3. Data Access

Adrian Adiaconitei

LINK Academy

Obiective

- ✓ Structura si regulile continutului de tip XML
- ✓ Validitare XML
- ✓ Avantaje si utilitate XML
- ✓ XML și JavaScript
- ✓ JSON
- ✓ Parsarea JSON
- ✓ Serializarea JSON

XML

CE ESTE XML (eXtensible Markup Language) ?

- ✓ este un set de reguli ce permit crearea unor limbaje de transport al datelor.
- ✓ este un limbaj de markup care defineste un set de reguli pentru encodarea continutului de tip text al unui fisier intr-un format care poate fi citit cu usurinta de catre oameni si interpretat cu usurinta de catre calculatoare
- ✓ deriva din SGML (Standard Generalized Markup Language), o specificatie complexa ce reprezinta parintele multor alte limbaje de acest fel (HTML, DocBook etc).
- ✓ XML foloseste etichete (tag-uri) si atribute

XML

Un fisier XML este un fisier text structurat, care contine:

- ✓ datele utile – informatia ce se doreste a fi transmisa
- ✓ informatii despre aceste date (meta-data) - de exemplu, felul in care datele se afla in relatie unele cu altele. Metainformatia este specificata cu ajutorul unor tag-uri (marcatoare), asemanatoare cu cele folosite in codul sursa HTML.

Un fisier XML este asemanator cu unul HTML:

- ✓ HTML a fost gandit pentru a specifica structura si modul de prezentare a datelor, pe cand XML are ca scop memorarea structurata a datelor in scopul transportului lor intre sisteme de calcul diferite
- ✓ HTML este un limbaj, el dispunand de un set prestabilit de tag-uri; XML este o specificatie ce permite programatorului sa defineasca propriile tag-uri, care sa descrie cat mai bine datele utile si relatiile dintre ele

XML

Avantaje si utilitate XML

- ✓ continutul este usor de vizualizat si inteles pentru noi, spre deosebire de fisiere cu format binar – ex: Word, Excel
- ✓ sunt usor de "inteles" pentru o aplicatie (software-ul care analizeaza fisierul si recunoaste elementele de structura si datele utile este suficient de simplu de implementat)
- ✓ nu depind de platforma hardware sau software pe care ruleaza aplicatiile ce lucreaza cu XML.
- ✓ Un fisier XML poate fi privit ca o “baza de date”, usor de manipulat

XML

SINTAXA:

- ✓ Jon Bosak , 1996
- ✓ Consortiu World Wide Web (W3C) combina puterea SGML-ului cu simplitatea HTML-ului realizand limbajul XML
- ✓ In 1998 este impus ca standard de W3C, ultima versiune de standard din Februarie 2004 fiind XML 1.0.
- ✓ Include multe din caracteristicile SGML-ului, printre care: structura si validarea

XML

```
<html>
<body>
<h2>Adrian Adiaconitei</h2>
<p>Link Academy<br>
https://www.link-academy.com/<br>
aadiaconitei@gmail.com<br>
</p>
</body>
</html>
```

```
<contact>
<name>Adrian Adiaconitei</name>
<address>Link Academy</address>
<web-page>https://www.link-
academy.com/</web-page>
<email>aadiaconitei@gmail.com</emai>
</contact>
```

XML

- ✓ Un document/fisier XML are extensia .xml
- ✓ Un document/fisier XML este **Case sensitive**
- ✓ Un document XML este format din:
  - ✓ **marcaje (tag-uri)**
  - ✓ **date caracter**

  Un marcaj (tag) este un sir de caractere delimitat de caracterele "<" si ">".
  Datele caracter reprezinta continutul marcajelor.
- ✓ Un fisier XML cuprinde urmatoarele sectiuni:
  - ✓ **Prolog (optional , <?xml version="1.0" encoding="UTF-8"?>)**
  - ✓ **Definitia tipului de document (optionala, DTD -** un set de reguli ce defineste structura unui fisier XML**)**
  - ✓ **Elementul radacina(**Un document XML are un singur element radacina)

Declararea XML-ului

- ✓ Fiecare document XML trebuie să conţină o declaraţie prin care va fi identificat documentul ca un document XML. Aceasta este prima construcţie din document:

