EÖTVÖS LORÁND TUDOMÁNYEGYETEM
INFORMATIKAI KAR

INFORMATIKAI SZAKMÓDSZERTANI CSOPORT

MyTutor

Készítette: **Menyhárt László Gábor**
egyetemi tanársegéd
2003. május 1.

1. Tananyag

A rendszerben jelen pillanatban ezek a tananyagok találhatóak meg:

1.1. XML

eXtensible Markup Language (Kiterjeszthető Jelölő nyelv)

- Jelölő nyelv (Markup Language)
- Általában szöveges fileban tároljuk.
- Címkékből (tag-ekből), attributumokból és magából a tartalomból állnak.
- Hierarchikus szerkezetű.

Például:

- HTML
- WML

1.1.1. Keletkezés

SGML-re építve kezdték 1996-ban kidolgozni. 1998-ban jelent meg az ajánlás első változata. Most az 1.0 verziót használjuk.

1.1.2. Célok

- Közvetlenül használható az Interneten
- Alkalmazások támogassák
- Egyszerű kezelehetőség
- Ember által olvasható és érthető
- Egyszerűen elkészíthető
- Dokumentum típus definíciók formális, tömör, gyorsan elkészíthető
- Szabadon használható és terjeszthető

1.1.3. Felhasználás

Az XML dokumentum képes magában hordozni a dokumentum típus definícióját. A feldolgozó programok nem csak szintaktikus, hanem szemantikus ellenőrzést is képesek végrehajtani. Egy XML alapú formátum leíró nyelv felhasználásával az adat és a megjelenés élesen elválasztható egymástól.

1.1.3.1. Szintaxis

Az XML dokumentumok szintaktikai követelményeit írom le ebben a részben.

Példa

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<!-- Készítette: Menyhárt László Gábor -->
<szakdolgozat>
   <oldal azon="0">
      <nev>MyTutor</nev>
      <lablec mi="oldalszam" />
   </oldal>
</szakdolgozat>
```

Az első sorban szerepel az XML deklaráció, mely megmutatja a használt XML verziót és a karakterkódolást.

A második sorban egy megjegyzés szerepel.

A harmadik sorban a gyökérelem, a "<szakdolgozat>" található.

Majd egy újabb elem következik, ahol már egy attributum is található, az "<azon>".

A következő sorban egy szöveget tartalmazó csomópont latható.

Ezt egy üres elem követi.

A hetedik sorban lezárjuk a "</oldal>"-lal az "oldal" csomópontot.

Végül a gyökérelem lezárásával végződik a dokumentum.

Záró tag-ek

A HTML-vel ellentétben itt kötelező megadni a záró tag-eket. A nem szerepelhet anélkül.

A nyitó és záró tag-eknek megegyezőknek kell lenniük a betűket tekintve. Fontos, hogy a kis és nagy betűk különbözőeknek számítanak.

Az egymásba ágyazás nagyon fontos. Míg a HTML-ben nem fontos a záró tag-ek sorrendje, az XML dokumentum felépítése a leszármazásra épül, ezért itt a második a helyes.

```
<b><i>Vastag és dölt szöveg. HIBÁS!</b></i>
<b><i>Vastag és dölt szöveg. JÓ!</i></b>
```

Egy gyökérelem

Minden XML dokumentumnak egy gyökéreleme kell, hogy legyen. Ennk lehet gyereke, mely ugyancsak tartalmazhat egy leszármazottat.

```
<gyoker>
   <gyerek>
      <leszarmazott>...</leszarmazott>
   </gyerek>
</gyoker>
```

Attributumok

Az attribututmok név-érték párként szerepelnek. Kötelező betartani, hogy az érték idézőjelek között van. Így csak a második példa helyes.

```
<oldal azon=0>
<oldal azon="0">
```

Megjegyzés

A megjegyzést ugyanúgy kell írni, mint a HTML-ben.

```
<!-- Ez egy megjegyzés. -->
```

1.1.3.2. XML elemek

Az elemek kiterjeszthetőek és van köztük rokonsági kapcsolat.

Kiterjesztés

Egy létező XML dokumentum adatai közé még felvehető újabb elem, ha bővül az információhalmazunk.

```
<szakdolgozat>
   <oldal azon="0">
      <nev>MyTutor</nev>
      <lablec mi="oldalszam" />
   </oldal>
</szakdolgozat>
```

Bővülhet újabb elemekkel. Pl.: fejléccel és tartalommal.

```
<szakdolgozat>
   <oldal azon="0">
      <fejlec mi="cim" />
      <nev>MyTutor</nev>
      <tartalom>tananyag</tartalom>
      <lablec mi="oldalszam" />
   </oldal>
</szakdolgozat>
```

Rokonság

```
<szakdolgozat>
   <oldal azon="0">
      <fejlec mi="cim" />
      <nev>MyTutor</nev>
      <tartalom>tananyag</tartalom>
      <lablec mi="oldalszam" />
   </oldal>
</szakdolgozat>
```

Itt a "szakdolgozat" a gyökér elem. Az "oldal" leszármazottja a "szakdolgozat"-nak, míg a "szakdolgozat" szülője az "oldal"-nak. A "fejlec", "nev", "tartalom" és "lablec" leszármazottai az "oldal"-nak, egymásnak pedig testvérei.

Tartalom

Egy elem mindig a kezdő tag-től a záró tag-ig tart. A tartalma több féle lehet. Tartalmazhat más elemeket, és csak azokat (szakdolgozat). Tartalmazhat elemeket és szöveget a kevert típusban (oldal). Tartalmazhat csak szöveget (nev). Végül lehet üres elem, melynek nincs szöveges tartalma (lablec).

```
<szakdolgozat>
   <oldal azon="0">Tartalom
      <nev>MyTutor</nev>
      <lablec mi="oldalszam" />
   </oldal>
</szakdolgozat>
```

Az üres elemnek nincs lezáró tag-je, de jelezni kell, hogy ez üres elem. Ezért van a kezdő tag neve után a per (/) jel. Ez különbség a HTML-hez képest, hiszen ott lehetett magában álló tag.

```
<BR>
```

helyett

```
<BR />
```

Elnevezés

Az elemek nevei állhatnak betűkből, számokból és egyéb karakterekből. A név nem kezdődhet számmal vagy központozással, és nem kezdődhet az xml (XML, Xml, ...) karakterekkel sem. Nem tartalmazhat szünetet (space). Jó megoldás az aláhuzással (_) történő összekötés, mert így nem lehet a kivonással vagy objektum részével sem összekeverni.

A nevek olyan hosszúak lehetnek, amekkorát a felhasználó szeretne. Ugyanakkor hasznos rövid neveket találni és nem érdemes túl hosszan összefűzni az értelmes neveket. Például: <kis_kacsa> és nem <a_kacsa_ami_kicsi>

1.1.3.3. Attributumok

Az attributumokat az elemekhez tartozó további információk tárolására használjuk, melyek a kezdő tag-ekben foglalnak helyet.

Hely

Mint a HTML nyelvben, ittt is az elemekhez tartozó további információkat a kezdő tag-ben tároljuk név-érték párban.

```
<tartalom milyen="pelda">tananyag</tartalom>
```

Stílus

A név-érték párok írási formája a következő: a név után egy egyenlőség jel után idézőjel vagy macskaköröm között helyezkedik el az adott névhez tartozó érték. Kötelező az idézőjeleket kitenni az érték elé és mögé is. Így:

```
<tartalom milyen="pelda">tananyag</tartalom>
<tartalom milyen='pelda'>tananyag</tartalom>
```

A kezdő idézőjellel kell lezárni az értéket, így lehetőség van értékként megadni a másik fajta idézőjelet.

```
<tartalom milyen="pe'lda">tananyag</tartalom>
<tartalom milyen='su"ru"'>tananyag</tartalom>
```

Elem vagy attributum

Nem feltétlenül kell attributumként tárolnunk a kiegészítő információt. Nincs arra szabály, hogy mikor kell használni. Így a következő két példa minegyike helyes.

```
<szemely nem="ffi">
  <nev>LAci</nev>
</szemely>
<szemely>
  <nem>ffi</nem>
  <nev>LAci</nev>
</szemely>
```

Sokféle lehetőség van ugyannak az adatnak a tárolására. A következő három példa ezt mutatja be.

