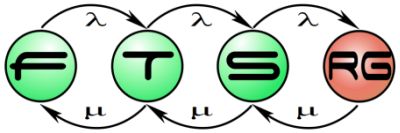
Adat reprezentáció

Szolgáltatás integráció előadás

Huszerl Gábor (BME MIT)



Kérdések

- Hogyan írhatóak le adatok több szolgáltatás számára is érthetően?
  - Alkalmazás- és platformfüggetlen módon
  - Miért nem kell minden szolgáltatás mellé külön értelmezőt (parser) írni?
    - Hogyan működhet egy értelmező?
  - Hogyan írható le az adatok nyelve?
- Nem lehet ezt másképp?
  - Az XML -lel csak a macera van (írás/olvasás)

Tartalom

- Nyílt adatformátumok
- XML
  - DTD
  - XSD
  - XML feldolgozása
    - DOM, SAX, JAXB, ...
- JSON

Nyílt adatformátumok

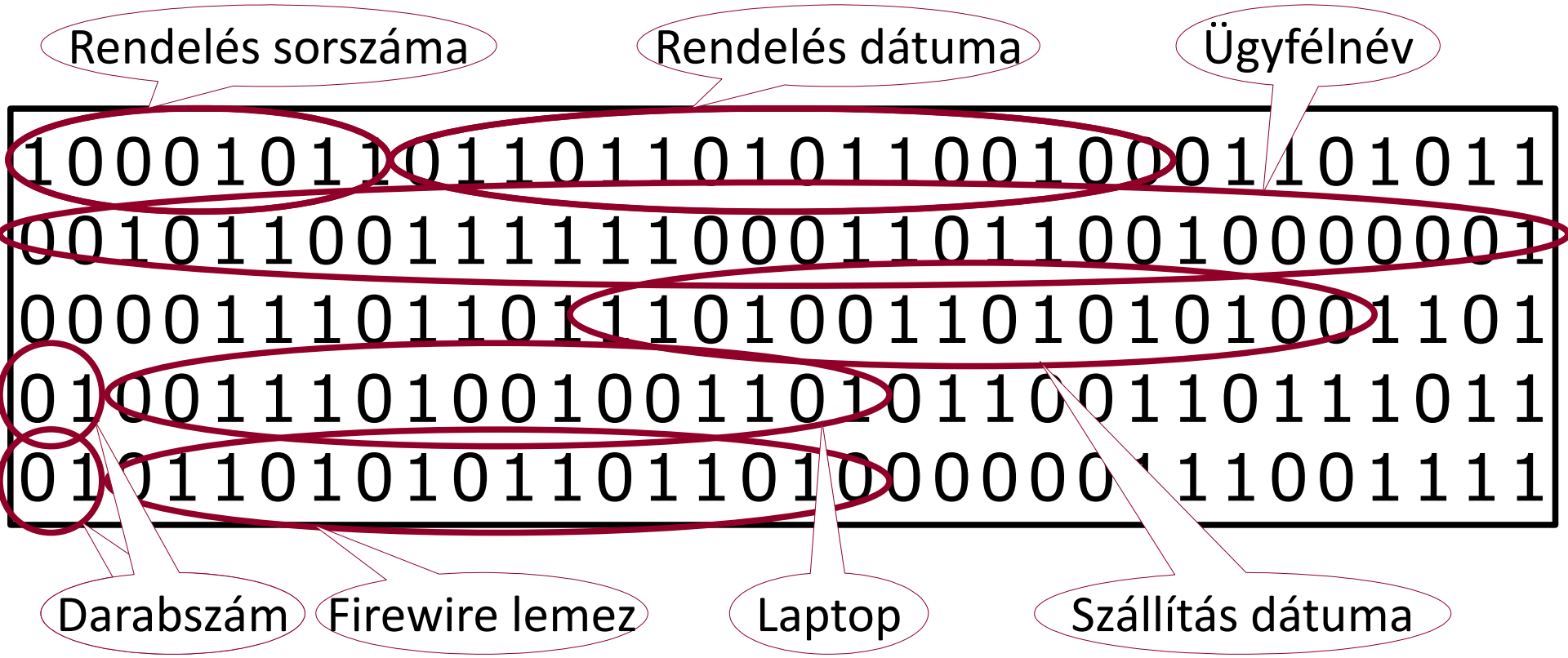
- Rendelkezési jog: Kié a tartalom?

- Nyílt adatformátum esetén
  - szerző (bármikor, később is)
  - tulajdonosok (bárki, esetleg vásárlás nélkül is)
- Védett adatformátum esetén
  - formátum tulajdonosa társtulajdonos
  - szerző/tulaj. jogai elvesznek formátumváltásnál
  - kompatibilitási problémák (verziók!)
  - cégfüggőség, támogatásfüggőség

Nyílt adatformátumok példa

```
10001011011011010110010001101011
00101100111111000110110010000001
00001110110111010011010101001101
01001110100100110101100110111011
01011010101101101000001110011111
```

Nyílt adatformátumok példa



- A szöveges formátumok legalább olvashatóak
  - Hibakeresésnél, kézi feldolgozásnál jól jön