•Forma de bază a declaraţiei XML:
<?xml version =“1.0”?>

- ✓ Forma opţională a declaraţiei XML:

<?xml version =“1.0” encoding= “UTF-8”?>
<?xml version =“1.0” encoding= “UTF-16”?>

- ✓ Forma generală a declaraţiei XML:

<? xml **version**=‘1.0’ **encoding=‘**character encoding’ **standalone**=‘yes|no’?>

Exemplu

- ✓ <?xml version='1.0' ?>
- ✓ <?xml version='1.0' encoding='US-ASCII' ?>
- ✓ <?xml version='1.0' encoding='US-ASCII‘ standalone='yes' ?>
- ✓ <?xml version='1.0' encoding='UTF-8' ?>
- ✓ <?xml version='1.0' encoding='UTF-16' ?>
- ✓ <?xml version='1.0' encoding='ISO-10646-UCS-2'?>
- ✓ <?xml version='1.0' encoding='ISO-8859-1' ?>
- ✓ <?xml version='1.0' encoding='Shift-JIS' ?>

Atributul version

- ✓ Atributul *version* trebuie să aibă valoarea 1.0. În situaţii extrem de rare poate primi valoarea 1.1. Pentru că atribuirea valorii 1.1 limitează utilizarea la un număr mai mic de analizatori, nu ar trebui aleasă această versiune fără un adevărat motiv.
- ✓ Dacă nu vorbiţi limba birmană, mongolă, cambodgiană, amhară sau maldiviană/dhivehi, dacă nu folosiţi semnele de control CO non-textuale (tabulatorul vertical, noua pagină, clopoţelul), nu aveţi niciun motiv să folosiţi versiunea 1.1.

Atributul encoding

- ✓ XML este un document format dintr-un text pur.
- ✓ Cum arată semnele într-un astfel de text? Sunt de tip ASCII, Latin-1, Unicode?
- ✓ Toate documentele XML sunt, implicit, codate cu coduri UTF-8, cu o lungime variabilă în setul Unicode.
- ✓ În urma analizei fişierului XML, analizatorul citeşte din metadate tipul de codare al acestui fişier şi îl aplică. Cu toate acestea, nu toate sistemele furnizează acest tip de codare și uneori este necesar să le enumerăm în document folosind declarația de codare de mai sus.
- ✓ Dacă datele despre coduri nu sunt prezente în fişier şi nu se specifică în document, analizatorul presupune că este vorba de Unicode.
- ✓ Dacă există ambele date, dar sunt contrare, analizatorul are încredere în sistem, adică respectă codarea documentului.

Atributul standalone

- ✓ Dacă acest atribut are valoarea *“no”,* atunci aplicaţia poate încărca DTD-ul extern, prin care se stabilesc valorile şi semnificaţia anumitor părţi din document.
- ✓ Documentul care nu are DTD, poate avea acest atribut setat la valoarea *“yes”.*
- ✓ Dacă este omis, se presupune că are valoarea *“no”.*

XML - Root

- ✓ Documentul XML conţine un element fără părinte.
- ✓ Acesta este primul element care conţine toate celelalte elemente.
- ✓ Acest element se numeşte element rădăcină (engl. *root).*

XML

In XML etichetele(**marcaje / tag-uri)** sunt de 2 feluri:

- de inceput sau de deschidere (open tag): **<post>**, **<companie>**, **<nivel>**

- de final sau de inchidere (end tag): **</post>**, **</companie>**, **</nivel>**

Continutul de tip text al unui element nu poate contine caracterele **<** si **>**, aceste caractere trebuie encodate cu **<** respectiv **>**

Entitate	Referinta la entitate
<	<
>	>
&	&
'	'
"	&quote;

XML

- ✓ Sunt doua modalitati (din punct de vedere al structurii) de a scrie date intr-un fisier XML

Exemplul: **Programator PHP Senior**

```
<posturi>
    <post>
        <titlu>Programator PHP</titlu>
        <nivel>Senior</nivel>
    </post>
</posturi>
```

sau

```
<posturi>
    <post
        titlu="Programator PHP"
        nivel="Senior"/>
</posturi>
```

Potenţialele probleme (restricţii) în timpul folosirii atributelor

1. Atributele nu pot conţine valori multiple (elementele pot).
2. Atributele nu se pot extinde aşa de uşor.
3. Atributele nu descriu structura.
4. Atributele se manipulează mai greu în codul de programare.
5. Valorile atributelor se testează mai greu în raport cu DTD (engl. Document Type Definition) – definirea tipului de document.