```
<cikk datum="2003/05/01">
  <szerzo>LAci</szerzo>
  <tartalom>tananyag</tartalom>
</cikk>
<cikk>
  <datum>2003/05/01</datum>
  <szerzo>LAci</szerzo>
  <tartalom>tananyag</tartalom>
</cikk>
<cikk>
  <datum>
    <ev>2003</ev>
    <honap>05</honap>
    <nap>01</nap>
  </datum>
  <szerzo>LAci</szerzo>
  <tartalom>tananyag</tartalom>
</cikk>
```

Nem érdemes attributumként tárolni az adatokat, ha összetett adatról van szó. Nem lehet bővíteni a tartalmat. Nehezebb programból kezelni az attributumokat.

1.1.3.4. Érvényesítés

A XML dokumentumokra különböző megszorításokat tehetünk. Megadhatjuk a pontos strukturát, de még az adatokat is korlátozhatjuk. Majd ellenőrzéseket hajthatunk végre a dokumentumon.

Jól formázott dokumentum

Az a jól formázott (Well Formed) XML dokumentum, ami megfelel az XML szintaktikának. A betartandó szabályokat az előző lapokon megismertük.

```
<?xml version="1.0" encoding="UTF-8"?>
<cikk datum="2003/05/01">
   <szerzo>LAci</szerzo>
   <tartalom>tananyag</tartalom>
</cikk>
```

Érvényes dokumentum

Az érvényes (Valid) XML dokumentum jól formázott és ezen kívül a strukturája megfelel a szerző által felállított szabályoknak. Ezeket a szabályokat is le kell írnunk, erre van a következő két lehetőség.

XML DTD

A DTD defíniálja az XML dokumentum elemeinek strukturáját. Bővebben a DTD-ről szóló részben írok erről.

XML Schema

Az XML Schema vagy XSD a DTD-nek egy XML formában leírt változata. Erről is lehet bővebben olvasni az XSD részben.

1.1.3.5. Hiba

Ha az XML dokumentum nem "jól formázott" vagy nem "érvényes", akkor a feldolgozás megszakad, amit a programoknak kell lekezelniük.

1.1.3.6. Böngészők

Az XML dokumentumok Interneten történő dokumentálása hatására a böngészők nagy része fel van készítve azok megjelenítésére.

Netscape 6

A Netscape a 6-os verziótól támogatja az XML dokumentumok kezelését.

Internet Explorer 5.0

Az Internet Explorer az 5.0-ás verziótól kezdve használja az XML dokumentumokat. Megjelenít (CSS-sel is), transzformál, dolgozik adatszigetekkel. Ugyanakkor a Microsoft saját függvényekkel egészítette ki a feldolgozást, így nem teljesen kompatibilis a W3C ajánlásával.
Az XML dokumentumok alapértelmezett megjelenítése úgy néz ki, hogy az elemeket szinezi és a struktúrában az elemek összegyűjthetők és kibonthatók. Hiba esetén az első hiba helyét megmutatja.
A HTML kódba beágyazható egy XML formátumú adatsziget (Data Island), melyet a böngésző feldolgoz. Ez az adatcsomag a HTML fájlban elkülönítve egy "xml" nevű elemként szerepel. A megjelenítésénél pedig erre hivatkozva dinamikusan megjelenik a megfelelő mennyiségű adat. Az xml elemet a következő két féle képpen adhatjuk meg.

```
<xml id="cikkek">
<cikkek>
   <cikk>
      <datum>2003/04/01</datum>
      <szerzo>LAci</szerzo>
      <tartalom>piszkozat</tartalom>
   </cikk>
   <cikk>
      <datum>2003/05/01</datum>
      <szerzo>LAci</szerzo>
      <tartalom>tananyag</tartalom>
   </cikk>
</cikkek>
</xml>
<xml id="cikkek" src="cikkek.xml">
</xml>
```

Példa az adatszigetre

```
<html>
<body>
<xml id="cikkek">
<cikkek>
   <cikk>
      <datum>2003/04/01</datum>
      <szerzo>LAci</szerzo>
      <tartalom>piszkozat</tartalom>
   </cikk>
   <cikk>
      <datum>2003/05/01</datum>
```

```
      <szerzo>LAci</szerzo>
      <tartalom>tananyag</tartalom>
   </cikk>
</cikkek>
</xml>

<table datasrc="#cikkek">
<tr>
<td><font color="#ff0000" size="6"><div datafld="datum"></div></font></td>
</tr>
<tr>
<td><font color="#0000ff" size="6" style="margin-left: 10pt"><div
datafld="szerzo"></div></font></td>
</tr>
<tr>
<td><font color="#00ff00" size="3" style="10pt;margin-left: 20pt"><div
datafld="tartalom"></div></font></td>
</tr>
</table>
</body>
</html>
```

1.1.3.7. Megjelenítés

Az XML dokumentumok nem csak adatokat tárolhatnak, közvetíthetnek, hanem azok megjelenítéséhez is tartalmazhatnak információkat.

Feldolgozási utasítás

A megjelenítést az XML dokumentumokat kezelő programok végzik. Például egy böngésző. A megjelenítéshez a feldolgozási utasítást maga az XML dokumentum tárolja. Vagyis megadja, hogy milyen típusú a feldolgozás és az ahhoz szükséges információkat, hol találja a megjelenítő program.

Kétféle képpen tudjuk megadni a megjelenítéshez a szükséges információkat. CSS (Cascading Style Sheets) illetve XSL (eXtensible Stylesheet Language) segítségével.

```
<?xml-stylesheet type="text/css" href="kinezet.css"?>
```

vagy

```
<?xml-stylesheet type="text/xsl" href="kinezet.xsl"?>
```

A referenciaként megadott hivatkozás egy külső, az XML dokumentumtól független másik fájl. Természetesen ennek a fájlnak elérhetőnek kell lenni a feldolgozás során.

CSS (Cascading Style Sheets)

Ezzel a módszerrel az XML dokumentumot szekvenciálisan feldolgozva tudjuk megjeleníteni úgy, mint egy HTML oldalt. A CSS fájlban írhatjuk le, hogy az egyes elemeket, milyen stílusban kerüljön megjelenítésre.

Példa a CSS fájlra

```
datum
{
Display: block;
color: #FF0000;
font-size: 20pt;
}
szerzo
{
Display: block;
color: #0000FF;
margin-left: 10pt;
font-size: 20pt;
}
tartalom
{
Display: block;
color: #00FF00;
font-size: 10pt;
margin-left: 20pt;
}
```

Példa a CSS-sel megjelenített XML dokumentumra.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="cikkek.css"?>
<cikkek>
   <cikk>
      <datum>2003/04/01</datum>
      <szerzo>LAci</szerzo>
      <tartalom>piszkozat</tartalom>
   </cikk>
   <cikk>
      <datum>2003/05/01</datum>
      <szerzo>LAci</szerzo>
      <tartalom>tananyag</tartalom>
   </cikk>
</cikkek>
```

XSL (eXtensible Stylesheet Language)

Az XSL segítségével nem csak szekvenciálisan lehet feldolgozni az XML dokumentumot, hanem hivatkozni lehet az egyes elemekre. Így összetettebb megjelenést lehet elérni.

Példa az XSL fájlra

```
<?xml version="1.0"?>
<xsl:stylesheet
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      version="1.0">
  <xsl:template match="//cikkek">
<html>
<body>
    <xsl:for-each select="//cikk">
      <font color="#ff0000" size="6"><xsl:value-of select="datum"/><br
/></font>
      <font color="#0000ff" size="6" style="margin-left: 10pt"><xsl:value-
of select="szerzo"/><br /></font>
      <font color="#00ff00" size="3" style="10pt;margin-left:
20pt"><xsl:value-of select="tartalom"/><br /></font>
```

```
    </xsl:for-each>
</body>
</html>
  </xsl:template>
</xsl:stylesheet>
```

Példa az XSL-lel megjelenített XML dokumentumra.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="cikkek2.xsl"?>
<cikkek>
   <cikk>
      <datum>2003/04/01</datum>
      <szerzo>LAci</szerzo>
      <tartalom>piszkozat</tartalom>
   </cikk>
   <cikk>
      <datum>2003/05/01</datum>
      <szerzo>LAci</szerzo>
      <tartalom>tananyag</tartalom>
   </cikk>
</cikkek>
```

1.1.3.8. Névterek

Az XML dokumentumok meghatározott elemeket tartalmazhatnak. Több dokumentumban előfordulhat, hogy azonos nevű elemek fordulnak elő. Ez névütközéshez vezet, amit ki kell küszöbölni.

Névütközés

Mivel az XML dokumentumok elemnevei nem rögzítettek, ezért előfordulhat több dokumentum kezelése során, hogy azonos elemnevek jelennek meg. Ezeket persze nem feltétlenül kell, sőt előfordulhat, hogy nem szabad azonos módon feldolgozni. Így nem lenne egyértelmű az adatkezelés.

A következő példában egy áru ára szerepel illetve egy cipészüzlet szerszámai közül egy ár adatai.