Nyílt adatformátumok példa

```
ISA*  *       * *       *ZZ*SENDER
   *ZZ*RECEIVER
   *041201*1200*U*00305*000000101*1*P*^!
 GS*PO*SENDER*RECEIVER*041201*1200*101*X*003
   050!
    ST*850*000000101!
      BEG*22*NE*101**041201*123456!
      FOB*DF*ZZ*JMJ!
      DTM*037*041205!
      DTM*038*041215!
      DTM*002*041218!
      TD1*CNT90*1!
      TD5****JJ*X!
      TD3*40!
      N1*OB**92*7759!
        N3*111 Buyer St!
        N4*Conyers*GA*30094*US!
      N1*SE*Foo Bar Sellers!
        N4****US!
        REF*DP*101!
      PO1*100*1*EA***ZZ*BL47*HD*100!
        PID*F****Widget!
        PO4**1*BC!
        N1*ST**9!
          N4****US!
      CTT*1*100!
    SE*22*000000101!
  GE*1*101!
 IEA*1*000000101!
```

EDI X12 példa
(szegmensekre tördelve
„*” szeparátorral)

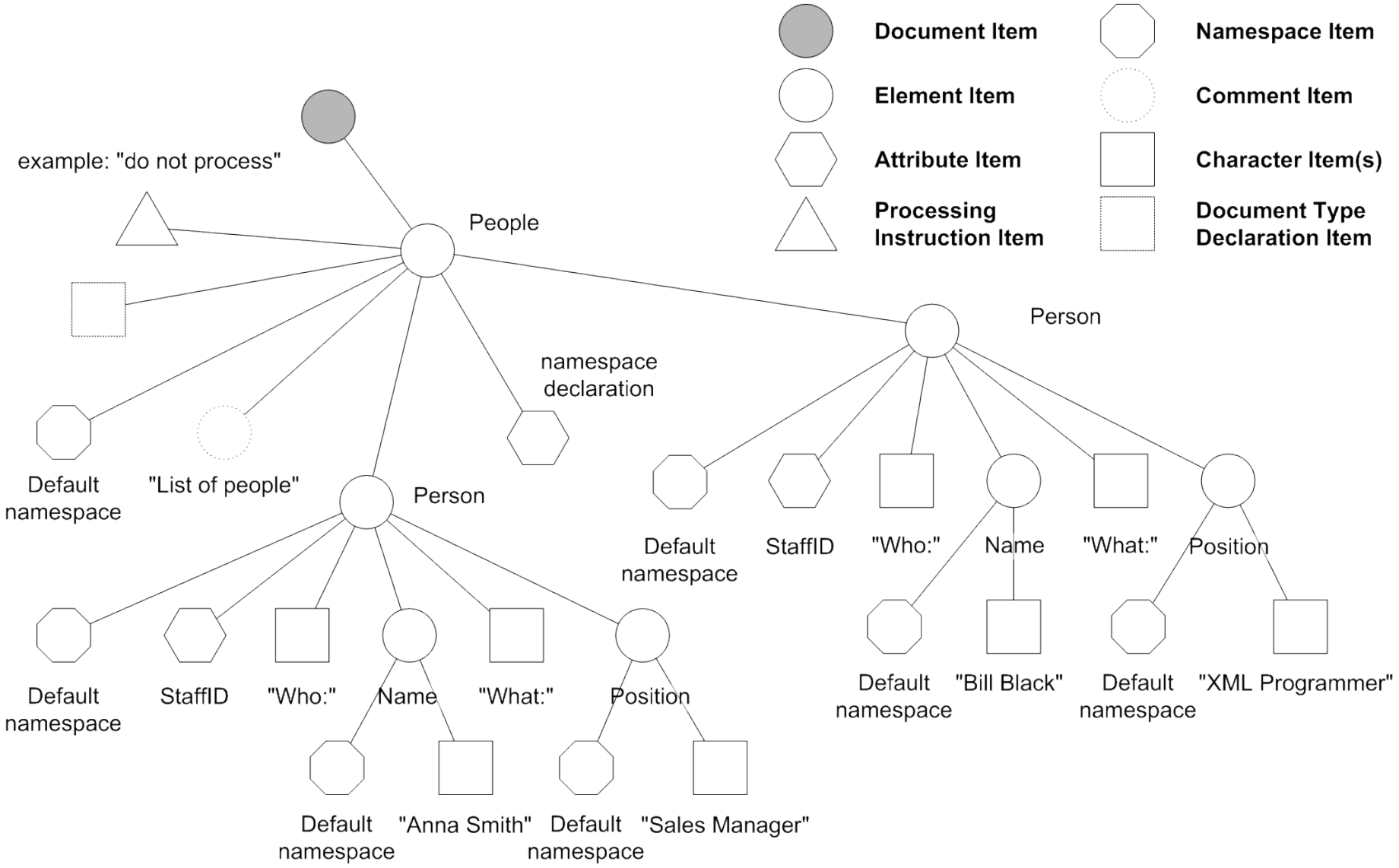
Tartalom

- Nyílt adatformátumok
- XML
  - DTD
  - XSD
  - XML feldolgozása
    - DOM, SAX, JAXB, ...
- JSON

Extensible Markup Language

- Kiterjeszthető jelölőnyelv
- Önleíró, „megjelenés helyett belső struktúra”, szabadon definiálható elemnevek, ...
- Előzmények
  - GML (1969), IBM (C. Goldfarb, E. Mosher, R. Lorie)
  - SGML (1985), ISO
  - HTML (1993), CERN
  - XML
    - 1.0 (1998), W3C
    - 1.0 – 2. edition (2000), W3C
    - 1.0 – 3. edition (2004), W3C
    - 1.1 (2004), W3C