XML

- ✓ **Aplicatie:**

Creati un document XML care sa reprezinte un catalog de articole. Catalogul trebuie sa contina minim 3 articole. Includeti numele articolului (spre exemplu: Coca Cola, Ciocolata, HP Pavilion, Aspirator etc) si tipul (categoria) articolului (spre exemplu: Bauturi, Alimente, Laptopuri, Electrocasnice etc). Fiecare articol trebuie sa aiba un identificator unic.

Catalog1.xml

XML

Un document XML poate contine urmatoarele tipuri de marcaje:

- ✓ **Elemente** <nume_tag> ..... </nume_tag> sau <nume_tag/>
- ✓ **Atribute** <nume_tag numeAtr1="val1" ... numeAtrN="valN">
- ✓ **Comentarii** <!-- comentariu -->
- ✓ **Entitati** &nume_entitate;
- ✓ **Sectiuni CDATA** <![CDATA[.....]]>
- ✓ **Instructiuni de procesare** <?xml-stylesheet type="text/css" href="mySheet.css"?>
- ✓ **Declaratia tipului de document** (DTD -Document Type Definition)

Entitate	Referinta la entitate
<	<
>	>
&	&
'	'
"	&quote;

XML/PHP Programming

CDATA pentru a include blocuri de text continand caractere care altfel ar fi recunoscute ca marcaje. Sunt folosite atunci cand dorim ca datele incluse in interiorul lor sa nu fie interpretate de catre analizor, ci sa fie considerate date caracter

```
<?xml version="1.0" encoding="UTF-8"?>
<exemplu>
        Un exemplu de creare a unui tabel in HTML:
        <![CDATA[
                <table align="center">
                        <tr><td>Coloana 1</td><td>Coloana 2</td></tr>
                </table>
        ]]>;
</exemplu>
```

XML

- ✓ **Declaratia tipului de document:** marcaj special ce poate fi inclus in documentele XML cu rolul de a specifica existenta si locatia definitiei tipului de document (DTD -Document Type Definition) https://www.w3schools.com/xml/xml_dtd.asp
- ✓ DTD-ul este un set de reguli care definesc structura unui document XML
- ✓ se face intre prolog si elementul radacina
- ✓ sintaxa declaratiei difera in functie de tipul DTD-ului: intern sau extern.

Sintaxa declararii tipului de document cu DTD intern:

```
<!DOCTYPE element_radacina [
<!-- Setul de reguli-->
]>
```

Sintaxa declararii tipului de document cu DTD extern:

```
<!DOCTYPE root SYSTEM "reguli.dtd">
```

Validare XML- **Documente well-formed**

- ✓ **reguli ce se aplica elementelor XML**
  - ✓ fisierul trebuie sa contina un singur element radacina
  - ✓ fiecare element ce contine date trebuie sa aiba atat tag de inceput, cat si de incheiere sau tagul vid
  - ✓ tagul de inceput poate contine spatiu intre nume si >, insa nu si intre < si nume (ex: este permis, pe cand < nume> este invalid)
  - ✓ fiecare element trebuie sa fie continut integral in interiorul altuia (cu exceptia elementului radacina)
  - ✓ numele elementelor:
    - ✓ sunt case-sensitive (nu putem scrie ca in HTML)
    - ✓ nu pot incepe cu xml (in orice combinatie de litere mici/mari)
    - ✓ pot incepe doar cu litera sau – (minus), fiind permise pe pozitiile urmatoare si numere, minus si punct.
    - ✓ pot contine caractere cu semne diacritice (ex: â, î, é etc)

