

# XML feldolgozás

# Áttekintés

- XML -bevezetés
- (XML érvényességének vizsgálata (DTD, XSD))
- XML feldolgozók (DOM, SAX)
- XML transformációk (XSLT)

# Áttekintés

- XML -bevezetés
- (XML érvényességének vizsgálata (DTD, XSD))
- XML feldolgozók (DOM, SAX)
- XML transformációk (XSLT)

# Áttekintés

- XML -bevezetés
- (XML érvényességének vizsgálata (DTD, XSD))
- XML feldolgozók (DOM, SAX)
- XML transformációk (XSLT)

# Áttekintés

- XML -bevezetés
- (XML érvényességének vizsgálata (DTD, XSD))
- XML feldolgozók (DOM, SAX)
- XML transformációk (XSLT)

# XML

- XML jelentése **EX**tensible **M**arkup **L**anguage (kiterjeszhető jelölő nyelv).
- főként adatrepresentációra alkalmas
- platformfüggetlen- illetve alkalmazástól független adatcserét tesz lehetővé
- a tag-ek nincsenek előre meghatározva
- az XML állományban tárolt adat szerkezete leírható DTD (Document Type Definition) vagy egy XML séma (XSD) segítségével
- új nyelv is definiálható a segítségével (lásd: WML)
- a W3C ajánlása (1998 február)

## különbség a HTML-hez képest

- HTML célja - az adatok megjelenítése
- XML célja - az adatok (információ) leírása

# XML szintaxis

- egyszerű, szigorú szintaktikai szabályok

## XML deklaráció

- az első sor megadja az XML állomány verziószámát, illetve a használt kódolást:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
```

Pl.

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<laborhazi sorszam="1">
  <a>ez a hazi a pontja</a>
  <b>ez pedig a b pont</b>
</laborhazi>
```

## szintaktikai szabályok:

- minden XML elemnek kell legyen záró tag-je
- XML tag-ek esetén számít a kis- vagy nagybetű
- az XML tag-eket helyesen kell egymásba ágyazni (nem lehetnek egymásba ékelve)
- az XML dokumentumnak egy és csakis egy gyöker eleme lehet
- az attribútumok értékeit kötelező idézőjelbe (" vagy ') tenni
- a fehér karakterek nincsenek figyelmen kívül hagyva
- újsor kódolása: LF
- <!-- – így néz ki egy XML megjegyzés – – >

- elemek közti viszonyok: szülő, gyerek, testvér (sibling)

## elem felépítése:

- kezdő tag, törzs, záró tag (lehet üres is <tagnev ... /tagnev>)
- az elemnek lehetnek attribútumai



# XML névterek (Namespaces)

- a névterek a különböző XML állományokból származó, azonos nevű elemek esetében a névkonfliktus feloldására szolgálnak

## Pl. - konfliktus

<code>&lt;ar&gt;</code>	<code>&lt;ar&gt;</code>
<code>  &lt;penznem&gt;Euro&lt;/penznem&gt;</code>	<code>  &lt;ev&gt;2007&lt;/ev&gt;</code>
<code>  &lt;ertek&gt;200&lt;/ertek&gt;</code>	<code>  &lt;vizszint&gt; 7.5m &lt;/vizszint&gt;</code>
<code>&lt;/ar&gt;</code>	<code>&lt;/ar&gt;</code>

## Konfliktus feloldása előtag (prefix) használatával

<code>&lt;p:ar&gt;</code>	<code>&lt;v:ar&gt;</code>
<code>  &lt;p:penznem&gt;Euro</code>	<code>  &lt;v:ev&gt;2007&lt;/v:ev&gt;</code>
<code>&lt;/p:penznem&gt;</code>	<code>  &lt;v:vizszint&gt; 7.5m</code>
<code>  &lt;p:ertek&gt;200&lt;/p:ertek&gt;</code>	<code>&lt;/v:vizszint&gt;</code>
<code>&lt;/p:ar&gt;</code>	<code>&lt;/v:ar&gt;</code>

## Konfliktus feloldása névtér használata segítségével

```
<p:ar xmlns:p="http://www.pl.ro/pl1/" >  
  <p:penznev>Euro</p:penznev>  
  <p:ertek>200</p:ertek>  
</p:ar>
```

```
<v:ar xmlns:v="http://www.pld2.ro/pl2/" >  
  <v:ev>2007</v:ev>  
  <v:vizszint> 7.5m </v:vizszint>  
</v:ar>
```