XML Information Set



XML Infoset szintaktika

- Elemnevek
  - Betű, szám, egyéb karakter lehet benne
  - Kis- és nagybetűk különböznek
  - Nem kezdődhet számmal, ponttal, vesszővel, …
  - Nem kezdődhet: xml, XML, Xml, …
  - Nem lehet benne szóköz (aláhúzás igen)
  - Kettőspont kerülendő (névterek!)
- Bármely elemnek lehet attribútuma
  - Az adatokat az elemek hordozzák
  - A metaadatokat az attribútumok

Karakteres sorosítás szintaktikája

- Első sor: `<? xml version=”1.1” encoding=”UTF-8” ?>`
- Fa mélységi bejárása
  - Címkék párban, egymásba ágyazva
  - Pontosan egy gyökér elem
- Attribútum értékek idézőjelben
- Megjegyzés: `<!-- megjegyzés -->`
- „Whitespaces” (elvileg minden karakter karakter)
  - `xml:space`
- Soknyelvűség (`xml:lang`) RFC-1766
  - `hu-HU, de-DE, de-CH, de-AT, de-LU, de-LI`
    `en-GB/US/CA/AU/NZ/IE/ZA/JM/CB/BZ/TT/ZW/PH`
    `en-cockney`

XML feldolgozás

- CDATA (karakteres adat, az értelmező ne dolgozza fel)
  - Speciális karakterek, sortörések, teljes XML, szabálytalan XML, szabálytalan kódolás
- Entitások
  - Pl: `<`, `&`, `&quot;`, `<`, `<`, `<`,
  - Információ szimbolikus reprezentációja
    - tiltott karakter helyett vagy makrószerűen
  - DTD-ben definiálható: `!ENTITY`
- Feldolgozási utasítások (PIs)
  - Szintaktika: `<?` … `?>`
  - Van neve, az értelmező végrehajtja, ha érti

XML validálás

- Jó dokumentum (well formed)
    - dokumentum megfelel az XML szintaktikai előírásainak
- Érvényes dokumentum (valid)
    - dokumentum jó, és megfelel az adott nyelvtannak

Tartalom

- Nyílt adatformátumok
- XML
  - DTD
  - XSD
  - XML feldolgozása
    - DOM, SAX, JAXB, ...
- JSON

Document Type Definition

- Dokumentumtípus definíció, nyelvtan leírása
- Adott XML nyelv dokumentumainak felépítése
- Több szerző vagy dokumentum közös nyelve
- Formális leírás → automatikus validáció, automatikus értelmezés
- Fájlba épített DTD
  - <!DOCTYPE *root-element* [*element-declaration*]>
- Külön DTD fájl
  - <!DOCTYPE *root-element* SYSTEM *"filename"*]>
  - Az elemdeklarációk mennek a fájlba

DTD példa: elemek

Típusok:

```
<!ELEMENT element-name EMPTY>
<!ELEMENT element-name (#PCDATA)>
<!ELEMENT element-name ANY>
<!ELEMENT element-name (#PCDATA|...|...)*>
<!ELEMENT element-name (child-element-name)>
<!ELEMENT element-name (c-e-n1, c-e-n2, c-e-n3, c-e-n4, ...)>
<!ELEMENT element-name (c-e-n1|c-e-n2|c-e-n3|c-e-n4|...)>
```

Számosság:

```
<!ELEMENT note (body)>     1
<!ELEMENT note (body+)>    1-n
<!ELEMENT note (body?)>    0-1
<!ELEMENT note (body*)>    0-n
```

DTD példa: attribútumok

<!ATTLIST element-name attribute-name attribute-type default-value>

- Szabad névválasztás
- Attribútum típusok
  - CDATA, ID, IDREF, IDREFS, NMTOKEN, NMTOKENS, ENTITY, ...
- Alapértelmezések
  - *value*, #DEFAULT *value*, REQUIRED, IMPLIED, FIXED *value*

Tartalom

- Nyílt adatformátumok
- XML
  - DTD
  - XSD
  - XML feldolgozása
    - DOM, SAX, JAXB, ...
- JSON

XML Schema Definition

- Dokumentumok struktúrája, nyelvtan
- DTD utóda, XML alapú nyelv
- Formális leírás → automatikus validáció, ...
- Sémák hivatkozhatnak egymásra
  - Öröklések, szűkítések, bővítések, névterek!!!
- Gazdag típus rendszer, számosság jelölés

anySimpleType → date/time/dateTime/duration,
gDay/gMonth/gYear/gMonthDay/gYearMonth,
float, double, boolean, hexBinary, base64Binary,
string → normalizedString → token → decimal →
integer → nonPositiveInteger → negativeInteger
long → int → short
nonNegativeInteger → positiveInteger → unsignedLong...