```
<ar>
  <mennyi>1234</mennyi>
  <valuta>HUF</valuta>
</ar>
<ar>
  <hossz>1234</hossz>
  <mertek>mm</mertek>
</ar>
```

Előnév

Megoldásként előneveket használunk. A név elé kettősponttal elválasztva beírunk még egy azonosítót. Így különbséget tudunk tenni az azonos nevű elemek között.

```
<aru:ar>
  <aru:mennyi>1234</aru:mennyi>
  <aru:valuta>HUF</aru:valuta>
</aru:ar>
<c:ar>
  <c:hossz>1234</c:hossz>
  <c:mertek>mm</c:mertek>
</c:ar>
```

Az előnév azonosítójának jelentését is megadjuk. Úgynevezett névtereket használunk az egyes elemnevek azonosításához, az elemek tartalmának típusáról.

A névtér megadása az elemnév első megjelenésénél attributumként történik. Az "xmlns" attributum értékeként kell megadni az egyedi nevet, mely azonosítja az előnevet.

```
<aru:ar xmlns:aru="http://xml.inf.elte.hu/2003/mytutor/xml/aru">
  <aru:mennyi>1234</aru:mennyi>
  <aru:valuta>HUF</aru:valuta>
</aru:ar>
<c:ar xmlns:c="http://xml.inf.elte.hu/2003/mytutor/xml/cipesz">
  <c:hossz>1234</c:hossz>
  <c:mertek>HUF</c:mertek>
</c:ar>
```

Az 'xmlns:aru="'' attributum adta meg az 'aru' előnév névterét. Alapértelmezett névteret is meg lehet adni, ami azt jelenti, hogy nincs előnév. Ezt a következő módon kell megtenni.

```
<ar xmlns="http://xml.inf.elte.hu/2003/mytutor/xml/aru">
  <mennyi>1234</mennyi>
  <valuta>HUF</valuta>
</ar>
```

1.1.3.9. Adatok

Az XML dokumentumban az adatok nem különülnek el az elemnevektől így a feldolgozás során azokat is végigolvassa a rendszer.

Elillanó karakterek (Escape Characters)

Vannak olyan karakterek, melyek feldolgozás megkönnyítése érdekében speciális tulajdonságokkal rendelkeznek. Például az "<" karakter helyett a "<" karaktersorozatot kell írni, mert az előző a tag-nél, vagyis az elemnév elejének és végének megadásánál van fontos szerepe.

Több ilyen XML feldolgozását megkönnyítő karakter van. Ezek helyett a helyettesítésükre definiált ENTITY karaktereket lehet írni. Az egyedek mindig a "&" karakterrel kezdődnek és a ";"-vel végződnek. Csak a < és & karakterek nem használhatóak az XML dokumentumokban, de van öt előre definiált egyed.

```
&lt;  <  kisebb, mint (less than)
```

```
&gt;  >  nagyobb, mint (greater than)
&amp; &  és (ampersand)
&apos;   '  aposztróf (apostrophe)
&quot;   "  macskaköröm (quotation mark)
```

CDATA

Olyan adathalmazok is, ahol nehéz lenne megoldani a karaktercseréket. Itt megadhatjuk, hogy az adott rész ne kerüljön feldolgozásra. Létre kell hoznunk egy CDATA, karakteradat szekciót. Így ebbe a részbe bármilyen adatot lehet írni. Kivétel a záró jelsorozatot nem lehet beírni a közepére, mert akkor ott érzékelné az adatcsomag végét.

```
<![CDATA[
```
A kezdő jelsorozat van az előző sorban.

Akármilyen, nem feldolgozott szöveg.

```
    <
&
```
A végét jelentő karaktersorozat:

```
]]>
```

Karakterkódolás

Az XML dokumentumokban bármilyen karaktert tárolhatunk. Hogy az feldolgozó programok is tudják, hogy milyen karakterekkel foglalkozniuk megadhatjuk a dokumentumban található karakterek kódolását. Ezt a legelső, az XML dokumentumot mindig elkezdő feldolgozási utasítás attributumaként tehetjük meg.

```
<?xml version="1.0" encoding="windows-1252"?>
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-16"?>
```

1.2. DTD

Document Type Definition (Dokumentum Típus Definíció)

1.2.1. Dokumentumtípus

Egy XML dokumentumnak elő lehet írni, hogy milyen adatokat tartalmazhat. Milyen elemekből állhat, azok milyen struktúrában épülnek fel. Az adattartalom pedig milyen típusnak kell, hogy megfeleljen.

Egyik lehetőség a dokumentumtípus definíció (DTD). Ezt két féle képpen tehetjük meg. Magában az XML dokumentuban tárolva vagy külön fájlban elhelyezve.

1.2.1.1. Belső deklaráció

Az XML dokumentum magában foglalva tárolja a dokumentumtípus definíciót. Ennek helye a feldolgozási utasítások után, de a gyökérelem előtt van. A megadás a következő formában történik.

```
<!DOCTYPE gyökérelem [elemdeklarációk]>
```

Példa DTD-re az XML dokumentumban megadva

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cikk [
   <!ELEMENT cikk (datum, szerzo, tartalom)>
   <!ELEMENT datum (#PCDATA)>
   <!ELEMENT szerzo (#PCDATA)>
   <!ELEMENT tartalom (#PCDATA)>
]>
<cikk>
   <datum>2003/05/01</datum>
   <szerzo>LAci</szerzo>
   <tartalom>tananyag</tartalom>
</cikk>
```

A második sorban van a dokumentumtípus definíciójának kezdete (!DOCTYPE). Itt megadjuk, hogy mi lesz a gyökér. Majd ezt követően felsoroljuk az elemeket és típusukat (!ELEMENT). A harmadik sorban van a gyökérelem neve illetve zárójelben felsorolva, hogy milyen alelemeket tartalmaz. A következő sorokban a "cikk"-ben szereplő elemek nevei és típusuk zárójelben.

1.2.1.2. Külső deklaráció

A dokumentumtípus definícióját külső fájlban is meg lehet adni. Ekkor a csatolandó fájhoz meg kell adni a hivatkozást a következő szintaxis szerint.

```
<!DOCTYPE gyökérelem SYSTEM "fájlnév">
```

Példa DTD-re az XML dokumentumon kívül megadva

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cikk SYSTEM "cikk2.dtd">
<cikk>
   <datum>2003/05/01</datum>
   <szerzo>LAci</szerzo>
   <tartalom>tananyag</tartalom>
</cikk>
```

A külső DTD fájl

```
<!ELEMENT cikk (datum, szerzo, tartalom)>
<!ELEMENT datum (#PCDATA)>
<!ELEMENT szerzo (#PCDATA)>
<!ELEMENT tartalom (#PCDATA)>
```

1.2.1.3. Érvényesítés

Hogy a dokumentumtípus definíció során megadott feltételeknek megfelel-e az XML dokumentum, azt a feldolgozó rendszer ellenőrzi le az ugynevezett validálás, érvényesítés során.

1.2.2. Építőelemek

Az XML dokumentum építőelemei nagyon hasonlítanak a HTML-hez. A DTD-ben a következőket adhatjuk meg.

1.2.2.1. Elemek

Az XML dokumentumoknak (, mint a HTML-ben) a fő építő kockái az elemek. Ezek tartalmazhatják az adatokat, más elemeket vagy lehetnek üresek.

1.2.2.2. Tag-ek

Ezek jelölik az elemeket. A kezdő tag az elem nevéből áll, míg a záró tagben az elem nevét megelőzi egy per (/) jel.

1.2.2.3. Attributumok

Az attributumok kiegészítő információkat tartalmaznak. Az elem kezdő tag-jében helyezkednek el. Egy attributum név-érték párban áll, köztük egyenlőségjellel, illetve az érték idézőjelek között szerepel.

1.2.2.4. Egyedek

Az egyedek (ENTITY-k) olyan változók, melyek valamilyen szöveggel (érték) helyettesítik a változót. Az XML tananyagban szerepel az XML-hez definiált öt egyed. Ezen kívül a HTML-ből ismert az   a "no-breaking-space" helyettesítése. Illetve mi is definiálhatunk egyedeket a DTD-ben.

1.2.2.5. PCDATA

Az elemek értelmezése során azok tartalma is feldolgozásra kerül.

1.2.2.6. CDATA

Külön kell megadni, hogy az elemek adatait ne értelmezze a rendszer.

1.2.3. Elemek

Elemek tartalmának megadása DTD-ben az ELEMENT paranccsal történik.

1.2.3.1. Szintaxis

A DTD-ben az elem deklarálása a következő formában történik.

```
<!ELEMENT elemnév kategória>
```
vagy

```
<!ELEMENT elemnév (az_elem_tartalma)>
```
Példák

Az üres elem kategóriája az EMPTY.

```
<!ELEMENT elemnév EMPTY>
```
Például:

```
<!ELEMENT br EMPTY>
```
Az XML-ben:

```
<br />
```
Ha csak szöveget tartalmaz az elem, akkor a PCDATA típust kell megadni.

