulul **xml-writer** pentru Node.js

Cum pot fi procesate documentele HTML?

procesare html

Aspect de interes:
ignorarea erorilor de sintaxă

documente bine formatate (*well formed*)
versus
documente valide

procesare html

Aspect de interes:
ignorarea erorilor de sintaxă

malformed markup

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 uzual – nerecomandată

Web scraping

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

un exemplu recurgând la expresii regulate:

scraping.pro/scraping-in-php-with-curl/

procesare html

Recurgerea la un procesor HTML/XML specific

scopuri importante:

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

+

detectarea și repararea erorilor sintactice (*HTML clean*)

vezi și cursurile
anterioare

procesare html: instrumente

Beautiful Soup – bibliotecă Python
www.crummy.com/software/BeautifulSoup/

goquery și soup – biblioteci Go
github.com/PuerkitoBio/goquery
github.com/anaskhan96/soup

Gumbo – procesor HTML5 implementat în C (Google)
github.com/google/gumbo-parser

html5lib – procesare + serializare HTML pentru Python
github.com/html5lib

procesare html: instrumente

HTML-Parser – modul Perl
github.com/gisle/html-parser

HAP (Html Agility Pack) – bibliotecă .NET (C#)
html-agility-pack.net

Jericho HTML Parser – bibliotecă Java de procesare HTML
jericho.htmlparser.net

jsoup – bibliotecă Java pentru HTML5
jsoup.org

procesare html: instrumente

Masterminds HTML5-PHP – procesor HTML5 în PHP
github.com/Masterminds/html5-php

Parse5 – modul Node.js
github.com/inikulin/parse5

Simple HTML DOM Parser – bibliotecă PHP
simplehtmldom.sourceforge.net

SwiftSoup – bibliotecă Swift (macOS, iOS, tvOS + Linux)
www.scinfu.com/SwiftSoup/

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 HTML5

procesare html: instrumente – exemplu

Preluarea datelor despre produse (albume musicale)
– denumire, cod de bare și preț – din documente HTML
(invalide) oferite de un sit Web de profil

se recurge la biblioteca **Simple HTML DOM Parser**
disponibilă sub licență deschisă (*open source*)

```

<div class="row">
  <div class="col-sm-12 title">
    <h1>Pink Floyd - <span class="alb">The Wall</h1>
  </div>
</div>
<div class="row">
  <div class="col-sm-6 left-col">
    <div class="cover">
      <a title="Pink Floyd - The Wall" id="product_img"
    <div class="info">
      <div class="row up">
        <div class="col-xs-6 section">
          <span class="text">Contine:</span>
          <span class="value">1 DVD</span>
        </div>
        <div class="col-xs-6 section">
          <span class="text">Piese:</span>
          <span class="value">36</span>
        </div>
      </div>
      <div class="row down">
        <div class="col-xs-6 section">
          <span class="text">Durata:</span>
          <span class="value">79:47</span>
        </div>
        <div class="col-xs-6 section">
          <span class="text">Anul:</span>
          <span class="value">1984</span>
        </div>
      </div>
    </div>
  </div>
  <div class="col-sm-6 right-col">
    <span class="barcode">Cod produs: 5099705019894</span>
    <span class="price">40.00 Lei</span>
    <span class="vat">include TVA 19%</span>
  </div>
</div>

```

incorect!
(malformed markup)

```
require_once('simple_html_dom.php');
define('URL', 'https://www.nicherecords.ro/catalog/'); // URL-ul sitului sursă

$cai_produce = [ // căile spre produse (i.e. documentele HTML procesate)
    "pop-rock/rock-psihedelic/pink-floyd-the-wall-dvd.html",
    "jazz/jazz-1/nina-simone-silk-soul-vinyl.html",
    "clasica/clasica-1/bach-simply-bach-cd.html",
    "metal/heavy-metal/king-diamond-graveyard-vinyl-1.html" ];
foreach ($cai_produce as $scale) { // preluăm date despre fiecare produs
    $html = file_get_html(URL . $scale);

    $prod = $html->find("span.alb", 0)->plaintext;
    $cod_bare = explode(":", $html->find("span.barcode", 0)->plaintext)[1];
    $pret = (int) explode(" ", $html->find("span.price", 0)->plaintext)[0];

    printf("<p><a title='Detalii' href='%s'>%s</a> &ndash;
        cod de bare: <code>%s</code>, preț: %d RON.</p>",
        URL . $scale, $prod, $cod_bare, $pret);
}
```

procesare html: instrumente

[The Wall](#) – cod de bare: 5099705019894, preț: 40 RON.

[Silk & Soul](#) – cod de bare: 8713748981037, preț: 120 RON.