XSD példa: XML fájl

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

<note xmlns=”www.w3schools.com”
    xmlns:xsi=”www.w3.org/2001/XMLSchema-instance”
    xsi:schemaLocation=”www.w3schools.com/note.xsd”>

    <to>Békeffi</to>
    <from>Lajtai</from>
    <heading>Üzenet </heading>
    <body>Hétre ma várom a Nemzetinél</body>

</note>
```

XSD példa: XSD fájl

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

<xs:schema xmlns:xs=”www.w3.org/2001/XML-Schema”
   targetNamespace=”www.w3schools.com”
   xmlns=”www.w3schools.com”
   elementFormDefault=”qualified”>

<xs:element name=”note”><xs:complexType><xs:sequence>

   <xs:element name=”to” type=”xs:string” />
   <xs:element name=”from” type=”xs:string” />
   <xs:element name=”heading” type=”xs:string” />
   <xs:element name=”body” type=”xs:string” />

</xs:sequence></xs:complexType></xs:element>

</xs:schema>
```

Séma elemek

- xs:element
  - (name, type, minOccurs, maxOccurs, ref, ...)
- xs:attribute (name, type, use, value)
  - use: `required, optional, prohibited, default`
- xs:simpleType (name, minOccurs, maxOccurs)
  - xs:restriction (base)
    - xs:maxExclusive (value), xs:minInclusive (value)
    - xs:pattern (value), xs:enumeration (value)
    - xs:list (itemType), xs:length (value)
  - xs:extension (base)
  - xs:union (memberTypes)

Séma elemek

- xs:complexType (name, minOccurs, maxOccurs)
  - xs:sequence, xs:choice, xs:all
- xs:unique (name)
  - xs:selector (kik között legyen egyedi)
  - xs:field (miben legyen egyedi)

```
<xs:element name=”root” type=”myList”>
   <xs:unique name=”myId”>
      <xs:selector xpath=”./a” />
      <xs:field xpath=”id” />
   </xs:unique>
</xs:element>
```

(a ”root”-on belül minden ”a” elemnek egyedi ”id”-ja legyen)

- xs:key, xs:keyref

Összetett séma definíciók

- include
  - Azonos célnévterű leírás(rész) beillesztése
- import
  - Másik névtér bevonása a leírásba
  - Pl. másik szabványban leírt nyelvtan használata
  - Névterek használata fontos!
- redefine
  - Leírások örököltetése, finomítása
  - xs:extension, xs:restriction segítségével

Tartalom

- Nyílt adatformátumok
- XML
  - DTD
  - XSD
  - XML feldolgozása
    - DOM, SAX, JAXB, ...
- JSON

Document Object Model

- Absztrakt interfészgyűjtemény (Dok. kezelő API)
- Dokumentum logikai modellje, struktúrája
  - Objektummodell
- „Dynamic HTML” utóda
  - Netscape, 1996, JavaScript támogatására
- XML dokumetumok értelmezése alapulhat rá
  - DOM alapú értelmezők
  - Az XML dokumetum modelljének létrehozása (validálás után) a memóriában, alkalmazás ezen dolgozik, majd kiírat (Nem maga!)
  - Minden csomóponthoz egy „Node” példány

Simple API for XML Parsing

- SAX alapú értelmezők
  - Értelmezés a dokumentum beolvasása nélkül
  - Eseményvezérelt, soros feldolgozás
    - „start document”, „end document”
    - „start element”, „end element”
    - „characters”
    - ezekhez eseménykezelőt kell írni
      (eldob / állapotot vált / megjegyez)
  - Gyors, helytakarékos, nehezen programozható
    - gyorsan indul, de csak előrefelé halad

Java API for XML Binding

- DOM tipikus problémája
  - Alkalmazás legtöbbször
    „Node” objektumokból üzletiekbe másol, majd vissza kiíratás előtt
- JAXB alapú elemzők
- Teljes dokumentum memóriába olvasása
  - Minden elemtípusnak saját osztály
    (abból példányosít, szemben a DOM-mal)
- Osztályok generálása a séma definícióból
  - Vagy fordítva