```
<!ELEMENT elemnév (#PCDATA)>
```
Például:

```
<!ELEMENT szerzo (#PCDATA)>
```
Az ANY kategóriába beletartozik minden értelmezhető adat.

```
<!ELEMENT elemnév ANY>
```
Például:

```
<!ELEMENT cikk ANY>
```
Az elem leszármazottait fel is sorolhatjuk vesszővel elválasztva a zárójelben.

```
<!ELEMENT elemnév (leszármazott)>
```
vagy

```
<!ELEMENT elemnév (leszárm1,leszárm2,...)>
```
Például:

```
<!ELEMENT cikk (datum,szerzo,tartalom)>
```
Ekkor a leszármazottak típusait is ebben a sorrendben kell megadni, és meg is kell adni. Ha hiányzik valami, hibás a dokumentumdefiníció.

```
<!ELEMENT cikk (datum, szerzo, tartalom)>
<!ELEMENT datum (#PCDATA)>
<!ELEMENT szerzo (#PCDATA)>
<!ELEMENT tartalom (#PCDATA)>
```
Egy elem megadásakor ez az egy elemnek kell de másnak nem szabad szerepelnie.

```
<!ELEMENT elemnév (leszármazott)>
```
Példa:

```
<!ELEMENT cikk (tartalom)>
```
A következőben az elem leszármazottja legalább egyszer kell, hogy szerepeljen. Ezt a plussz jellel jelöljük.

```
<!ELEMENT elemnév (leszármazott+)>
```
Például:

```
<!ELEMENT cikk (tartalom+)>
```
Egy elem leszármazottjának nullaszor vagy többször való előfordulását a csillaggal jelöljük.

```
<!ELEMENT elemnév (leszármazott*)>
```

Például:

```
<!ELEMENT cikk (tartalom*)>
```
A nullaszor vagy egyszer való előfordulást a kérdőjel jelöli.

```
<!ELEMENT elemnév (leszármazott?)>
```
Például:

```
<!ELEMENT cikk (tartalom?)>
```
Vagy az egyik, vagy a másik elemet tartalmazza a szülő, azt a függőleges vonallal (pipe) jelöljük. A következő példában a cikk tartalmazza a dátumot és a tartalmat vagy a szerzőt és a tartalmat.

```
<!ELEMENT cikk ((szerzo|datum),tartalom)>
```
A következő példában a gyökérelem tartalmazhat kevert adatokat, vagyis szöveget és alelemeket is.

```
<!ELEMENT cikk (#PCDATA|datum|szerzo|tartalom)*>
```

1.2.4. Attributumok

Attributumok deklarálása a DTD-ben az ATTLIST paranccsal történik.

1.2.4.1. Szintaxis

Az attributumok deklarálása a következő formában történik.

```
<!ATTLIST elemnév attributumnév attributumtípus alapértelmezett_érték>
```
Példa:

```
<!ATTLIST cikk datum CDATA "2003/05/01">
```
esetén csak ez lehet:

```
<cikk datum="2003/05/01" />
```
Az attributumok típusa lehet:

- CDATA - Karakteres adat
- (v1|v2|..) - Választható a felsorolásból
- ID - Egyedi azonosító
- IDREF - Egy másik elem azonosítója
- IDREFS - Más azonosítók listája
- NMTOKEN - Érvényes név
- NMTOKENS - Érvényes nevek listája
- ENTITY - Egyed
- ENTITIES - Egyedek listája
- NOTATION - Jelölés neve

Az alapértelmezett_érték lehet:

- Érték
- #REQUIRED - kötelező ez az attributum
- #IMPLIED - nem feltétlenül tartalmazza ezt az attributumot
- #FIXED érték - fix érték

Példák

Az alapértelmezett érték megváltoztatható.

DTD:

```
<!ELEMENT cikk EMPTY>
<!ATTLIST cikk datum CDATA "2003/05/01">
```

Érvényes XML:

```
<cikk datum="2003/04/01" />
```

Nem kötelező attributum.

DTD:

```
<!ATTLIST cikk datum CDATA #IMPLIED>
```

Érvényes XML:

```
<cikk datum="2003/05/01" />
```

Érvényes ez is:

```
<cikk />
```

Kötelező attributum.

DTD:

```
<!ATTLIST cikk datum CDATA #REQUIRED>
```

Érvényes XML:

```
<cikk datum="2003/05/01" />
```

Ez nem érvényes:

```
<cikk />
```

Kötött értékű attributum.

DTD:

```
<!ATTLIST cikk szerzo CDATA #FIXED "LAci">
```

Érvényes XML:

```
<cikk szerzo="LAci" />
```

Ez nem érvényes:

```
<cikk szerzo="Masvalaki" />
```

Felsorolástípusú attributumérték.

DTD:

```
<!ATTLIST tartalom tipus (alap|pelda) "alap">
```

XML:

```
<tartalom tipus="alap" />
```

vagy

```
<tartalom tipus="pelda" />
```

1.2.5. Egyedek

A DTD-ben saját magunk is definiálhatunk egyedeket, melyeket az XML dokumentumban tudunk használni.

Szintaxis

Lehet megadott értékkel illetve külső hivatkozással is definiálni egy egyedet.

```
<!ENTITY egyednév "érték">
<!ENTITY egyednév SYSTEM "URL">
```

Példák

A cikk szerzőjét többször használjuk ezért bevezetünk egy változót.

DTD:

```
<!ENTITY iro "LAci">
```

XML:

```
<szerzo>&iro;</szerzo>
```

<!ENTITY tartalom SYSTEM "http://xml.inf.elte.hu/2003/mytutor/dtd/anyag.xml">

1.3. XSD

XML Schema Definition (XML séma definíció)

1.3.1. Bevezetés

Az XML séma alternatívája a DTD-nek. Az XML dokumentum strukturája, adatainak típusa írható le benne XML formátumban. Definiálhatjuk benne az XML dokumentumban szereplő elemeket, attributumokat, leszármazásokat, az elemek számára írhatunk előírásokat, adattípusokat használhatunk a megadás során és a rögzitett adatokat is megadhatjuk.

A DTD-nél jobban használható, több információt adhatunk meg. XML formátumú az XSD fájl. Támogatja az adattípusokat és a névtereket. Nem utolsó sorban pedig kiterjeszthető.

1.3.1.1. Példa

Megnézzük, hogyan kell egy XML dokumentumhoz megadni a dokumentumtípus definíciót DTD-ben illetve XSD-ben. Illetve hogyan kell megadni a dokumentumban a hozzá tartozó definíciós fájlt.

XML

Példánkban az XML dokumentum a következő lesz.

```
<?xml version="1.0"?>
<cikk>
   <datum>2003/05/01</datum>
   <szerzo>LAci</szerzo>
   <tartalom>tananyag</tartalom>
</cikk>
```

DTD

Az ehhez tartozó DTD:

```
<!ELEMENT cikk (datum, szerzo, tartalom)>
<!ELEMENT datum (#PCDATA)>
<!ELEMENT szerzo (#PCDATA)>
<!ELEMENT tartalom (#PCDATA)>
```

Ezt az XML dokumentumhoz a DTD tananyagban leírtak szerint lehet kötni. Például:

```
<!DOCTYPE cikk SYSTEM "cikk.dtd">
```

XSD

A példában szereplő XML-hez a séma (XSD):

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="cikk">
      <xs:complexType><xs:sequence>
            <xs:element name="datum" type="xs:string"/>
            <xs:element name="szerzo" type="xs:string"/>
            <xs:element name="tartalom" type="xs:string"/>
      </xs:sequence></xs:complexType>
   </xs:element>
</xs:schema>
```

Az XML dokumentumban így kell megadni:

```
<?xml version="1.0" encoding="UTF-8"?>
<cikk xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="cikk.xsd">
   <datum>2003/05/01</datum>
   <szerzo>LAci</szerzo>
   <tartalom>tananyag</tartalom>
</cikk>
```

1.3.2. Gyökérelem

A séma leírása XML formában történik. Az XML dokumentum gyökere: <schema>

Szintaxis

A sémaleíró nyelv (XSD) specifikációjában leírt kötött elemeket használhatunk az XML dokumentum strukturájának és az adatok típusának megadására.

A gyökérelem mindig a "schema", melynek névterét is megadhatjuk. A névtér a leírónyelv specifikációjának egyedi azonosítója.

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

1.3.3. Egyszerű típusok

Ebben a részben ismertetem az egyszerű típusokat.

1.3.3.1. Egyszerű elemek

Az egyszerű elemek csak szöveges adatot tárolnak. Nem tartalmazhatnak más elemet vagy attributumot.

