Az XML 1.0 szabvány

tanulmány

Készítette: Mészáros Tamás

Budapest, 2001

BME Méréstechnika és Információs Rendszerek Tanszék

Tartalomjegyzék

Bevezetés

Jelen tanulmány célja összefoglalni az Extensible Markup Language (XML) alapszabványt. A szabvány technológiai ismertetője előtt kitérünk kialakulásának rövid történetére, illetve a tanulmány végén röviden összevetjük az SGML szabvánnyal, valamint ismertetjük a World Wide Web Consortium (W3C) tevékenységét a szabványhoz kapcsolódó területeken.

Az XML rövid története

1969-ben egy IBM munkacsoport kidolgozott egy GML (Generalized Markup Language – Általánosított Jelölő Nyelv) nevű leíró nyelvet különböző rendszereken alkalmazott eltérő dokumentum formátumok problémájának kezelésére. A GML számos IBM dokumentációs rendszer alapját képezte, és az elkövetkező években nemzetközi szabvánnyá alakult, amely a Standard Generalized Markup Language (SGML) nevet kapta. Az SGML 1986-ban vált ISO szabvánnyá (ISO 8879).

Az SGML nagy iparvállalatok dokumentációs szabványává vált olyan területeken, mint a hadiipar, elektronikus kiadványszerkesztés, illetve az autógyártás. Főbb erősségei az implementáció-függetlenség, strukturáltság, az általános és bővíthető leírási formátum, illetve a programozott feldolgozás egyszerűsége voltak. Ezek népszerűvé tették a technológiát a már említett alkalmazási területeken. Ugyanakkor számos gátló tulajdonsága (mint például a túlzott definíciós szabadsága, a drága implementációk) megakadályozták abban, hogy az elektronikus dokumentum-feldolgozás más területein is elterjedjen.

1989-ben kutatók a CERN Európai Nukleáris Kutatóközpontban információ megosztási problémák megoldására kidolgoztak egy SGML-en alapuló hipertext dokumentum formátumot, a HTML-t (Hypertext Markup Language). A HTML, mint SGML alkalmazás számos tulajdonságot örökölt az SGML-től (implementáció-független jelölő nyelv), ugyanakkor az egyszerűbb alkalmazhatóság érdekében erősen korlátozta annak képességeit (rögzített jelölő készlet). A HTML formátum, illetve az azt használó World Wide Web a kilencvenes években robbanásszerűen elterjedt. A formátum, illetve a web egyéb szabványainak gondozását a World Wide Web Consortium (W3C) vette át.

A W3C a HTML, illetve azt kiegészítő szabványok számos változatát készítette el az évek során. A kilencvenes évek végére a HTML dokumentumok számának dinamikus növekedése, az azokban tárolt információ elérésének gondjai előtérbe helyezték a HTML eredeti, SGML-el szembeni korlátjait. Eleinte kiegészítő szabványok és elképzelések próbálták a problémákat orvosolni, azonban ezek kudarcát látva a W3C úgy döntött, hogy kidolgoz egy új, az SGML-en alapuló, de annak tartalomleíró képességeit nem korlátozó új szabványt, amit XML-nek neveztek el.

Az Extensible Markup Language (XML) a W3C új web dokumentum szabványa. Az 1.0-ás változata 1998-ban vált szabvánnyá (W3C Recommendation). Az SGML számos tulajdonságát megőrzi (bővíthető jelölő nyelv), ugyanakkor egyszerűbben implementálható eszközök készítését teszi lehetővé, így széles körben alkalmazható.

Az XML szabvány első munkapéldánya számos célkitűzést megfogalmazott, amit a szabványnak teljesítenie kell:

- legyen egyszerűen használható a webes rendszerekben,
- az alkalmazások széles körét támogassa,
- legyen kompatibilis az SGML-el.
- legyen egyszerű XML dokumentumokat feldolgozó programokat írni,
- minél kevesebb opcionális képesség, tulajdonság legyen benne,
- ember által olvasható, világos szerkezetű dokumentumok legyenek,
- a szabvány tervezése és kialakítása formális és dinamikus legyen,
- legyen egyszerű XML dokumentumokat készíteni, és végül
- a tömörség nem lényeges szempont.

A következőkben ismertetjük a szabvány leglényegesebb elemeit.