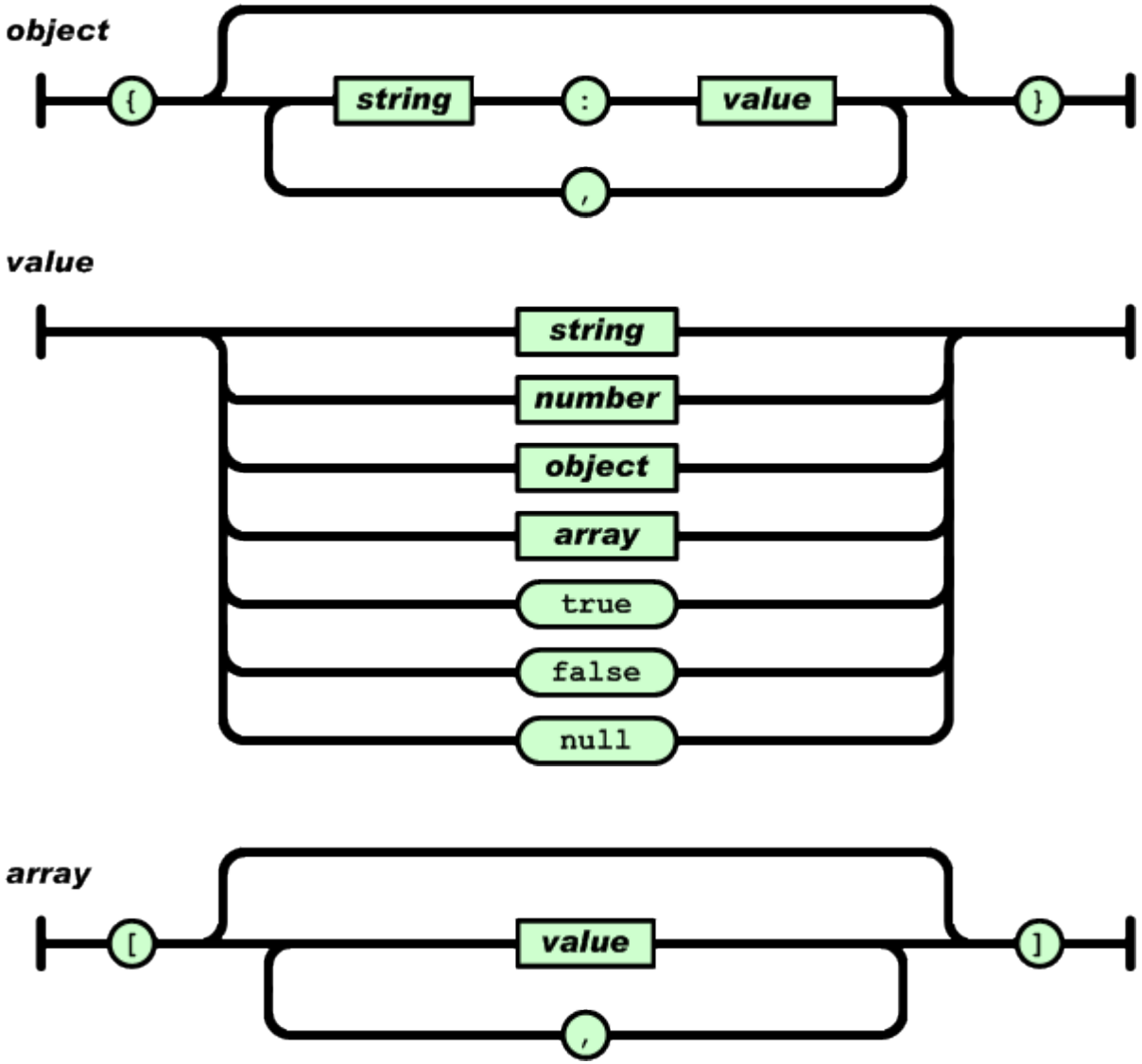
Tartalom

- Nyílt adatformátumok
- XML
  - DTD
  - XSD
  - XML feldolgozása
    - DOM, SAX, JAXB, ...
- JSON

JavaScript Object Notation

- Karakteres nyílt adatformátum
- Főleg JavaScript környezethez
- Szigorú fa struktúra, egyszerű feldolgozás
  - Kicsi, könnyű, olcsó
- Nincs nyelvtan definíció → nincs validálás
  - Nincsenek névterek, mert minek?
  - Semmi önleírás vagy hasonló dolog

JSON szintaktika



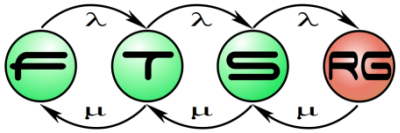
Tartalom

- Nyílt adatformátumok
- XML
  - DTD
  - XSD
  - XML feldolgozása
    - DOM, SAX, JAXB, ...
- JSON

Adat transzformáció

Szolgáltatás integráció előadás

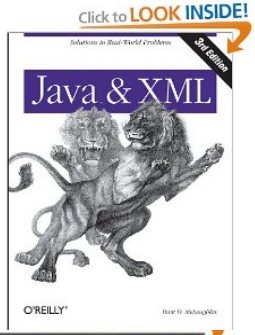
Guta Gábor (BME MIT)



Tartalom

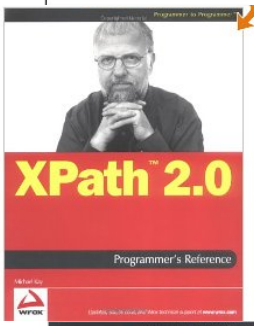
- XML Streaming API – rövid áttekintés

  Brett McLaughlin, Justin Edelson:
  Java and XML (O'Reilly)



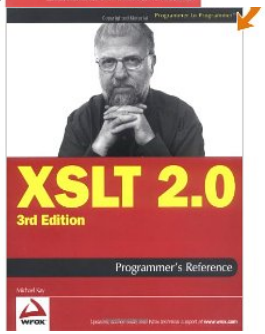
- XPath

  Michael Kay: XPath 2.0
  Programmer's Reference (Wrox)



- XSLT

  Michael Kay: XSLT 2.0
  Programmer's Reference (Wrox)



StAX

- Stream API XML-hez (SAX kifordítva)
- „pull API” — mi kérünk egy elemet
- Esemény: egy XML elemet elérünk/elhagyunk
- Igazából két API:
    - Cursor → egy *XMLStreamReader*
    - Event → az *XMLEventReader*től kapunk XMLEvent szekvenciát

+ ugyan erre a mintára tudunk kiírni is

StAX példa

```
import javax.xml.stream.*;

XMLInputFactory ifact=XMLInputFactory.newInstance();

XMLStreamReader reader=ifact.createXMLStreamReader(
                    new FileInputStream(filename));

// elso elemre mutat

int eventType=reader.getEventType();

// a tobbi elem olvasasa

while (reader.hasNext()){

      eventType=reader.next();

}

reader.close();
```

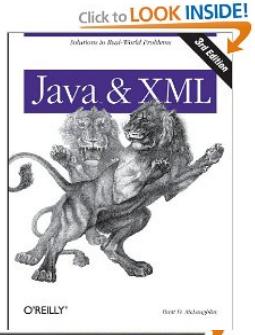
StAX példa

```
if (eventType==XMLEvent.START_ELEMENT) {
    System.out.println(reader.getName());

    System.out.println(reader.getAttributeCount());

    if (reader.getLocalName()=="table") {

        System.out.println(reader

            .getAttributeValue(null, "name"));

}
```