Validare XML - **Documente well-formed**

- ✓ **reguli ce se aplica atributelor**
  - ✓ numele atributelor respecta aceleasi reguli ca si numele de elemente
  - ✓ in cadrul aceluiasi element nu este permisa existenta mai multor atribute cu acelasi nume
  - ✓ fiecare atribut trebuie sa aiba valoare (nu este permis, de exemplu,)
  - ✓ valoarea fiecarui atribut trebuie inclusa intre " " sau ' '. In interiorul ghilimelelor pot fi folosite apostroafe si invers

Validare XML - **Documente well-formed**

- ✓ **reguli pentru CDATA**
  - ✓ nu este permisa folosirea caracterelor , &, " si '.
  - ✓ Daca CDATA contine asemenea simboluri, pentru a preintampina interpretarea lor ca metacaractere (inceput de tag, sfarsit de tag etc) ele trebuie reprezentate folosind secvente predefinite – asa-numitele entity references (vezi tabelul alaturat)

Un document care se conformeaza acestor reguli are proprietatea de a fi **well-formed** (corect din punct de vedere sintactic, si deci parsabil de catre o aplicatie).

XML

Exemplu de DTD declarant intern

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE limbaje [
<!ELEMENT limbaje (limbaj+)>
<!ELEMENT limbaj (nume,descriere?)>
<!ELEMENT nume (#PCDATA)>
<!ELEMENT descriere (#PCDATA)>
]>
<limbaje>
<limbaj>
<nume>PHP</nume>
<descriere>Descrierea limbajului PHP</descriere>
</limbaj>
<limbaj>
<nume>Java</nume>
<!-- Fără descriere -->
</limbaj>
</limbaje>
```

!DOCTYPE limbaje declară că elementul rădăcină al documentului va fi elementul limbaje

!ELEMENT limbaje (limbaj+) declară că elementul limbaje trebuie să conțină cel puțin (caracterul +) un element numit limbaj

!ELEMENT limbaj (nume,descriere?) declară că elementul limbaj trebuie sa conțină un singur element nume și zero sau un element descriere (caracterul ?) **exact în această ordine**

!ELEMENT nume (#PCDATA) declară că elementul nume este de tip PCDATA (are conținut interpretabil)

!ELEMENT descriere (#PCDATA) declară că elementul descriere este fie de tip PCDATA

PCDATA înseamnă conținut interpretabil (parsed character data). Acest tip de conținut va fi interpretat de către parser

XML

Aplicatie: catalog3.xml

Adaugati un DTD in cadrul documentului catalog.xml . DTD-ul trebuie sa specifice:

- elementul radacina al documentului
- catalogul poate contine zero sau mai multe articole
- un articol poate avea un singur nume, o singura categorie si optional o descriere
- numele si categoria articolului au continut interpretabil
- descrierea articolului trebuie poate sa contina HTML

XML

Aplicatie: catalog4.xml

Extrageti DTD-ul din catalog3.xml intr-un fisier separat si documentul XML sa facă referire la acest DTD extern

Nota: Fisierul trebuie sa fie creat in acelasi director in care se afla documentul XML

Spaţiile goale în XML

- În XML se salvează spaţiul gol. Folosind XML, spaţiul gol este prezentat în documentul parsat. De exemplu:

<body>Multe salutări din Timişoara</body>

va fi parsat ca:

Multe salutări din Timişoara

Exemple de spaţii goale utilizate corect

```
<pre:Vehicle xmlns:pre='urn:example-org:Transport'
type='car' >
<seats> 4 </seats>
<colour> White </colour>
<engine>
<petrol />
<capacity units='cc' >1598</capacity>
</engine >
</pre:Vehicle >
```

Exemple de spaţii goale utilizate incorect

- <pre:Vehicle xmlns:pre='urn:exampleorg: Transport‘ Type>='car'>
- <seats>4</seats>
- </pre:Vehicle>

Validare XML - **Documente valide**

- ✓ Documentul este well-formed, insa am dori sa putem impune cateva restrictii(comparam cu o tabela dintr-o baza de date, in care fiecare inregistrare trebuie sa aiba o anumita lista de coloane). Cu alte cuvinte, sa definim un set de reguli suplimentare, specifice aplicatiei ce genereaza/parseaza continutul.
- ✓ Specificarea acestor conditii suplimentare se poate realiza in doua moduri:
  - ✓ **DTD** (Document Type Definition) - modalitatea care foloseste un limbaj diferit de XML
  - ✓ **XSD** (XML schema definition) - solutia moderna si care foloseste chiar XML pentru specificarea elementelor de structura.
- ✓ Un document care este well-formed si, in plus, se conformeaza restrictiilor suplimentare de structura poarta denumirea de **document XML valid**.
- ✓ Un document XML poate fi well-formed dar nu valid, in schimb nu poate fi valid fara a fi well-formed

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<limbaje
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="schema.xsd">
<limbaj>
<nume>PHP</nume>
</limbaj>
<limbaj>
<nume>Javascript</nume>
</limbaj>
<limbaj>
<nume>Java</nume>
</limbaj>
</limbaje>
```