Az XML szabvány

Az XML alig különbözik az SGML szabványtól. Majdnem minden olyan tulajdonságot birtokol, amit az SGML, ugyanakkor bizonyos korlátozásokat alkalmaz a dokumentumok készítése során, amelyek azonban nem korlátozzák azok feldolgozását és megjelenítését. Ennek alapvető oka az, hogy az XML elsősorban a felhasználással kapcsolatos területekre koncentrál, és kis mértékben korlátozza a dokumentumok készítésének SGML-ben létező szabadságát annak érdekében, hogy XML-alapú rendszereket egyszerűbb legyen implementálni.

A következő részben összefoglaljuk az XML szabvány legfontosabb elemeit. Az SGML szabványban jártas olvasók ez a fejezetet átléphetik, számukra az „SGML és XML szabványok közti különbség” c. fejezetet ajánljuk.

Ahhoz, hogy tisztán lássunk az XML technológia területén a legfontosabb fogalmak tisztázásával kezdünk.

Jelölő nyelvek

Egy jelölő nyelv (markup language) ún. jeleket (markup) használ a reprezentált szöveg különböző jellemzőinek, illetve értelmezési, feldolgozási instrukcióinak leírására. A jelölés (markup) kifejezés történetileg a szerzők, illetve kiadók által a nyomtatott oldalakra elhelyezett gépelőknek, illetve nyomdászoknak szánt formázási utasításokat jelentette. Az elektronikus dokumentum feldolgozásban azonban jelentése kibővült mindazon, a szöveges információt kiegészítő jelekkel, amelyek azok feldolgozását, értelmezését, illetve megjelenítését segítik.

A címkék általában speciális karakterek által jelölt szövegek, amelyek a megjelölt szöveg előtt illetve után állnak, így közrefogják azt. Az alábbi példa bemutat egy szövegrészletet, amit egy „cím” jellel jelöltünk meg:

```
<dőlt_betű>Az XML technológia</dőlt_betű>
```

A jeleket gyakran címkének (tag) is nevezik, illetve a magyar nyelvben elterjed az eredeti angol szó fonetikus átírása is (teg, tegelés címke, illetve címkézés értelemben).

A szöveges dokumentumokban elhelyezett jelek a szöveg egy-egy részét fogják közre és csatolnak hozzá oda tartozó információt. Egy jelölés alapvetően kétféle lehet: procedurális, illetve leíró avagy szemantikus. A procedurális jelölés (procedural markup) meghatározza, hogy a jelölt szövegrészhez érve mi a teendő, hogyan kell feldolgozni, megjeleníteni, stb. A leíró jelölés (descriptive markup), vagy más néven szemantikus jelölés (semantic markup) utasítások helyett a jelölt szövegrész értelmezésében segít. Procedurális jelölés alkalmazásának tipikus példája a HTML nyelv, ahol az alkalmazott jelölések (legalábbis az eredeti, alap HTML szabványban) a jelölt szövegrész megjelenítési utasításait tartalmazzák.

A leíró jelölés tágabb teret enged a szöveget feldolgozó, illetve megjelenítő rendszereknek, illetve lehetővé teszi a jelölt szöveg jobb emberi értelmezését is. A feldolgozás során lehetővé válik a jelöléshez (annak értelmezésétől, illetve a végzett feladattól függően) többféle eljárás rendelése, például külön a megjelenítés, illetve tartalmi szűrés számára. Leíró jelölésre mutat példát a következő szövegrészlet:

```
<Cím>1119 Budapest, Magyar tudósok körútja 2.</cím>
```

Leíró jelölést alkalmazó nyelv az SGML, illetve az XML is.

Strukturált jelölő nyelv

A jelölő nyelvekben alkalmazott címkék nemcsak egy adott szövegrészlet megjelölésére alkalmasak, hanem a szöveg struktúrájának, tartalmi felépítésének a leírására is. A címkék nemcsak egymás után, lineárisan helyezkedhetnek el a szövegben, hanem meghatározott szabályok szerint egymásba ágyazva is. Ezáltal lehetővé válik a szöveg tartalmi struktúrájának reprezentálása is.