Tartalom

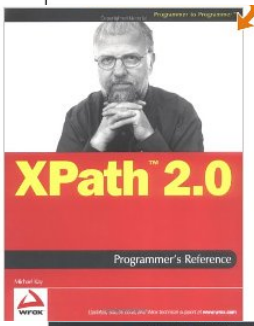
- XML Streaming API – rövid áttekintés

  Brett McLaughlin, Justin Edelson:
  Java and XML (O'Reilly)



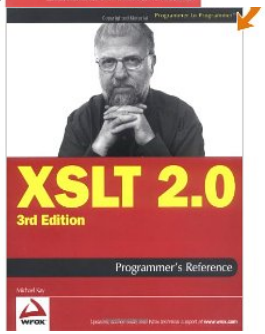
- XPath

  Michael Kay: XPath 2.0
  Programmer's Reference (Wrox)



- XSLT

  Michael Kay: XSLT 2.0
  Programmer's Reference (Wrox)



Példa XML

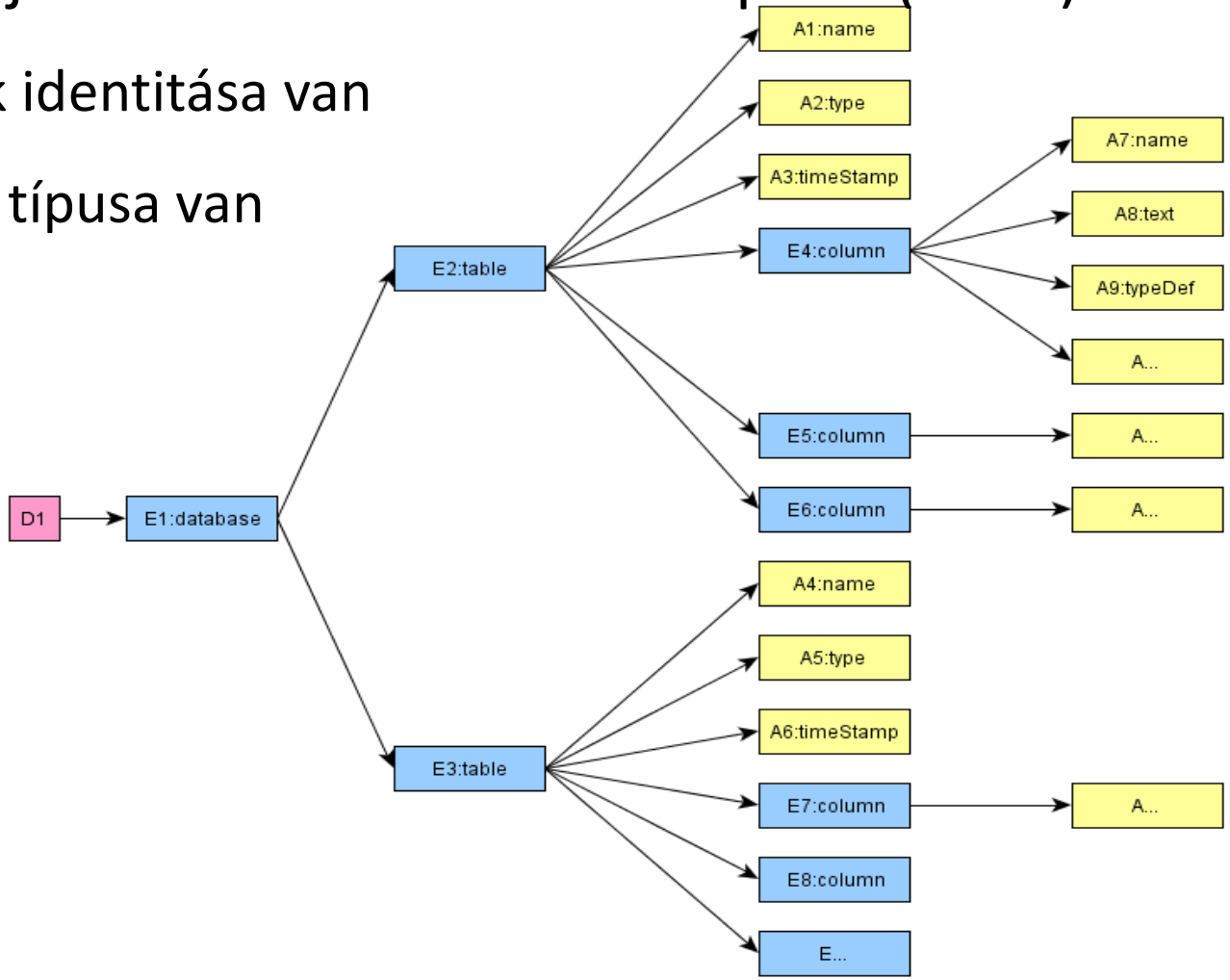
```
<?xml version="1.0" encoding="UTF-8"?>
<database>
  <table name="AdrAssignee" type="large"
      timeStamp="false">
    <column name="Number" text="Number"
        typeDef="midstr" visible="true" default=""
        searchable="true" grid="true"/>
    <column name="Name" text="Name"
        .....
```

XPath

- A szemantikája XML Data Modelen alapszik (XDM)