Pentru a atasa schema de validare documentului XML adaugam atributul **xsi:noNamespaceSchemaLocation** la elementul radacină si ii atribuim valoarea **schema.xsd** care reprezinta calea catre fisierul care contine declaratia schemei de validare

Prefixul **xsi** este mapat catre **http://www.w3.org/2001/XMLSchema-instance**

Prefixul poate fi schimbat atata timp cat valoarea ramane **http://www.w3.org/2001/XMLSchemainstance**

Nu este obligatoriu sa atasam schema de validare la elementul radacina ci o putem atasa elementului de unde dorim sa incepem validarea

XML

Conținutul fișierului schema.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="limbaje"/>
<xs:complexType>
<xs:sequence>
<xs:element name="limbaj" type="TipLimbaj"
minoccurs="1" maxoccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:complexType name="TipLimbaj">
<xs:sequence>
<xs:element name="nume" type="xs:string"
minoccurs="1" maxoccurs="1"/>
</xs:sequence>
</xs:complexType>
</xs:schema>
```

Elementul radacina al acestui document este **xs:schema**. Acesta face parte din spațiul de nume **http://www.w3.org/2001/XMLSchema**. Acest spațiu de nume este alocat prefixului **xs** (nu conteaza ce prefix este folosit atâta timp cât namespace-ul ramâne același). Doua dintre cele mai des folosite prefixe sunt **xs** și **xsd.** Elementele sunt declarate folosindu-ne de eticheta **xs:element** care provine din spațiul de nume **http://www.w3.org/2001/XMLSchema**. În cazul de față declarăm că documentul poate conține un element cu numele **limbaje** care poate conține o secvență (**xs:sequence**) de elemente cu numele **limbaj** care poate apărea cel puțin odată Specificarea elementul **limbaj** este realizată cu ajutorul etichetei **xs:complexType** care provine din spațiul de nume **http://www.w3.org/2001/XMLSchema**. În exemplul de față elementul de tip **TipLimbaj** poate conține un singur element **nume** de tip string

XML

- ✓ **DTD** (document type definition)
  - ✓ DTD este mai ușor de folosit și înțeles însă este mai limitat
  - ✓ Se pretează la documente cu o structură relativ simplă
  - ✓ Poate fi declarat direct în cadrul documentului XML sau intr-un fisier separat caz în care
  - ✓ documentul XML va face referire la acest fișier extern
- ✓ **XSD** (XML schema definition)
  - ✓ XSD este mai complex , poate acoperi o plaja mult mai mare de reguli pentru schema documentului
  - ✓ Poate fi declarat numai intr-un fisier extern, deci documentul XML va trebui sa faca referinta catre acesta

XML

Aplicatie:

Adaugati o schema de validare pentru documentul catalog5.xml . Schema de validare trebuie sa cuprinda:

- elementul radacină al documentului
- catalogul poate contine zero sau mai multe articole

un articol poate avea un singur nume, o singura categorie si optional o descriere

- numele si categoria au continut interpretabil
- descrierea articolului trebuie sa poata contine HTML

Parsarea XML-ului

- Trebuie creat un parser.
- Browserele au încorporat obiectul de tip parser (în afară de IE6 şi IE7, care au ActiveXParser).

Parsarea XML-ului