Típus

Az egyszerű elem típusa lehet az XSD-ben előre definiált típus vagy saját típus, amiről később lesz szó.

Definíció

Egy egyszerű elem megadásakor le kell írni a nevét és típusát. Ezt a következő formában kell megtenni.

```
<xs:element name="elemnév" type="elemtípus"/>
```

Az előző példában már láttuk:

```
<datum>2003/05/01</datum>
<szerzo>LAci</szerzo>
<tartalom>tananyag</tartalom>
```

Ehhez a séma:

```
<xs:element name="datum" type="xs:string"/>
<xs:element name="szerzo" type="xs:string"/>
<xs:element name="tartalom" type="xs:string"/>
```

Gyakori típusok

Az XML sémában sok előre definiált típus van. Ezek közül a leggyakrabban használtak:

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

Alapértelmezett vagy fix érték

Az egyszerű elemnek megadhatunk alapértelmezett vagy kötött értéket.

Alapértelmezett érték:

```
<xs:element name="szerzo" type="xs:string" default="LAci"/>
```

Fix érték:

```
<xs:element name="szerzo" type="xs:string" fix="LAci"/>
```

1.3.3.2. Attributumok

Az attributumokat egyszerű típussal kell deklarélni, de csak komplex elemtípusokban szerepelhetnek attributumok.

Definíció

```
<xs:attribute name="attributumnév" type="attributumtípus"/>
```

A cikkes példánkban például így kell megadni az attributumot.

```
<cikk datum="2003/05/01">anyag</cikk>
<xs:attribute name="datum" type="xs:string"/>
```

Alapértelmezett vagy fix érték

Az attributumoknak megadhatunk alapértelmezett vagy kötött értéket.

Alapértelmezett érték:

```
<xs:attribute name="datum" type="xs:string" default="2003/05/01"/>
```

Fix érték:

```
<xs:attribute name="datum" type="xs:string" fixed="2003/05/01"/>
```

Kötelező vagy opcionális

Az alapértelmezés szerint az attributumokat nem feltétlenül kötelező megadni. Mégis előírhatjuk.

Kötelező

```
<xs:attribute name="datum" type="xs:string" use="required"/>
```

Opcionális

```
<xs:attribute name="datum" type="xs:string" use="optional"/>
```

Megszorítások

Megszorításokat adhatunk meg az XML elemek és attributumok adattartalmára.

Érték

Az adatok értékének megadhatjuk, hogy milyen intervallumba essen.

```
<xs:element name="mennyi">
 <xs:simpleType>
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="0"/>
    <xs:maxInclusive value="10000"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

Értékhalmaz

Pontosan megadhatjuk egy listában, hogy milyen értékeket vehet fel az a típusú elem.

```
<xs:element name="szerzo"> <xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:enumeration value="LAci"/>
    <xs:enumeration value="Másvalaki"/>
  </xs:restriction>
</xs:simpleType></xs:element>
```

Minta

A szöveg karaktereire mintát illeszthetünk.

```
<xs:element name="tartalom">
 <xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:pattern value="([a-z])*"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

Az előző szerint a karakterek kisbetűk voltak. Itt is lehet használni a plussz (+), csillag (*) és kérdőjel (?) karaketerket. Például legalább egy kis- vagy nagybetűt a következőképpen kell leírni.

```
<xs:pattern value="([a-z][A-Z])+"/>
```

Felsorolhatjuk a lehetséges elemeket.

```
<xs:pattern value="LAci|Másvalaki"/>
```

Megadhatjuk a mintában a pontos számot is.

```
<xs:pattern value="[a-zA-Z0-9]{8}"/>
```

Nem megjelenő karakterek

Az XML feldolgozó nem távolítja el a nem megjelenő karaktereket.

```
<xs:element name="address">
 <xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:whiteSpace value="preserve"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

A feldolgozás során szóközökkel helyettesítődnek a nem megjelenő karakterek.

```
<xs:element name="address">
 <xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:whiteSpace value="replace"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

A nem megjelnő karakterek eltávolításra kerülnek oly módon, hogy csak egy szóköz lesz ott.

```
<xs:element name="address">
 <xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:whiteSpace value="collapse"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

Méret

Az adatok hosszára is tehetünk megszorítást. Például egy jelszónak megadjhatjuk, hogy pontosan nyolc (8) karakter hosszú legyen.

```
<xs:element name="jelszo">
 <xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:length value="8"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

Vagy, hogy minimum öt (5), de maximum tíz (10) karakter hosszú lehet.

```
<xs:element name="jelszo">
 <xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:minLength value="5"/>
    <xs:maxLength value="8"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

Referencia

Megszorítás	Leírás
enumeration	Listaelemek megadása
fractionDigits	A szám jegyeinek maximális száma. Nullánál nagyobb vagy egyenlő.
length	A karakterek száma vagy a lista elemszáma. Nullánál nagyobb vagy egyenlő.
maxExclusive	Az érték maximumának megadása.
maxInclusive	Az érték maximumának megadása, mellyel egyenlő is lehet.
maxLength	A karakterek számának maximuma. Nullánál nagyobb vagy egyenlő.
minExclusive	Az érték minimumának megadása.
minInclusive	Az érték minimumának megadása, mellyel egyenlő is lehet.
minLength	A karakterek számának minimuma. Nullánál nagyobb vagy egyenlő.
pattern	Az érték mintáját definiálja
totalDigits	A maximálisan megengedett jegyek száma. Nullánál nagyobb vagy egyenlő.
whiteSpace	Nem megjelenő karakterek kezelése

1.3.4. Komplex típusok

Ebben a részben ismertetem az összetett, komplex típusokat.

Bevezetés

Komplex típusú elemek a következők. Az üres elem, aminek csak attributuma van.

```
<megjelenes datum="2003-05-01"/>
```

Több leszármazott elemet tartalmaz.

```
<cikk>
   <datum>2003-04-01</datum>
   <szerzo>LAci</szerzo>
   <tartalom>piszkozat</tartalom>
</cikk>
```

Összetett típusú az az elem, ami csak szöveget tartalmaz, de van attributuma is.

```
<tartalom megjelenhet="true">anyag</tartalom>
```

Végül a kevert típus, melyben felváltva fordulhat elő szöveg és leszármazott elem.

```
<tartalom>A <kiemel>fontos</kiemel> anyag.</tartalom>
```

Szintaxis

Több lehetőség van a komplex típus megadására. Definiálhatjuk az elemnél közvetlenül.

```
<xs:element name="cikk">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="datum" type="xs:string"/>
      <xs:element name="szerzo" type="xs:string"/>
      <xs:element name="tananyag" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Megadhatjuk külön típusként is, amit akár többször is fel tudunk használni.

```
<xs:element name="cikk" type="Tcikk"/>

<xs:complexType name="Tcikk">
  <xs:sequence>
    <xs:element name="datum" type="xs:string"/>
    <xs:element name="szerzo" type="xs:string"/>
    <xs:element name="tananyag" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

1.3.4.1. Üres elem

Tartalom nélküli legfeljebb attributumokat tartalmazó elem az üres elem.

Szintaxis

Üres elemnek magában nincs sok értelme, csak akkor ha attributumokat tartalmaz.

```
<megjelenes datum="2003-05-01"/>
```

Ezt a következőképpen kell megadni:

```
<xs:element name="megjelenes">
  <xs:complexType>
    <xs:attribute name="datum" type="xs:date"/>
  </xs:complexType>
</xs:element>
```

Vagy típussal így:

```
<xs:element name="megjelenes" type="Tmegjelenes" />

<xs:complexType name="Tmegjelenes">
  <xs:attribute name="datum" type="xs:date"/>
</xs:complexType>
```

1.3.4.2. Szöveges elem

Szöveget is és attributumot tartalmazó elem definiálása.

Szintaxis

Az egyszerű tartalmú típust meg lehet szorítani feltételekkel vagy ki lehet terjeszteni attributumokkal.

```
<xs:element name="elemnév">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="alaptípus">
        ...
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

vagy

```
<xs:element name="elemnév">
  <xs:complexType>
    <xs:simpleContent>
      <xs:restriction base="alaptípus">
        ...
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>


<tartalom megjelenhet="true">anyag</tartalom>
```

Ehhez a sémarészlet:

```
<xs:element name="tartalom">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="megjelenhet" type="xs:boolean" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

Természetesen itt is lehetett volna új típust bevezetni.

1.3.4.3. Kevert elem

Egy elemben a szöveges adat közepén is elhelyezkedhet egy leszármazott elem.

Szintaxis

Kevert elemben szöveg és leszármazott elem is található. Ennek a definíciója nagyon egyszerű akár új típus bevezetésével, akár anélkül is.