## egy elem kezdő tag-jébe helyezett xmlns attribútum:

- szintaxisa: xmlns:namespace-prefix="namespaceURI"
- az illető elembe ágyazott összes elem, melynek ugyanaz a prefix-e, ugyanahhoz a névtérhez fog tartozni
- az URI csupán egy egyedi nevet rendel a névterülethez
- alapértelmezett névtér (prefix nélkül): xmlns="namespaceURI"

# Jól formált illetve érvényes XML

- egy XML dokumentum *jól formált* (well formed): ha megfelel az XML szintaktikai szabályainak
- egy XML dokumentum *érvényes* (valid): ha jól formált, és megfelel a dokumentum séma definíciójának (pl. egy bizonyos DTD-ben vagy XML sémában –XSD– megadott szabályok)

# DTD

## DTD – Document Type Definition

- elterjedt séma-leíró módszer
- az XML dokumentum érvényes építőelemeit adja meg (elemek, attribútumok), illetve ennek felépítését
- szabványos, de nem XML alapú, ezért népszerűsége valószínűleg csökkenni fog

## megadható:

- az XML állományon belül (ritkábban használt):  
    <DOCTYPE gyoker-elem [elem-deklaraciok]>
- külön dtd kiterjesztésű állományban (lásd pl. a web.xml szerkezetét leíró DTD)

## egy XML állomány alkotóelemei a DTD sémaleíró szempontjából:

- Elemek
- Attribútumok
- Egyedek (Entities) – speciális karakterkódok  
(pl. &lt; – <, &amp; – &, &quot; – ”, &apos; – ’)
- PCDATA – parsed character data
- CDATA – character data

## elem deklarációja:

```
<!ELEMENT elem-nev categoria>
```

vagy

```
<!ELEMENT elem-nev (elem-tartalom)>
```

- üres elem: `<!ELEMENT elem-nev EMPTY>`
- csak (feldolgozott) szöveget tartalmazó elem:  
`<!ELEMENT elem-nev (#PCDATA)>`
- bármilyen tartalommal rendelkező elem:  
`<!ELEMENT elem-nev ANY>`
- beágyazott (gyerek) elemeket tartalmazó elemek:  
`<!ELEMENT elem-nev (gyerek1)>`  
vagy  
`<!ELEMENT elem-nev (gyerek1,gyerek2,...)>`
- gyerek egyszeri előfordulása: `<!ELEMENT elem-nev (gyerek-nev)>`
- gyerek legalább egyszeri előfordulása:  
`<!ELEMENT elem-nev (gyerek-nev+)>`
- ... nulla vagy többszöri előfordulásnál: \*, nulla vagy egyszeres előfordulásnál: ?
- ... felsorolásban szereplő "vagy": |

## attribútum deklaráció:

szintaxis: `<!ATTLIST elem-nev attributum-nev attributum-típus alapertelmezett-érték>`

## attributum-típus lehetséges értékei:

- CDATA, felsorolás: (en1| en2 | ...), ID, IDREF, IDREFS, NMTOKEN, NMTOKENS, ENTITY, ENTITIES, NOTATION, xml:

## alapertelmezett-érték lehetséges értékei:

- *érték*, #REQUIRED, #IMPLIED, #FIXED *érték*

## egyedek (entities)

- belső egyed deklaráció `<!ENTITY egyed-nev "egyed-érték">`  
használata XML állományban: `& egyed-nev;`
- külső egyed deklaráció:  
`<!ENTITY entity-name SYSTEM "URI/URL">`

# XML séma definíció

- XSD – XML Schema Definition
- a DTD-nek XML alapú alternatívája

## egy XML séma:

- meghatározza, hogy milyen elemek szerepelhetnek egy dokumentumban
- milyen attribútumok szerepelhetnek egy dokumentumban
- milyen beágyazott (gyerek) elemek vannak
- meghatározza a gyerek elemek előfordulásának sorrendjét
- meghatározza a gyerek elemek számát
- meghatározza, hogy egy elem üres-e vagy tartalmazhat szöveget
- az elemek illetve attribútumok típusa is definiálható
- megadható az elemek illetve attribútumok alapértelmezett illetve rögzített értéke