A következő példában bemutatunk egy egyszerű struktúrát:

```
<cím>
     <irányítószám>1119</irányítószám>
     <város>Budapest</város>
     <utca>Magyar tudósok körútja</utca>
```

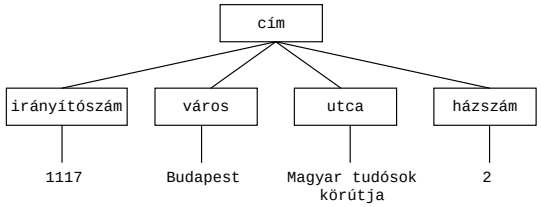
```
    <házszám>2</házszám>
</cím>
```

A példában látható város címke például a cím struktúrán belül található, így tartalmilag ahhoz tartozik, azaz cím város részadatát jelöli.

A címkék alkalmazásának legfontosabb szabálya, hogy egymásba ágyazásuk nem lehet fésűszerű, azaz két címke által megjelölt szövegrész nem lapolódhat át. A következő példa egy ilyen, hibásan jelölt szövegrészletet mutat.

```
<cím>
    <irányítószám>1119<város></irányítószám>Budapest</város>
</cím>
```

A címkék egymásba ágyazása meghatározza egy dokumentum struktúráját. A címkékkel jelölt szövegrészletek, illetve azok egymáshoz képesti viszonya lényegében egy fa struktúrát határoz meg. Az alábbi ábra a helyes cím szövegrészletet fa felépítését mutatja



A fa ágain levő csomópontok az egyes címkéknek felelnek meg, míg a fa levelein található a tényleges szöveges információ.

Nyelv és metanyelv

A jelölő nyelv tisztázása után egy másik fontos fogalmat kell megvizsgálnunk: a metanyelvet.

Míg a HTML egy rögzített jelölő készlettel rendelkező nyelv, addig az XML (mint az SGML is) egy metanyelv, amely alkalmas arra, hogy segítségével céljainknak megfelelő jelölő készletet, azaz nyelvet definiáljunk. A konkrét jelölő készletet (azaz egy konkrét nyelvet) az az alkalmazás határozza meg, amely céljaira készítjük a címkéket.

Az XML szabvány tág teret enged a címkék definiálásának. Egy meghatározott karakterkészlet-korlátozás betartásával tetszőleges jeleket, azaz tetszőleges nyelvet definiálhatunk. A nyelv definíciója nemcsak az alkalmazható címkék készletét határozza meg, hanem azok lehetséges struktúráját, illetve azokhoz kapcsolódó kiegészítő információkat is.

A HTML eredetileg egy ilyen módon SGML-ben definiált nyelv, más szóval élve SGML alkalmazás volt. Az XHTML szabvány a W3C által definiált az XML szabványnak megfelelő nyelv, azaz egy XML alkalmazás.

Dokumentum típus deklaráció

Egy dokumentumban használható címkéket, azaz az alkalmazható nyelvet az ún. dokumentum típus deklaráció (document type declaration, DTD) határozza meg. Ez a definíciós rész meghatározza a címkék neveit, azok tartalmát, lehetséges struktúráját, illetve kiegészítő részeit. Az XML szabványban ez részét képezheti egy XML dokumentumnak, illetve attól különálló dokumentumban is megtalálható. Ahhoz, hogy egy dokumentum érvényes legyen, szükséges, hogy rendelkezzen egy DTD résszel.

Mi az XML?

A legfontosabb fogalmak tisztázása után nézzük, mi is az XML!

Az XML egy metanyelv, amely segítségével strukturált jelölő nyelveket hozhatunk létre. Az XML szabvány [2] egy W3C Ajánlás (W3C Recommendation), amely részletesen meghatározza az XML nyelv szintaxisát, az XML dokumentumok logikai és fizikai felépítését, DTD-k és XML dokumentumok készítését.

Az XML, mint technológia ugyanakkor túlmutat ezen, és általánosságban egy olyan technológiai családot jelöl, amely egyrészt a web rendszereinek újfajta az XML szabványra épülő kidolgozását, másrészt az XML mint szabványos, önleíró adatformátumok létrehozására alkalmas eszköz alkalmazását az elektronikus üzleti kommunikációban. Mindkét terület rohamos léptekben fejlődik, sorra jelennek meg az XML szabványt kiegészítő, alkalmazását teljessé tévő technológiai ajánlások, szabványok, és szoftver termékek.

„Az XML szabvány” c. fejezetben az alap szabványra koncentrálunk. Annak web, illetve üzleti kommunikációs alkalmazásához kapcsolódó technológiákat a további fejezetek ismertetik.

Hogyan néz ki egy XML dokumentum?