```
<tartalom>A <kiemel>fontos</kiemel> anyag.</tartalom>
```

Ehhez a következő sémarészlet tartozik:

```
<xs:element name="tartalom">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element name="kiemel" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

1.3.4.4. Típusjelölő

Egy elem leszármazottjainak típusát jelezhetjük.

Bevezetés

A típusok jelölését három csoportba oszthatjuk:

Rendezés

- All
- Choice
- Sequence

Előfordulás

- maxOccurs
- minOccurs

Csoportosítás

- Group name
- attributeGroup name

Rendezés

Megadhatjuk a leszármazott elemeknek az előfordulási sorrendjét. A következő sémarészletben definiálunk egy elemet, melynek aleleme egyszer bármilyen sorrendben előfordulhatnak, de egyszer mindenféle képpen.

```
<xs:element name="cikk">
  <xs:complexType>
    <xs:all>
      <xs:element name="szerzo" type="xs:string"/>
      <xs:element name="datum" type="xs:string"/>
      <xs:element name="tartalom" type="xs:string"/>
    </xs:all>
  </xs:complexType>
</xs:element>
```

A következő példában definiált alelemek közül egy és csak egy fordulhat elő az XML dokumentumban.

```
<xs:element name="telefon">
  <xs:complexType>
    <xs:choice>
      <xs:element name="siemens" type="Tsiemens"/>
      <xs:element name="nokia" type="Tnokia"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Itt pedig a felsorolt alelemek között mindegyiknek kell szerepelnie egyszer.

```
<xs:element name="cikk">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="datum" type="xs:string"/>
      <xs:element name="szerzo" type="xs:string"/>
      <xs:element name="tartalom" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Előfordulás

Az elemek előfordulására megszorításokat tehetünk. Megadhatjuk a minimumot és a maximumot is a "minOccurs" illetve a "maxOccurs" attributumok segítségével.

```
<xs:element name="cikkek">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="cikk" type="Tcikk"
      maxOccurs="100" minOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Csoportosítás

Csoportba foglalhatjuk az elemek és attributumok halmazát, amit fel tudunk használni a típus megadásánál. Az elemeket a "group", míg az attributumokat az "attributeGroup" segítségével. Egy példa az elemekre.

```
<xs:group name="kiegeszito_informacio_csoport">
  <xs:sequence>
    <xs:element name="datum" type="xs:string"/>
    <xs:element name="szerzo" type="xs:string"/>
  </xs:sequence>
</xs:group>

<xs:element name="cikk" type="Tcikk">

<xs:complexType name="Tcikk">
  <xs:sequence>
    <xs:group ref="kiegeszito_informacio_csoport"/>
    <xs:element name="tartalom" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

1.3.4.5. Mindegyik elem

Az XML dokumentum elemtípusok megadásánál lehetőség van együtt kezelni az elemeket és az attributumokat.

Szintaxis

Például egy elem alelemeinek előfordulását egyszerre is be tudjuk állítani.

```
<xs:element name="cikk">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="datum" type="xs:string"/>
      <xs:element name="szerzo" type="xs:string"/>
      <xs:element name="tartalom" type="xs:string"/>
      <xs:any minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Az elemeket közösen az "any", az attributumokat pedig az "anyAttribute" segítségével tudjuk elérni.

1.3.5. Adattípusok

Itt az adattípusok részletesebb ismertetése található meg.

1.3.5.1. Szöveg

A szöveges típusok alatt a karaktereket tartalmazó adatcsomagot értjük.

Szintaxis

Karakterek sorozatát tartalmazó típus.

```
<xs:element name="tartalom" type="xs:string"/>
```

Az előzőben minden karaktert tartalmazott a típus. Meg lehet adni, hogy normalizálva legyen a szöveg, vagyis ne legyen benne sorvége, kocsivissza és tabulátor jel.

```
<xs:element name="tartalom" type="xs:normalizedString"/>
```

Több féle szöveges típus létezik. Ezek a következők.

Név Leírás

ENTITIES

ENTITY

ID Az ID attributumot reprezentálja

IDREF Az IDREF attributumot reprezentálja

IDREFS

language Szöveges adat, ami létező nyelv azonosítóját tartalmazza

Name Érvényes XML nevet tartalmazó szöveges adat

NCName

NMTOKEN NMTOKEN attributumot reprezentál

NMTOKENS

normalizedString Normalizált szöveg, mely nem tartalmaz tabulátort, soremelés és kocsivissza jeleket

QName

string Szöveg

token Szöveg, mely nem tartalmaz kocsivissza, soremelés, tabulátor és több szóközt

Szöveges adattípusra tehető megszorítások:

- enumeration
- length
- maxLength
- minLength
- pattern
- whiteSpace

1.3.5.2. Szám

Számértéket tartalmazó típusok.

Szintaxis

Az egyszerű típusok megadása számok esetén.

```
<xs:element name="mennyi" type="xs:decimal"/>
```

Számtípusok:

Név Leírás

byte Előjeles 8 bites egész szám

decimal Decimális érték

int előjeles 32-bit egész szám

integer Egész szám

long Előjeles 64-bit egész szám

negativeInteger Negatív egész szám (.., -2, -1.)

nonNegativeInteger Nem negatív egész szám (0, 1, 2, ..)

nonPositiveInteger Nem pozitív egész szám (.., -2, -1, 0)

positiveIntegerPozitív egész szám (1, 2,..)

short Előjeles 16-bit egész szám

unsignedLong Előjel nélküli 64-bit egész szám

unsignedInt Előjel nélküli 32-bit egész szám

unsignedShort Előjel nélküli 16-bit egész szám

unsignedByte Előjel nélküli 8-bit egész szám

Megszorítások számtípusokra:

- enumeration
- fractionDigits
- maxExclusive
- maxInclusive
- minExclusive
- minInclusive
- pattern
- totalDigits
- whiteSpace

1.3.5.3. Dátum és idő

Dátumot és időt is tartalmazhatnak az elemek.

Szintaxis

A dátum típus alapértelmezés szerint a következő "CCYY-MM-DD" formában van definiálva, ahol a CC az évszázador, a YY az évet, az MM a hónapot végül a DD a napot jelöli.

```
<xs:element name="datum" type="xs:date"/>
```

Az eddig használt 2003/05/01-et ezért szöveges típusként kezeltük. Mostantól ennek a formája:

```
<datum>2003-05-01</datum>
```

Az időzónákat is megadhatjuk.

```
<datum>2003-05-01+01:00</datum>
```

Az időt "hh:mm:ss" formában kell megadni, ahol hh az órát, mm a perceket, az ss pedig a másodperceket jelöli.

```
<xs:element name="kezdés" type="xs:time"/>
```

Itt is megadhatjuk az időzónákat.

```
<kezdes>08:15:00+01:00</kezdes>
```

A dateTime típusban a dátum és idő is tárolva van "CCYY-MM-DDThh:mm.ss" formában. Az előzőekhez hasonló jelentéssel. A "T" karakter fixen szerepel az adatban.

```
<kezdes>2003-05-01T08:15:00-01:00</kezdes>
```

Dátum- és időtípusok:

Név Leírás

date Dátum érték

dateTime Dátum és idő egyben

duration Idő intervallum

gDay Nap (DD, dátum része)

gMonth Hónap (MM, dátum része)

gMonthDay Hónap és nap (MM-DD, dátum része)

gYear Év (CCYY, dátum része)

gYearMonth Év és hónap (CCYY-MM, dátum része)

time Idő érték

Megszorítások:

- enumeration
- maxExclusive
- maxInclusive
- minExclusive
- minInclusive
- pattern
- whiteSpace

1.3.5.4. Egyéb

Más adattípus, mely lehet logikai, bináris, lebegőpontos, ...

Szintaxis

Logikai típusok megadása:

```
<xs:attribute name="megjelenhet" type="xs:boolean"/>
<tartalom megjelenhet="true">anyag</tartalom>
```

Bináris adattípus megadása:

```
<xs:element name="kód" type="xs:hexBinary"/>
```

URI adattípus megadása:

```
<xs:attribute name="src" type="xs:anyURI"/>
<kép src="http://xml.inf.elte.hu/2003/mytutor/images/6/komm2.jpg" />
```

1.3.5.5. Egyéb adattípusok:

Név Leírás

anyURI

base64Binary

boolean

double

float

hexBinary

NOTATION

QName

Megszorítások:

- enumeration (a Boolean data type can NOT use this constraint)
- length (a Boolean data type can NOT use this constraint)
- maxLength (a Boolean data type can NOT use this constraint)
- minLength (a Boolean data type can NOT use this constraint)
- pattern
- whiteSpace

1.4. XPath

XML dokumentum címzése

1.4.1.1. Bevezetés

Az XPath egy XML dokumentum elemét címzi meg, a fa bejárásának megadásával. Az alapfüggvények használhatóak benne. Fontos alkalmazása az XSLT-ben van (később). Ugyanakkor ez az ajánlás nem XML alapokra épít.

A kifejezés hasonló a fájlok elérési utjának megadásához. Az egyes elemek nevével hivatkozhatunk az elem tartalmára.

Példa