## előnyei a DTD-vel szemben:

- rugalmasabb, bővebb
- XML alapú
- megadható az adatok típusa
- névterek használata

## XML séma gyökér eleme:

```
<?xml version="1.0"?>  
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  
targetNamespace="http://www.pltarget.com"  
xmlns="http://www.pltarget.com"  
...  
</xs:schema>
```

## a schema elem attribútumai (sorrendben):

- a sémában használt, előre definiált elemek és adattípusok névtére, illetve a prefix (xs), amivel használva lesznek
- ebben a sémában definiált elemek névtére
- alapértelmezett névtér (vagyis az itt definiált elemeket prefix nélkül adjuk meg)

## Hivatkozás a sémára egy XML állományból:

```
<?xml version="1.0" ?>  
<ar xmlns="http://www.pl.ro/pl1/"  
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xsi:schemaLocation="http://www.pl.ro/pl1/ arak.xsd" >  
  <penznem>Euro</penznem>  
  <ertek>200</ertek>  
</ar>
```

## egyszerű elem definíciója:

```
<xs:element name="xx" type="yy" />
```

- beépített típusok: xs:string, xs:decimal, xs:integer, xs:boolean, xs:date, xs:time

- egyszerű elemnek lehet alapértelmezett (default="..." – attribútum) vagy rögzített (fixed="...") értéke

## attribútum definíciója:

```
<xs:attribute name="xx" type="yy" />
```

- az attribútumnak is lehet alapértelmezett (default="..." – attribútum) vagy rögzített (fixed="...") értéke
- ha kötelező: use="required"

## Megszorítások (facets)

- megadhatjuk az elemek illetve attribútumok elfogadható értékeit megszorítások (facet) formájában

### Pl. min. max. érték:

```
<xs:element name="ev" >
  <xs:simpleType>
    <xs:restriction base="xs:integer" >
      <xs:minInclusive value="1990" />
      <xs:maxInclusive value="2007" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

## más megszorítások:

- felsorolás: `<xs:enumeration value="ertek" />`
- reguláris kifejezés: pl. `<xs:pattern value="[a-z]" />`
- fehér karakterekre vonatkozó megszorítások:  
`<xs:whiteSpace value="preserve|replace|collapse" />`
- hosszra vonatkozó megszorítások: `<xs:length value="6" />`  
(minLength, maxLength)
- stb.

- az egyszerű típushoz rendelhetünk nevet (name attribútum), ekkor elérhető lesz más elem számára is

## Összetett (komplex) elemtípus:

- más beágyazott elemeket és/vagy attribútumokat is tartalmazó elemek

## összetett elem-típusok:

- üres (törzs nélküli) elemek
- csak más beágyazott elemeket tartalmazó elemek
- csak szöveget tartalmazó elemek
- szöveget, illetve más elemeket is tartalmazó elemek

– mindenkinek lehet attribútuma is

- `<any>` illetve `<anyAttribute>` elemek segítségével kibővíthetővé tehetjük a dokumentumot

## Összetett elem definíciója (pl.):

```
<xs:element name="ar" >
  <xs:complexType>
    <xs:sequence>
      <xs:element name="penznem" type="xs:string" />
      <xs:element name="ertek" type="xs:integer" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

- a sequence elem meghatározza, hogy a felsorolt beágyazott elemek a megadott sorrendben kell szerepeljenek
- itt is rendelhetünk nevet az összetett típushoz
- már definiált összetett típus kiterjeszhető további elemekkel, illetve attribútumokkal

# Indikátorok

## sorrendet meghatározó indikátorok:

- All – a beágyazott elemek bármilyen sorrendben előfordulhatnak
- Choice – egyik vagy másik
- Sequence – az elemek csak a megadott sorrendben fordulhatnak elő

## előfordulást meghatározó indikátorok:

- maxOccurs
- minOccurs

## csoport indikátorok:

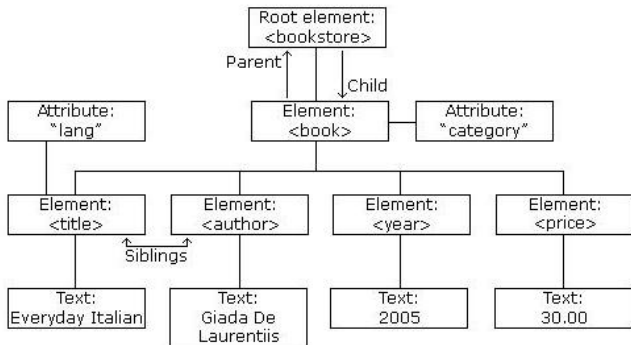
– csoportosíthatjuk az egyes elemeket illetve attribútumokat:

- Group name
- attributeGroup name



## XML feldolgozók (parser) – DOM

- XML DOM: XML Document Object Model – az XML dokumentumok feldolgozásához biztosít egy standard API-t.
- a DOM az XML dokumentumot egy fa-szerkezet formájában ábrázolja, melynek csomópontjai az elemek, attribútumok, illetve szövegrészek.



## jellemzők:

- az XML DOM (Document Object Model for XML) – objektum modellt definiál az XML dokumentumhoz
- az XML DOM platform- illetve nyelvfüggetlen
- az XML DOM standard hozzáférésmódot biztosít az XML dokumentumokhoz (olvasás, módosítás)
- az XML DOM W3C standard

## hozzáférés az egyes csomópontokhoz:

- `getElementsByTagName("tag-nev")` metódus segítségével – csomópontok listáját téríti vissza
- `parentNode`, `firstChild`, `lastChild` mezőket használva
- gyökér elem: `document.documentElement`

információ az illető csomópontról az alábbi mezőkben:

- nodeName
  - nodeValue
  - nodeType
- 
- egy elem *attributes* mezője az attribútumokat tartalmazza map formájában (NamedNodeMap)

lásd pl.

DomParserExample, XMLCreatorExample

# XML feldolgozók (parser) – SAX

## SAX – Simple API for XML:

- XML dokumentumok szekvenciális feldolgozására szolgáló API.
- a DOM egy igen elterjedt alternatívája
- a DOM-tól eltérően nincs neki megfelelő formális specifikáció (a Java implementációt tekintik iránymutatónak a SAX-ot implementáló többi platformok)

## SAX feldolgozó (parser)

- Egy SAX-ot implementáló feldolgozó *adattolyam feldolgozó*ként (parser) működik, eseményvezérelt API-val.
- Az egyes eseményekre a felhasználó által definiált ún. “callback”-metódusokat fogja meghívni a feldolgozó.
- A feldolgozás egyirányú: a már feldolgozott adatot nem lehet újraolvasni (csak ha újratekdjük a feldolgozást)

# SAX

## SAX események:

Egy-egy esemény generálódik az alábbi elemek feldolgozásának kezdetén illetve végén:

- szöveget tartalmazó XML csomópontok (Text nodes)
- XML elem csomópontok
- XML feldolgozó utasítások
- XML megjegyzések

az attribútumok az elemet feldolgozó megfelelő callback metódus paramétereként lesznek elérhetőek

### előnyök a DOM-al szemben:

- lényegesen kevesebb memóriát vesz igénybe, mint a DOM (ennek esetében a teljes fastruktúrát tartalmazó objektum betöltődik a memóriába)
- gyorsabb feldolgozást tesz lehetővé
- ha a dokumentum mérete akkora, hogy a neki megfelelő objektum nem férne be a memóriába, akkor DOM-al nem dolgozható fel

### hátrányok a DOM-al szemben:

- SAX-al nem tudjuk módosítani/lementeni a forrás XML állományt, csak szekvenciálisan feldolgozni
- ezzel szemben DOM-al bármikor hozzáférhető bármelyik csomópont, módosítható, illetve file-ba lementhető

lásd pl.

SAXParserExample

# XSL

- XSL – **EX**tensible **S**tylesheet **L**anguage
- XML alapú stílusállomány
- egy XSL állomány leírja, hogy az XML dokumentumot hogyan kell megjeleníteni
- AZ XSL több, mint stílus-leíró nyelv

az XSL három részből áll:

- XSLT - az XML dokumentumok transzformálására szolgáló nyelv
- XPath - XML dokumentumok bejárására, lekérdezésére szolgáló nyelv
- XSL-FO - XML dokumentumok formázására szolgáló nyelv

# XPath

az XPath - az XML dokumentumok bejárását, adatok lekérdezését lehetővé tevő nyelv.