  - Az elemeknek identitása van

  - Az adatoknak típusa van



Terminológia: node=elem

XPath rövidítések

- Elérési utak a XML adatmodellben

  Az aktuális elem: /database/table[1]

column	Az aktuális elem összes column eleme
/database/table	A database elem összes table eleme
/database//column	Az összes column elem
.	Az aktuális elem
..	A database elem (szülő)
@name	A name attributum
*	Bármely elem
@*	Bármely attributum

XPath tengelyek

- Bonyolultabb pozíció leírásra alkalmasak

  Az aktuális elem: /database/table[1]

child::column	A table[1] összes column gyereke
/database/child::table	A database elem összes table gyereke
/database/descendant::column	Az összes column elem
self::*	Az aktuális elem (table[1])
parent::*	A szülő elem (/database)
ancestor-or-self::*	Aktuális elem és a felette lévők
preceding::*	Tőle balra lévők

- child, descendant, attribute, self, descendant-or-self, following-sibling, following, parent, ancestor, preceding-sibling, preceding, ancestor-or-self

XPath predikátumok

- Szűrőként működik
- Kiértékeli a kifejezést: ha igaz, akkor az adott elemet beleveszi az eredményekbe, ellenkező esetben nem

  Az aktuális elem: /database/table[1]

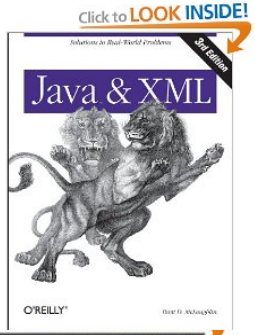
column[1]	Az első column elem
column[last()]	Az utolsó column elem
column[position() lt 5]	Az első 4 column elem (position() < 5)
column[@grid]	Van grid attribútuma
column[@grid='true']	Van grid attribútuma, és az 'true'

- Feltételek

Tartalom

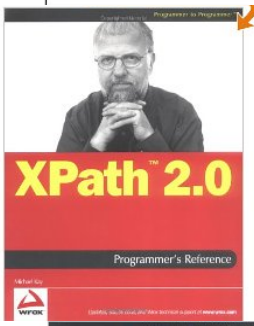
- XML Streaming API – rövid áttekintés

  Brett McLaughlin, Justin Edelson:
  Java and XML (O'Reilly)



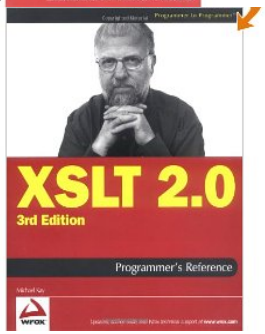
- XPath

  Michael Kay: XPath 2.0
  Programmer's Reference (Wrox)



- XSLT

  Michael Kay: XSLT 2.0
  Programmer's Reference (Wrox)



XSLT

- XSL két nagy szabvány:
    - XSLT (www.w3.org/TR/xslt20/)
    - *XSL-FO (www.w3.org/TR/xsl/)* itt most nem lesz
- Eredetileg az adatok reprezentációtól való elválasztására találták ki
  adat:XML, megjelenés:HTML, leképzés:XSLT
- Deklaratív transzformációs nyelv (leírás, végrehajtó motor)
- Egy XML dokumentum
- Sablonok halmazából áll
  + vezérlési utasításokból azokon belül