```
<?xml version="1.0"?> <cikkek> <cikk> <datum>2003/04/01</datum>
<szerzo>LAci</szerzo> <tartalom>piszkozat</tartalom> </cikk> <cikk>
<datum>2003/05/01</datum> <szerzo>LAci</szerzo>
<tartalom>tananyag</tartalom> </cikk> </cikkek>
```

Az egyes elemeket a per (/) jellel választjuk el. Az abszolút megadást, azaz ha a gyökértől kezdjük a címzést, a per jellel kellkezdeni.

A gyökérelem:

```
/cikkek
```

Az első cikk:

```
/cikkek/cikk
```

annak a szerzője:

```
/cikkek/cikk/szerzo
```

Használhatóak az alapfüggvények számokra, szöveges adatokra és logikai kifejezésekre.

A 2003/05/01-ei cikk tartalmát adja vissza:

```
/cikkek/cikk[datum='2003/05/01']/tartalom
```

Szintaktika

Az XPath kifejezés megadásának formája a következő.

Elemek megnevezése

A gyökért?l való címzést a per (/) jellel kell kezdeni.

```
/cikkek/cikk/tartalom
```

Ha két per jelet adunk meg (//), akkor bármelyik szinten levő adott nevű elem tartalmát kapjuk vissza.

```
//tartalom
```

Helyettesítés

A csillag (*) karakter helyettesíti egy elem nevét, így egy elemet.

Az összes "cikk"-beli elem:

```
/cikkek/cikk/*
```

A "cikkek" második leszármazott "szerzo"-je:

```
/cikkek/*/szerzo
```

A második szinten levő "datum" (valamilyen gyökér, valamilyen szülő):

```
/*/*/datum
```

Minden elem:

```
//*
```

1.4.1.2. Szűrés

Az XPath kifejezésben szögletes zárójelek között tudunk az elemekre és az adattartalomra is szűrni.

Ha olyan elemet címzünk amiből több is van, akkor megadhatjuk, hogy hányadiknak a tartalmát szeretnénk megkapni.

```
/cikkek/cikk[first()]
/cikkek/cikk[2]
/cikkek/cikk[last()]
```

Olyan elemre is szűrhetünk, amiben szerepel valamilyen másik elem, vagy annak értékére szűrünk.

```
/cikkek/cikk[datum='2003/05/01']/tartalom
```

Több útvonal megadása

Több útvonalat megadhatunk a függőleges vonal (pipe, |) segítségével.

```
/cikkek/cikk/datum | /cikkek/cikk/tartalom
```

Hivatkozás attributumokra

Az attributumokra úgy tudunk hivatkozni, hogy a nevük elé írjuk az "at"-jelet (@).

```
<?xml version="1.0" encoding="UTF-8"?>
<cikkek>
   <cikk datum="2003/04/01">
      <szerzo>LAci</szerzo>
      <tartalom>piszkozat</tartalom>
   </cikk>
   <cikk datum="2003/05/01">
      <szerzo>LAci</szerzo>
      <tartalom>tananyag</tartalom>
   </cikk>
</cikkek>


//cikk[@datum='2003/05/01']/tartalom
```

1.4.1.3. Helyi címzések

Abszolút és relatív címzések az XPath-ban.

Abszolút címzés

Az abszolút címzést a per (/) jellel kell kezdeni, és minden leszármazottat végig meg kell adni.

```
/cikkek/cikk/tartalom
```

Relatív címzés

A relatív címzésnél az aktuális pozíciótól kezdődik a következő elemnév keresése.

```
cikk/tartalom
```

1.4.1.4. Irány megadása

A rokonsági kapcsolatot is megadhatjuk, mint irányt a következő elemnév kereséséhez. Egy lépéshez tehát a következőket adhatjuk meg.

```
irány::elemnév[feltétel]
child::cikk[datum='2003/05/01']
```

ancestor Az ős, aki lehet a szülő, nagyszülő, ...

ancestor-or-self Az ős vagy az aktuális elem

attribute Attributum

child A gyermek, az első leszármazott

descendant Leszármazott, aki lehet gyermek, unoka, ...

descendant-or-self Leszármazott vagy az aktuális

following Az XML dokumentumban következő elemek

following-sibling A következő testvér elemek

parent Szülő

preceding Az aktuális elem előtti elemek az XML dokumentumban

preceding-sibling Az aktuális elemet megelőző testvérelemek

self Az aktuális elem

1.4.1.5. Feltételek

A feltételeket szögletes zárójelben kell megadni ([]) mindig arra az elemre vonatkozóan, ami mögé írjuk a szűrést.

Az első öt cikk.

```
//cikk[position()<6]
```

Helyettesítés

A következő rövidítéseket lehet alkalmazni az irányok megadására.

```
child::
nem kell kiírni
pl.: child::cikk - cikk

attribute::
@
pl.: attribute::datum - @datum

self::node()
.
pl.: self::node()/child::cikk - ./cikk

parent::node()
..
pl.: parent::node()/child::cikk - ../cikk
```

1.4.1.6. Kifejezések

Az XPath-ban használható kifejezések.

Numerikus

Összeadás és kivonás

```
+
6+4=10
-
10-6=4
```

Szorzás

```
*
5*2=10
```

Osztás egésszel

```
div
10 div 3=3
```

Maradék képzés

```
mod
10 mod 3=1
```

Egyenlőség

Egyenlő

```
=
mennyi=1234
```

Nem egyenlő

```
!=
mennyi!=1000
```

1.4.1.7. Reláció

Kisebb, mint és kisebb vagy egyenlő, mint

```
<
mennyi<2000
<=
mennyi<=2000
```

Nagyobb, mint, és nagyobb vagy egyenlő, mint

```
>
mennyi>1000
>=
mennyi>=1000
```

Logikai

Logikai "vagy" művelet.

```
or
datum='2003/05/01' or datum='2001/01/01'
```

Logikai "és" művelet.

```
and
szerzo='Laci' and datum='2003/05/01'
```

1.4.1.8. Függvények

Az XPath-ban használható függvények.

Elemek halmaza

Visszatér az elemek számával.

```
count(elemek) count(//cikk)=2
```

Visszaadja az elemek egyedi azonosítóját.

```
elemek_halmaza=id(érték)
```

Visszaadja az elemek halmazásból az utolsót.

```
last()
```

Visszatér az elem nevével.

```
name() - name(/cikkek/cikk)=cikk
```

Visszaadja az elemek halmazából az aktuális elem sorszámát.