Az XML dokumentum alapvetően egy ember által is olvasható szöveges dokumentum. A szövegben elhelyezett címkéket < és > jelek veszik körül. A szabványban rögzített, illetve a dokumentum készítője által definiált címkéket az XML ún. elemeknek nevezi. Az XML dokumentumok felépítésének teljes, minden részletre kiterjedő megismeréséhez ajánlható az annotált XML specifikáció [4].

Egy dokumentum alapvetően az XML deklarációból, egy prológusból és a dokumentum törzsből áll. A prológus tartalmazza a típus deklarációt, illetve feldolgozási utasítást, a törzs pedig a tényleges információs tartalmat.

XML deklaráció

Minden XML dokumentum az XML deklarációval kezdődik. Ez azonosítja, hogy a dokumentum XML formátumú, illetve meghatározza annak néhány alaptulajdonságát.

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

Mindhárom paraméter opcionális, de sorrendjük kötött. A verziószám a dokumentumra érvényes XML szabvány verzióját azonosítja. Az „encoding” meghatározza a dokumentumban használt karakterkészletet (lásd később), míg a „standalone” attribútum azt határozza meg, hogy a dokumentum önmagában, külső deklarációk nélkül értelmezendő.

Típus deklaráció

Az XML deklaráció után következik a dokumentum típus deklarációja a következő szerkezetben:

```
<!DOCTYPE gyökérElem [...deklarációk...]>
```

A deklarációs rész felépítését részletesen a későbbiekben ismertetjük.

Egy dokumentumban mindig egy és pontosan egy ún. gyökér elem (root element) található. Ez a dokumentum fa kiinduló csomópontja. A gyökér elem felépítését a dokumentum deklaráció határozza meg. Ez a deklaráció (az előbbi formájában) megtalálható a dokumentumon belül – ezt nevezzük belső DTD részhalmaznak, illetve azon kívül – ez a külső DTD részhalmaz. Ez utóbbi deklarációjának egy lehetséges formája a következő:

```
<!DOCTYPE gyökérElem SYSTEM “uri” [...deklarációk...]>
```

A deklarációs rész ez esetben teljesen el is maradhat, így a dokumentum deklarációs része az URI-val azonosított külső erőforrásban található meg. A külső deklaráció hozzárendelhető ún. publikus azonosítóval is a következőképpen:

```
<!DOCTYPE gyökérElem PUBLIC “publicIdentifier” “uri”
[...deklarációk...]>
```

A publikus azonosító egy szabványos azonosító, amely egyértelműen meghatározza a külső deklarációkat. Első karaktere egy + vagy – jel, attól függően, hogy a nemzetközi szabványosítási szervezet, az ISO (vagy egy általa elfogadott másik nemzetközi testület) fogadta el, avagy nem. Ez után található a DTD készítőjének a neve (pl. W3C, amely nem nemzetközi szabványosítási testület, ezért az azonosítói mínusz jellel kezdődnek), majd az azonosító neve és végül az őt leíró nyelv azonosítója.

A következő karaktersorozat a W3C HTML 4.0 szabványt azonosítja:

```
-//W3C/DTD HTML 4.0 Final//EN
```

Mivel az így meghatározott azonosítóknak nincs egyedi névfeloldási rendszere, ezért az XML szabvány megköveteli, hogy az azonosító után egy URI is álljon.

A teljes példa tehát a következő:

```
<!DOCTYPE html PUBLIC “-//W3C/DTD HTML 4.0 Final//EN”
“http://www.w3.org/TR/PR-html40/strict.dtd”>
```

Tartalom

A deklaráció után a dokumentum a gyökér elemmel folytatódik, és azon belül tartalmazza a tényleges szöveges információt a deklarációnak megfelelően jelölve.

A következő teljes példa egy dokumentum összefoglaló fájlt mutat:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<!DOCTYPE summary SYSTEM
"http://i2rt.mit.bme.hu/template/summary.dtd">
<summary>
     <doc_title>Dokumentum elemzési és keresési
módszerek</doc_title>
     <doc_type>1</doc_type>
     <topics>
          <topic_id>1340</topic_id>
     </topics>
     <doc_abstract>A tanulmány ismerteti...</doc_abstract>
</summary>
```

Egyéb részek

Az XML fájl opcionálisan tartalmazhat feldolgozási utasítást, entitás definíciókat, illetve megjegyzéseket. A megjegyzések alakja a következő:

```
<!-- tetszőleges karakterek, kivéve az egymás utáni két mínusz
jelet -->
```

A prológus részben található **feldolgozási utasítás (processing instruction, PI)** abban segít, hogy az XML leíró címkéit procedurális információkkal láthassuk el, amely valamely feldolgozó program számára szükséges. Ezek nem részei a dokumentumban található adatoknak, ugyanakkor átadódnak a dokumentumot feldolgozó alkalmazásnak. Tipikus példa egy XML stíluslap hozzárendelése a dokumentumhoz:

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

Az XML dokumentumokban alkalmazott **entitások (entity)** a többször felhasznált dokumentum szövegrészleteket látják egy rövid jelöléssel (hasonlóképpen mint a magas szintű programozási nyelvek makrói). Ezek az ún. **általános entitások (general entities)**, nevükkel is megkülönböztetve őket a később ismertetett paraméter entitásoktól. Az általános entitások definíciója a dokumentum deklarációban történik (illetve vannak előre definiált entitások), felhasználásuk pedig az XML dokumentumban az elemek, illetve attribútumok belsejében a következő formában lehetséges:

```
&entitás_név;
```

Az & jelzi az entitás kezdetét, a pontosvessző pedig annak végét. A teljes karaktersorozat helyére behelyettesítődik a definiált entitás értéke. Amennyiben a behelyettesített szövegrészlet további entitásokat tartalmaz, úgy azok is kiértékelődnek és behelyettesítődnek. Az entitásokról részletesebben a későbbiekben lesz szó.

Dokumentum deklaráció

Egy XML dokumentumban alkalmazható elemeket, illetve más összetevőket a dokumentum típus deklaráció rögzíti. Az XML DTD – a korábbiakban ismertettek szerint – megadható az XML fájl deklarációs részében, illetve egy önálló, ún. DTD dokumentumban is.

A DTD főbb részei a következők:

- elemek (element), melyek az információ fő hordozói
- attribútumok (attribute), amelyek az elemekhez rendelnek kiegészítő információkat, valamint
- entitások (entity), amelyek többször használt, ill. külső erőforrásban található szövegrészleteket rögzítenek.

Egy XML dokumentum fő információ hordozó forrásai az elemek és azok attribútumai. Számtalan mód létezik XML dokumentum struktúrák készítésére, az elemek és attribútumok definiálására: annak meghatározására, hogy milyen információt tárolunk elemekben, illetve mit attribútumban. Nincs egyetlen, tökéletes ajánlás: az alkalmazott módszert javarészt a tárolt információ, illetve annak felhasználási módja dönti el.

Elemek

A DTD-ben definiált elemek (element) határozzák meg a DTD alapján készülő XML dokumentumok nyelvtanát (azaz az alkalmazható címkéket), illetve azok lehetséges struktúrájának alapját. Az elemek egy nevekkel ellátott hierarchiát határoznak meg, amely a DTD alapján készített XML dokumentumokban a címkék segítségével jelenik meg.

Egy elem alapvetően két részből áll: névből és tartalom modellból. A név lesz az XML fájlban található címke azonosítója, míg a tartalom modell meghatározza az elem belső felépítését, a tartalom lehetséges szerkezetét (a konkrét tartalom természetesen csak az adott XML fájlban található meg).

A név betűvel, aláhúzással, vagy kettősponttal kezdődik, és betűket, számjegyeket, elválasztó karaktert, aláhúzást, valamint pontot tartalmazhat. A kis- és nagybetűket megkülönböztetik. (A betűk között megengedett a magyar ékezetes karakterek alkalmazása is.) Elemek nem kezdődhetnek „xml”, „XML”, illetve ezek kis- és nagybetűs keverékét tartalmazó betűsorozattal. A nevek választására ezen kívül nincs szabály – a szabvány a fejlesztőre bízza azok kialakítását. Fontos, hogy a választott nevek jól illeszkedjenek az alkalmazáshoz: egy jól kitalált dokumentum struktúra is használhatatlanná válhat rosszul választott elem nevekkel. A rövidítések egyes esetekben hasznosak (pl. P a paragrafus), de hamar olvashatatlanná válnak. Túl hosszú és beszédes nevek alkalmazása feleslegesen növeli a későbbi XML fájlok méretét. A nagy- és kisbetűk alkalmazására is ügyelni kell: kerülni kell használatuk felesleges keveredését. Egyes szerzők csupa nagybetűk alkalmazását javasolják – így a címkék jól elkülönülnek az XML dokumentumok egyéb szövegeitől, míg mások (pl. a legtöbb W3C szabvány, pl. az XHTML) csak kisbetűket használnak.

Egy elem tartalmazhat szöveges adatot, további elemeket, illetve lehet üres is. Ezt definiálja a tartalom modell, melynek szintaxisa a következő:

```
<!ELEMENT elem_név tartalom_modell>
```

Üres elem (EMPTY)

A legegyszerűbb tartalom modell az üres elem (empty element). Ebben az esetben a címke nyitó, illetve záró párja között nem található információ. A címke egy speciális jellel lezárva önmagában is állhat az XML fájlban:

```
<üres_elem></üres_elem>
<üres_elem />
```

Mindkét forma helyes (figyeljük meg, hogy a második esetben a / jel előtt szóköz áll). Az ilyen elemhez tartozó DTD deklaráció a következő:

```
<!ELEMENT üres_elem EMPTY>
```

Tetszőleges tartalom (ANY)

A következő lehetséges tartalom modell az ún. ANY (tetszőleges), amely nem köti meg a lehetséges tartalmat. Az ilyen elemek által meghatározott címkék közrefoghatnak más elemeket, szöveget, illetve üresek is lehetnek.

A tartalom modell definiálása a következő:

```
<!ELEMENT tetszőleges_elem ANY>
```

A következő sorok példákat mutatnak ilyen elemekre:

```
<tetszőleges_elem></tetszőleges_elem>
<tetszőleges_elem />
<tetszőleges_elem><üres_elem></üres_elem></tetszőleges_elem>
<tetszőleges_elem>szöveg<üres_elem></üres_elem>másik
szöveg</tetszőleges_elem>
```

A lehetőségek száma szinte végtelen. Bár ez egy egyszerű és nagyon szabad tartalom modell, használatát sokan mégsem javasolják, mivel a dokumentum deklaráció így nem tölti be az XML dokumentumok struktúrájának megkötésére kitalált szerepét.

Karakteres adat (#PCDATA)

Tetszőleges karakteres adatot a #PCDATA (parsed character data, elemzett karakteres adat) tartalom modellel határozhatunk meg.

A tartalom modell definiálása a következő:

```
<!ELEMENT szöveg (#PCDATA)>
```

A következő sor példát mutat ilyen elemre:

```
<szöveg>Ez egy karakter sorozat.</szöveg>
```

Az ilyen típusú tartalom modell nem engedi meg további elemek beágyazását a „szöveg” címkék közé. Ott csak karakteres adat szerepelhet.

Strukturált tartalom modellek

Az XML 1.0 szabvány speciális operátorok, szimbólumok kis halmazával lehetővé teszi egy elemen belüli struktúra meghatározását. Az alábbiakban összefoglaljuk ezeket.

Operátor	Példa	Jelentés
+	név+	a „név” nevű elem egy vagy több alkalommal szerepel
*	név*	az elem nulla vagy több alkalommal szerepel
?	név?	az elem nulla vagy egy alkalommal szerepel
nincs	név	az elem pontosan egy alkalommal szerepel (azaz nem ismételhető)
,	név1,név2	a felsorolt elemek ilyen sorrendben, egymás után szerepelnek
|	név1|név2	a két megadott elem közül az egyik, és csakis az egyik szerepel
( )	(név1,név2)|(név3,név4)	a zárójelek lehetővé teszik a többi operátor hatáskörének meghatározását. A példában vagy a „név1” majd „név2” elemek szerepelnek ilyen sorrendben az elemen belül, vagy a „név3” majd „név4” elem.

A tartalom modellben az elemek nevei, illetve a fenti operátorok kombinálhatóak egy adott struktúra létrehozása érdekében. A következőkben bemutatunk néhány deklarációs példát:

```
<!ELEMENT term (name*, definition*, link*, ref*)>
<!ELEMENT definition (#PCDATA)>
<!ELEMENT en (#PCDATA)>
<!ELEMENT hu (#PCDATA)>
<!ELEMENT link EMPTY>
<!ELEMENT ref EMPTY>
```

A következő XML fájl részlet előbbi DTD alapján készült.