XSLT felépítése

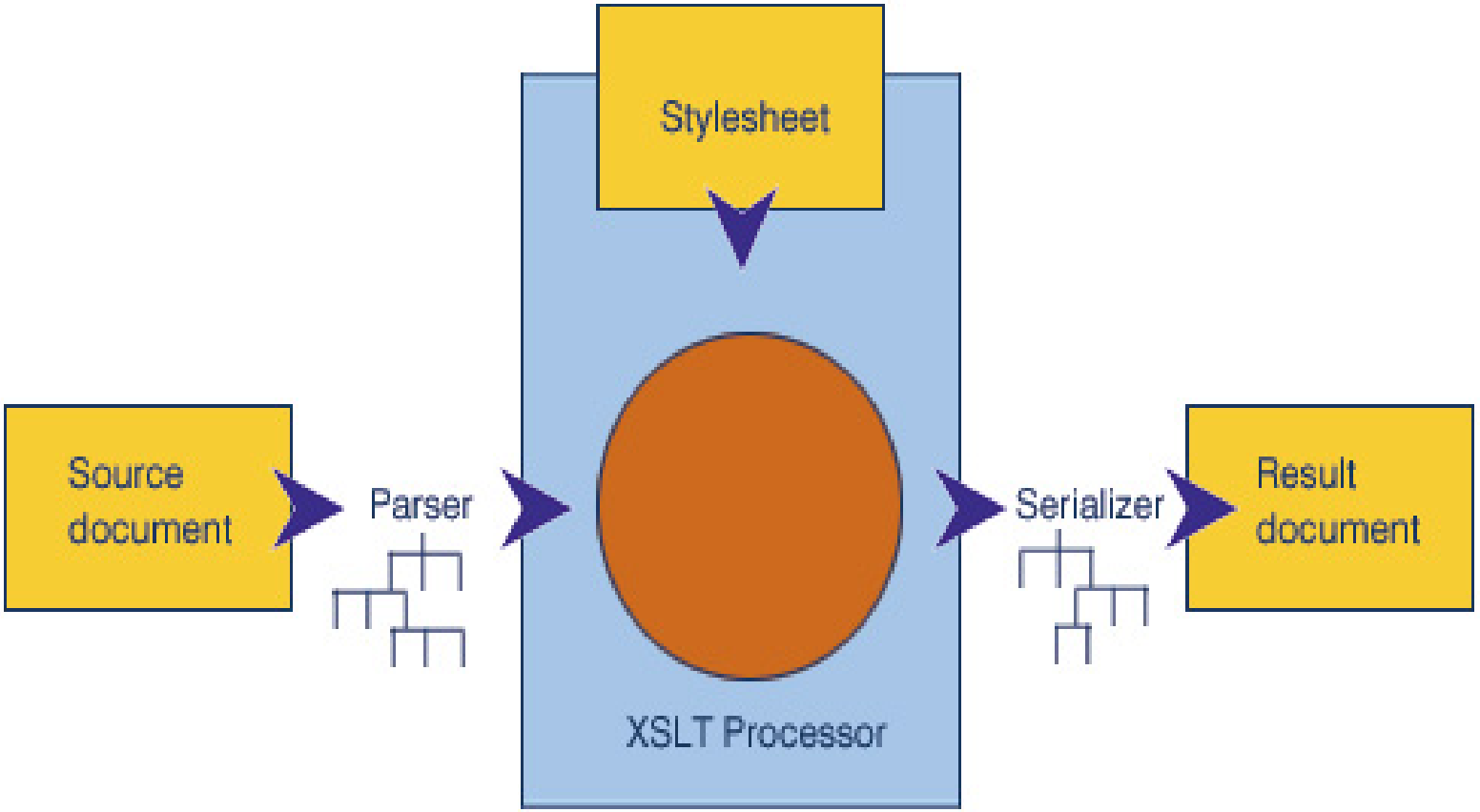
```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
  <xsl:template match="/">
    <html>
      <head><title>Table structures</title></head>
      <body>
        <xsl:apply-templates select="database/table"/>
      </body>
    </html>
  </xsl:template>
  <xsl:template match="table">
      <h2>Structure of the table <xsl:value-of select="@name"/></h2>
      <table>
       <tr><th>Name</th><th>Type</th></tr>
        <xsl:apply-templates select="column"/>
      </table>
  </xsl:template>
  <xsl:template match="column">
    <xsl:choose>
      <xsl:when test="@grid='true'">
        <tr>
          <td bgcolor="yellow"><xsl:value-of select="@name"/></td>
          <td bgcolor="yellow"><xsl:value-of select="@typeDef"/></td>
        </tr>
      </xsl:when>
      <xsl:otherwise>
        <tr>
          <td><xsl:value-of select="@name"/></td>
          <td><xsl:value-of select="@typeDef"/></td>
        </tr>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
</xsl:stylesheet>
```

Kiírás (value-of)

“Switch” szerkezet

Template

XSLT processzor



Source: http://www.ibm.com/developerworks/xml/library/x-xslt/

Szabály alkalmazási szabályok :)

- Aminek a fejléce megfelel (match + mode + priority)
- A leginkább specifikus
- Default, ha nincs más!!!

```
<xsl:template match="/">
   <xsl:apply-templates select="//table[1]" mode="first"/>
   <xsl:apply-templates select="//table[2]" mode="second"/>
</xsl:template>
<xsl:template match="table" mode="first">
      <p>Name of the first table is
         <xsl:value-of select="@name"/></p>
</xsl:template>
<xsl:template match="table" mode="second">
      <p>Name of the second table is
         <xsl:value-of select="@name"/></p>
</xsl:template>
```

XSLT vezérlési utasítások

- Feltételes

```
<xsl:if test="expression">
```

- Többszörös elágazás

```
<xsl:choose>
    <xsl:when test="expression">
            ...
    <xsl:otherwise>
            ...
```

- Ciklus

```
<xsl:for-each select="expression">
```

XSLT vezérlési utasítások

- Sablon hívás

```
<xsl:template match="/">
   <xsl:call-template name="named_template">
      <xsl:with-param name="arg1" select="42">
   </xsl:call-template>
</xsl:template>

<xsl:template name="named_template">
  <xsl:param name="arg1" />
</xsl:template>
```

XSLT változók

- Egyszer kapnak csak értéket (nem-változók)
- Változó deklaráció

  ```
  <xsl:variable name="i" select="'whatever'" />
  ```

- Érték kiíratás

  ```
  <xsl:value-of select="$i" />
  ```

- Részfa átvétele az inputról

  ```
  <xsl:copy-of select="//column" />
  ```

XSLT extrák

- A kimenti formátum (XML, HTML, Text)

```
<xsl:output method="text" encoding="UTF-8"/>
```

- Lehet külön fájlokba írni, aminek a nevét pl. a forrás XML-ből tudjuk felszedni

```
<xsl:result-document href="'filename'" format="text">
```

- Attribútumok hozzáadása értékkel

```
<xsl:attribute name="name">
  <xsl:value-of select="@name"/>
</xsl:attribute>
```

- Külső fájlok:

```
<xsl:include href="modul.xsl"/>
```

XSLT futtatása

- Parancssorból:

  java -jar saxon9he.jar -t -s:input.xml
  -xsl:transformation.xsl -o:output.xml
  -explain Megmutatja az XSLT szerkezetét
  -T Megmutatja mi futott le az XSLT-ből

- API-ból: JAXP, MSXML6 (1.0)!, Saxon

- XSLT 2.0: Oracle XDK, AltovaXML
- XSLT 1.0: Xalan, tDOM, etc.