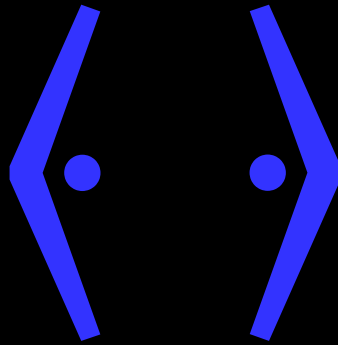
[Simply Bacn](#) – cod de bare: 0698458242327, preț: 40 RON.

[Graveyard](#) – cod de bare: 0039842506715, preț: 140 RON.

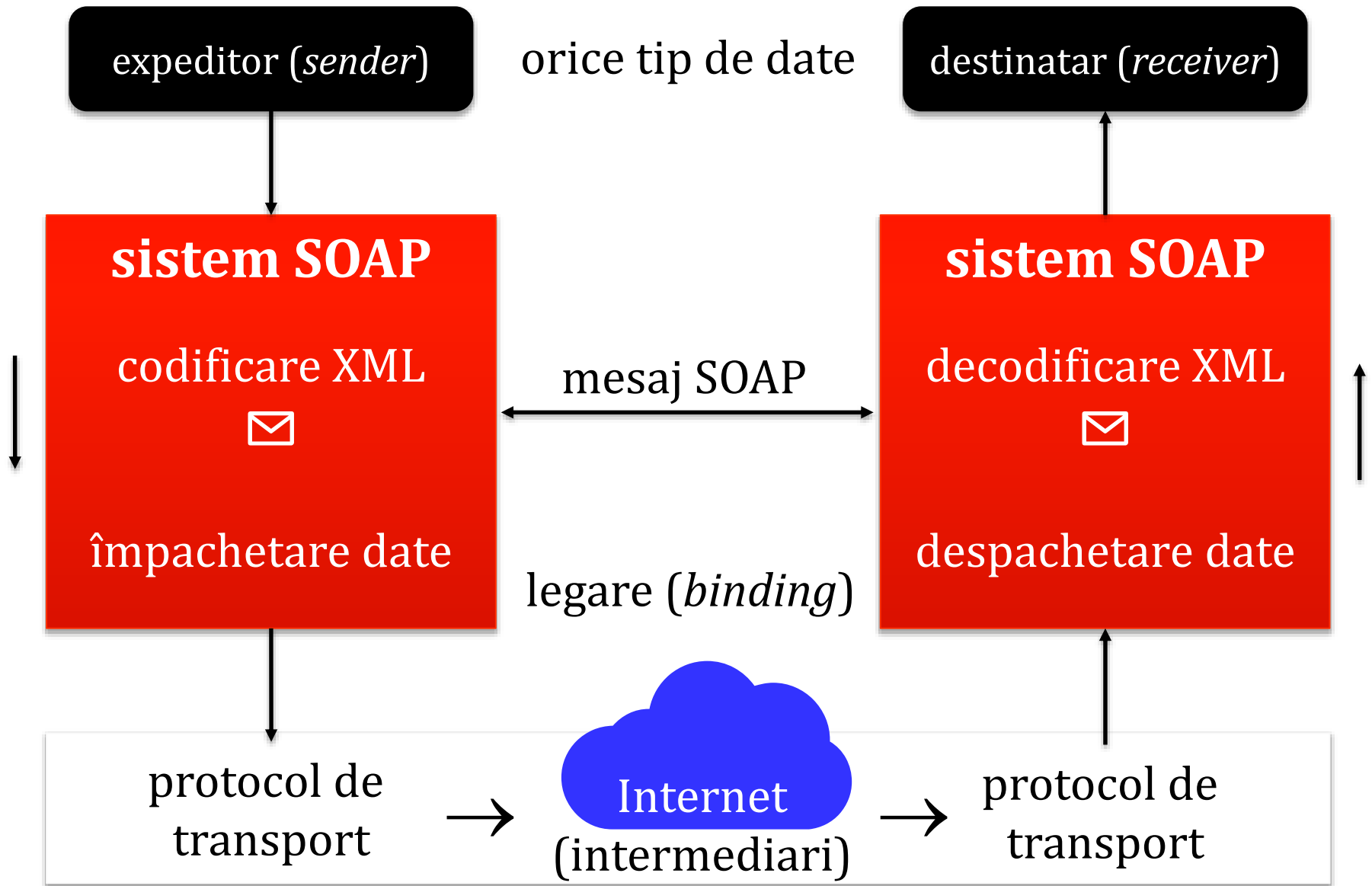
<https://www.nicherecords.ro/catalog/jazz/jazz-1/nina-simone-silk-soul-vinyl.html>

reprezentarea HTML a datelor preluate de pe situl Web

rezumat



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



episodul viitor: **servicii Web prin SOAP**