```
var parser = new DOMParser()


xmlDoc =
    parser.parseFromString(text,"text/xml");
  Ap1.html
  Ap2.html
```

Parsarea XML-ului (versiunile de IE mai vechi)

```
xmlDoc = new
   ActiveXObject("Microsoft.XMLDOM");


xmlDoc.async = false
xmlDoc.loadXML(text);
```

innerHTML vs. textContent

- innerHTML tratează conţinutul ca HTML.
- textContent oferă doar text.

Exemplu de citire

- var root = xmlDoc.children[0];

```
for(var i=0; i < root.children.length; i++){
    var c = root.children[i];
    console.log(c.textContent);
}
```

Exemplu de citire

- var root = xmlDoc.children[0];

```
for(var i=0; i < root.children.length; i++){
    var c = root.children[i];
    console.log(c.attributes[“unit”].value);
}
```

XPath

- XML Path Language
- O altă modalitate de parsare a fişierelor XML.
- Este necesar parserul XPath.

XPath

nodename – returnează toate nodurile cu numele “nodename”;

/ - returnează nodurile din elementul rădăcină;

// - returnează toate nodurile din document, indiferent unde se află;

. – returnează nodul curent;

.. – returnează tot din nodul curent;

@ - returnează atributele.

XPath

- * - orice element;
- @* - orice atribut;
- node() – orice element sau atribut.

XPath

- xmlDoc.evaluate(
  xpath,
  xmlDoc,
  null,
  XPathResult.ANY_TYPE,
  null);

XPath

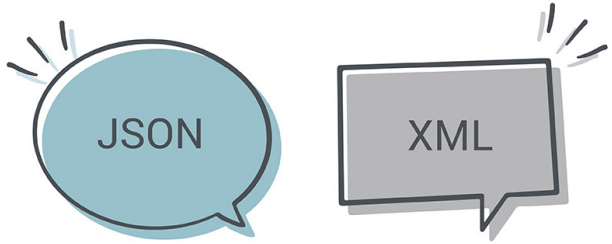
```
var parser = new DOMParser();
var doc = parser.parseFromString(xmlStr, "text/xml");

var ev = doc.evaluate("/bookstore/book/title", doc, null,
    XPathResult.ANY_TYPE, null);
var result = ev.iterateNext();
while(result) {
        console.log(result.children[0].textContent);
        result = ev.iterateNext();
}
```

Book.html

JSON

- ✓ JavaScript Object Notation.
- ✓ Nu îl confundaţi cu obiectul JS.
- ✓ JSON este reprezentarea textuală a obiectelor JS.

JSON vs XML




```
{
    "heroes":[
        {
            "name": "Superman",
            "power": 2000
        },
        {
            "name":"Batman",
            "power": 1000
        },
    ]
}
```

```
<heroes>
    <hero>
        <name>Superman</name>
        <power>2000</power>
    </hero>
    <hero>
        <name>Batman</name>
        <power>1000</power>
    </hero>
</heroes>
```

JSON

- ✓ JSON.parse(str) – parsează stringul.
- ✓ JSON.stringify(obj) – creează un JSON din obiect.

JSON

JSON vs XML avantaje:

- ✓ e mai ușor de citit de către programator
- ✓ Conține mai puțină informație suplimentară față de cea utilă, ceea ce are următoarele implicații:
  - ✓ se transmite mai puțină informație prin rețea și eficiența transmisiunilor crește
  - ✓ necesită putere de procesare/memorie mai mică, ceea ce îl face mai potrivit pentru dispozitivele mobile care dispun de resurse limitate
- ✓ format nativ JavaScript - poate fi folosit ca atare de aplicațiile Javascript care rulează în browser și nu numai (vezi cazul AJAX)

JSON

JSON vs XML dezavantaje:

- ✓ sintaxa este mai simplă, suportand doar câteva tipuri de date (pe când în XML se pot crea tipuri de date la nevoie)
- ✓ la decodarea unui obiect JSON nu se cunosc informații despre variabila de origine, pe baza careia a fost generat (array asociativ/ obiect) ,în XML avem numele de elemente ce indică semnificația informației
- ✓ JSON nu beneficiază încă, în mod standardizat, de diversele tehnologii conexe disponibile de mult timp în XML