## az XPath jellemzői:

- az XPath egy szintaxist ad az XML dokumentumok egyes részeinek meghatározására
- kifejezéseket használ az XML dokumentum bejárására
- tartalmaz egy standard függvénykönyvtárat
- az XPath is W3C standard



## XPath elérési utat megadó kifejezések (Path Expressions):

- az XPath kifejezéseket használ az egyes csomópontok vagy csomópont-halmazok elérésére/kiválasztására

## XPath standard függvények

- több, mint 100 beépített függvényt használ (string-értékek, numerikus adatok, dátum, idő feldolgozása, stb.)

## csomópont-típusok:

elem, attribútum, szöveg, névtér, feldolgozó utasítás, megjegyzés, document (gyökér) csomópont.

# XSLT

Az XSLT az XML dokumentumok XHTML vagy más XML dokumentummá való transzformálását teszi lehetővé.

## XSLT jellemzői:

- XSLT – XSL Transformations (XSL átalakítások)
  - az XSL legfontosabb része
  - XPath-ot használ az XML dokumentumok bejárására
  - W3C ajánlás (1999. novemberől)
- 
- fogalmazhatunk úgy, hogy az XSLT az XML forrás-fát átranszformálja egy XML eredmény-fává

## működési elve:

- a transzformáció során az XSLT az XPath-ot használja arra, hogy meghatározza a forrás dokumentum azon részeit, melyek egy vagy több előredefiniált sablonra (template) illeszkednek
- illeszkedés esetén az XSLT áttranszformálja a forrás dokumentum illeszkedő részét az eredmény dokumentummá

## XSL deklaráció:

```
< xsl:stylesheet version="1.0"  
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
```

vagy:

```
< xsl:transform version="1.0"  
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

- Egy XSL stílusállomány szabályok összességéből épül fel – ezeket sablonoknak (template) nevezzük.
- Az egyes sablonok arra vonatkozó szabályokat tartalmaznak, hogy mi történjen, ha egy illeszkedő csomópontot találunk.

### Az `<xsl:template>` elem

- Az `<xsl:template>` elem segítségével határozhatunk meg egy sablont.
- A *match* attribútum segítségével feleltethetünk meg egy sablont egy XML elemmel.
- Szintén a *match* attribútum segítségével lehet megadni a teljes XML dokumentumra illeszkedő sablont (`match="/" /`).

## a `<xsl:value-of>` elem

A `<xsl:value-of>` elemet használjuk arra, hogy egy csomópont értékét kinyerjük

## `<xsl:for-each>` elem

A `<xsl:for-each>` elem lehetővé teszi, hogy ciklusban hajtsuk végre a transzformációt.

a kimenetet szűrhetjük azáltal, hogy különböző kritériumokat rendelünk a `select` attribútumhoz a `<xsl:for-each>` elemen belül:

```
<xsl:for-each select="catalog/cd[artist='Gryllus Vilmos']" >
```

## Érvényes szűrő (filter) műveletek:

- =
- !=
- &lt; (kisebb)
- &gt; (nagyobb)

### <xsl:sort> elem

A kimenet rendezése a <xsl:sort> elem segítségével történik (az <xsl:for-each> elemen belül).

### <xsl:if> elem

Az <xsl:if> elem segítségével valamilyen feltételt szabhatunk az XML tartalmával kapcsolatban.

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

```
...szoveg amennyiben a kifejezes igaz ...
```

```
</xsl:if>
```

### `<xsl:choose>` elem

Az `<xsl:choose>` elemet együtt használjuk az `<xsl:when>` és `<xsl:otherwise>` elemekkel.

### `<xsl:apply-templates>` elem

Az `<xsl:apply-templates>` elem alkalmazza a sablont az aktuális elemre vagy ennek gyerek-csomópontjaira.

lásd pl.

SimpleXSLTTransform, SimpleTransformXSLTServlet

## Tipp Web-alkalmazásban való használatra:

HTML-t generálhatunk szerver oldalon XSLT segítségével – adatok megmutatására (pl. report generálás), form generálására is használható

## XML Editor-ok

A specializált XML szerkesztők segítenek

- a hibamentes XML dokumentumok szerkesztésében
- XML érvényességének vizsgálatában
- rákényszeríthetnek, hogy egy megadott XML struktúra szabályait betartsuk

pl. XMLSpy, XMLNotepad 2007 (XML szerkesztők)