```
position()
```

Szöveg

Visszaadja a paraméterként megkapott szövegeket összefűzve.

`concat(p1,p2,..) - concat('LAci',' ','szakdolgozata')='LAci szakdolgozata'`
Igazat ad vissza, ha másodikként megadott szöveget tartalmazza az első, különben hamis.

`contains(szöveg,rész) - contains('LAci','Ac')=IGAZ`
Eltávolítja a fölösleges helyeket a szövegből.

`normalize-space(szöveg) - normalize-space('LAci szakdolgozata')='LAci szakdolgozata'`
Igazzal tér vissza, ha az első paraméter azzal a második paraméterrel kezdődik.

`starts-with(szöveg,rész) - starts-with('LAci','LA')=IGAZ`
Szöveggé konvertál.

`string(érték) - string(100)='100'`
Visszaadja a szöveg hosszát.

`string-length(szöveg) - string-length('szakdolgozat')=12`
Visszaadja a paraméterszöveg egy részét.

`substring(szöveg,mettől,mennyit) - substring('LAci',1,2)='LA'`
Visszaadja az első paraméterben előforduló második paraméter utáni részt.

`substring-after(szöveg,rész) - substring-after('08:15',':')='15'`
Visszaadja az első paraméterben előforduló második paraméter előtti részt.

`substring-before(szöveg, rész) - substring-before('08:15',':')=08`
Visszaadja az első paramétert, miután kicserélte az összes második paramétert a harmadikra.

`translate(szöveg,mit,mire) - translate('08:15',':','.')='08.15'`

Numerikus

Visszaadja a legkisebb egészet, amely nem kisebb, mint az argumentumként megadott szám.

`ceiling(szám) - ceiling(3.14)=4`
Visszaadja a legnagyobb egészet, ami nem nagyobb, mint az argumentumként megadott szám.

`floor(szám) - floor(3.14)=3`
Számmá konvertál egy szöveget.

`number(szöveg) - number('100')=100`
Kerekít a számot a legközelebbi egészre.

`round(szám) - round(3.14)=3`
Visszaadja az elemhalmaz értékeinek összegét.

`sum(elemhalmaz) - sum(//ar/mennyi)=1234`

Logikai

Az értéket logikaira konvertálja.

`boolean(érték)`
Azonosan hamis illetve azonosan igaz értékkel tér vissza.

`false(), ture()`
Hamissal tér vissza, ha a paraméter igaz, és igazzal, ha a paraméter hamis.

`not(logikai)`

1.5. SAX

Simple API for XML (Egyszerű programozási interfész az XML-hez)

1.5.1. Bevezetés

Az XML dokumentumok szöveges fájlokban vannak. Ezek feldolgozásának legegyszerűbb formája, ha a dokumentumot szekvenciálisan, azaz az elejétől a végéig haladva kezeljük az adatokat.

A dokumentumon való végighaladás közben események történnek, amik hatására eseménykezelők futnak le. Öt féle eseménykezelő van.

1. Dokumentum kezdete
2. Elem kezdete
3. Karaktersorozat
4. Elem vége
5. Dokumentum vége

1.5.2. Eseménykezelők

Ebben a részben az elvi, logikai elképzelést ismertetem. Az egyes programozási nyelvekben megírt XML-t feldolgozó függvények könyvtárai ettől eltérően is működhetnek.

Dokumentum kezdete

(StartDocument) A dokumentum elején egyszer hívódik meg. Nincs paramétere.

Elem kezdete

(StartElement) Az elem kezdő tag-hez érve hívódik meg. Paraméter az elem neve és a kezdőtagben szereplő attributumok listája név/érték párban.

Karaktersorozat

(Characters) Az elem szöveges tartalmának feldolgozásakor hívódik meg. Paraméterként maga a tartalom kerül átadásra.

Elem vége

(EndElement) Az elem feldolgozásakor a záró tag olvasásakor történik ez az esemény. Paraméterként átadódik az elem neve.

Dokumentum vége

(EndDocument) A dokumentum feldolgozása végén egyszer hívódik meg. Nincs paramétere.

Feldolgozási utasítás

(ProcessingInstruction) A feldolgozási utasításoknál is meghívódik egy eseménykezelő. Paraméterek az attributumok.

Példa

```
<?xml version="1.0"?>
<cikk>
   <datum>2003/05/01</datum>
   <szerzo>LAci</szerzo>
   <tartalom>tananyag</tartalom>
</cikk>
StartDocument();
StartElement("cikk","");
StartElement("datum","");
Characters("2003/05/01");
EndElement("datum");
StartElement("szerzo","");
Characters("LAci");
EndElement("szerzo");
StartElement("tartalom","");
Characters("tananyag");
EndElement("tartalom");
EndElement("cikk");
EndDocument();
```

1.6. DOM

Document Object Model (Dokumentum Objektum Modell)

1.6.1. Bevezetés

A DOM egy programozási interfész, mely az XML dokumentumok feldolgozásához tartalmaz könyvtárakat.

Az XML dokumentum megnyitása, értelmezése (parszolás, parse) során betöltődik a memóriába, ahol fa strukturában lehet elérni az elemeket (rekurzív típus).

A gyökér elemet a "documentElement" adja meg. Minden elemnek van egy vagy több leszármazottja ("childNodes"). Ezek a leszármazottak tartalmazzák az elemeket, de fontos jegyezni, hogy egy elem nem egy csomópont (node). Egy csomópont több részből áll. Ezek a következők: kezdő tag, attributumok, elem tartalma.

1.6.2. Érvényesítés

Mint a SAX-ban itt is van érvényesítés. Ekkor nem a szintikát, hanem az XML dokumentumhoz tartozó DTD vagy séma szerint leírt struktúra és adattartalom ellenőrzése történik.

1.6.3. A csomópontok típusai

A csomópontok több félék lehetnek, melyeket az alábbi táblázat mutat be.

Típus Típusnév Csomópont neve Csomópont értéke

1 element tagName null

2 attribute name value

3 text#text content of node

4 cdatasection #cdata-section content of node

5 entityreferenceentity reference name null

6 entity entity name null

7 processinginstruction target content of node

8 comment #comment comment text

9 document #documentnull

10 documenttype doctype name null

11 documentfragment #document fragment null

12 notation notation name null

A csomópontok típusahoz tartozó konstans nevek a következők.

Típus Konstans nevek

1 ELEMENT_NODE

2 ATTRIBUTE_NODE

3 TEXT_NODE

4 CDATA_SECTION_NODE

5 ENTITY_REFERENCE_NODE

6 ENTITY_NODE

7 PROCESSING_INSTRUCTION_NODE

8 COMMENT_NODE

9 DOCUMENT_NODE

10 DOCUMENT_TYPE_NODE

11 DOCUMENT_FRAGMENT_NODE

12 NOTATION_NODE

1.6.4. Objektumok

Ezen az oldalon táblázatos formában bemutatom a DOM-ban szereplő objektumokat. Vannak olyan tulajdonságok illetve eljárások, amelyeket csak az Microsoft Internet Explorer támogat, és nem felelnek meg a W3C ajánlásnak.

Csomópont

Név Leírás

attributes Visszaadja az attributumokat

childNodes Visszaadja a leszármazott csomópontok listáját

firstChild Visszaadja az első leszármazott csomópontot

lastChild Visszaadja az utolsó leszármazott csomópontot

nextSibling Visszaadja a következő testvér csomópontot.

nodeName Visszaadja a csomópont nevét

nodeType Visszaadja a csomópont típusát

nodeValue Visszaadja a csomópont értékét

ownerDocument Visszaadja a tulajdonos dokumentumot

parentNode Visszaadja a szülő csomópontot

previousSibling Visszaadja az előző testvér csomópontot

appendChild(newChild) A megadott csomópontot leszármazottként fűzi a csomóponthoz

cloneNode(boolean) Visszaadja a csomópont másolatát. Igaz paraméter megadása esetén a leszármazottakat is.

hasChildNodes() Igazzal tér vissza, ha van leszármazott csomópont

insertBefore(newNode,refNode) Beszúr egy csomópontot a hivatkozás csomópont elé

removeChild(nodeName) Eltávolítja a megadott nevű csomópontot

replaceChild(newNode,oldNode) Lecseréli a megadott régi csomópontot a megadott újra

Csomópontok listája

Név Leírás

length Visszatér a lista elemszámával

item(i) Visszaadja a lista egy elemét (az i-diket)

Dokumentum

Név Leírás

documentElement Visszatér a dokumentum gyökérelemével

doctype Visszaadja a DTD-t vagy sémát

createAttribute(attributeName) A megadott névvel egy attributumot hoz létre

createCDATASection(text) Létrehoz egy CDATA részt a megadott tartalommal

createComment(text) Létrehoz egy megjegyzést a megadott tartalommal

createElement(tagName) Létrehoz egy elemet a megadott névvel

createEntityReference(referenceName) Létrehoz egy egyedhoivatkozást a megadott névvel

createProcessingInstruction(target,text) Létrehoz egy feldolgozási utasítást megadó csomópontot a megadott hellyel és tartalommal

createTextNode(text) Létrehoz egy szöveges csomópontot a megadott tartalommal

getElementsByTagName(tagName) Visszatér a megadott nevű elemek listájával

Elem

Név Leírás

tagName Beállítja vagy visszatér az elem nevével

getAttribute(attributeName) Visszaadja a megadott nevű attributum értékét

getAttributeNode(attributeName) Visszatér a megadott nevű attributummal, mint csomópont

getElementsByTagName(tagName) Listában adja vissza a megadott nevű elemeket

normalize() Normalizál

removeAttribute(attributeName) Kitörli a megadott nevű attributum értékét

removeAttributeNode(attributeNode)Kitörli a megadott nevű attributumot

setAttribute(attributeName, attributeValue) Új attributmot szúr be név és érték alapján

setAttributeNode(attributeNodeName) Beszúr egy adott attributumot

Attributum

Név Leírás

name Beállítja vagy visszatér az attributum nevével

specified Logikai értékkel tér vissza. Hamis, ha az attributum értéke alapértelmezett a DTD vagy séma szerint

value Beállítja vagy visszatér az attributum értékével

Szöveg

Szöveges csomópont létrehozásakor sem kell megadni a nevet csak az értéket, de itt "#text" lesz a név.

```
createTextNode("szöveg");
```

CDATA

CDATA rész létrehozásához csak annak a szövegét kell megadni, hiszen ennek csomópontnak nincs neve, csak értéke.

```
createCDATASection("Ez lesz a CDATA-ban < &");
```

Megjegyzés

Megjegyzés létrehozásához csak a megjegyzés szövegét kell megadni, hiszen a megjegyzés csomópontnak nincs neve, csak értéke.

```
createComment("Ez lesz a megjegyzés szövege.");
```

2. Tartalomjegyzék