```
<term>
     <name>gyakori termékhalmaz</name>
     <name>large(frequent) itemset</name>
```

```
    <definition>Egy termékhalmaz gyakori, ha ...</definition>
    <definition>The large itemset ...</definition>
</term>
```

Attribútumok

Az attribútumok (attribute) az elemekhez rendelt attribútum listában található információ hordozók. Míg az elemek alkalmasint bonyolult belső struktúrával rendelkeznek, addig az attribútumok egyszerű adattároló helyek, melyek azonban rendelkeznek típus információval, illetve esetenként értékkészlettel is.

Az attribútumok nem önálló DTD komponensek: mindig elemekhez rendeljük őket ún. attribútum lista (attribute list) segítségével. A definíció szintaxisa a következő:

```
<!ATTLIST elem_név attribútum_lista>
```

A legegyszerűbb attribútum lista az üres lista. Ilyenkor az elemhez nem tartozik egyetlen attribútum sem. Mivel alapértelmezettként ez egyébként is igaz, így ilyen deklarációkat tenni felesleges.

A nem üres lista attribútum nevek és szóközzel elválasztott típusok felsorolását tartalmazza.

Az attribútumok elnevezése az elemek elnevezéséhez hasonló elvek szerint történhet. A nevek betűvel vagy aláhúzással kezdődhetnek, és betűket, számokat, elválasztó karaktert, aláhúzást és pontot tartalmazhatnak. Természetesen ezek is megkülönböztetik a kis-, illetve a nagybetűket.

Lényeges különbség az elemek és az attribútumok között, hogy utóbbiakhoz típus rendelünk. Az XML 1.0 szabvány a típusok széles skáláját definiálja. Ezek a típusok azonban nem a programozási nyelvekben megismert adattípusok, sokkal inkább a dokumentum feldolgozás során felmerülő feladatok megoldására szolgálnak. Speciális típusú attribútumokkal lehetővé válik például dokumentum részek összekapcsolása, illetve speciális kényszerek beállítása.

A legtipikusabb típus a CDATA, azaz karakteres adat, amely kevés megkötést tartalmaz az attribútum lehetséges értékére, illetve annak felhasználására, jelentésére. Szintén tipikus az ún. azonosítók (identifier, ID), illetve az ezekre hivatkozó referenciák (IDREF, IDREFS). Ezek a dokumentumon belül érvényes egyedi azonosítókat, illetve létező azonosítókra hivatkozó referenciákat hoznak létre.

További típusok az entitás (ENTITY, ENTITIES), amellyel nem elemzett entitásokra lehet hivatkozni (ezekről később lesz szó), a tokenizált típusok (NMTOKEN, NMTOKENS), melyek hasonlóak a CDATA típushoz, de azoknál szigorúbb értékkészlet megkötéseket tartalmaznak (lényegében az attribútum, illetve az elemnévnél megismert szabályokat alkalmazzák). A tokenek értelmezése, használata teljes egészében az XML fájlt feldolgozó alkalmazásra van bízva.

Az utolsó attribútum típus a felsorolás, amely az attribútum értékkészletét rögzíti adott karaktersorozatok halmazára. Az attribútum az XML fájlban a megadott lehetséges értékek egyikét veheti fel. Felsorolás típusnál az egyszerű karakteres adaton kívül megadhatunk úgynevezett elnevezéseket is (NOTATION), melyek a DTD-ben valahol másutt deklarált elnevezésekre hivatkoznak (ezekről is később lesz szó).

Az attibútumok deklarációjának utolsó (opcionális) része az alapértelmezett érték, illetve az attribútum használati típusát meghatározó módosító jelző. Az alapértelmezett érték az attribútum lehetséges értékkészletének megfelelően meghatározza azt az értéket, amit az attribútum akkor vesz fel, ha az XML fájl elemében nem kap más értéket. Egy attribútum használati típusa lehet opcionális (IMPLIED), kötelező (REQUIRED), illetve rögzített értékű (FIXED).

A következőkben az ismertetett attribútum típusokra mutatunk példákat.

Az összetett elemeknél ismertetett példánál maradva a „link”, illetve a „ref” üres elemek voltak, illetve a „name” és „definition” elemben megadott név, illetve definíció nyelve sem volt deklarált.

Egészítsük ki ezeket az elemeket a következők szerint! A „link” és „ref” elemek egy kötelező azonosító („id”) illetve egy opcionális „href” attribútumot kapnak (amiket a dokumentumokat feldolgozó alkalmazás kezel majd).

```
<!ATTLIST link
    id        CDATA     #REQUIRED
    href      CDATA     #IMPLIED
```

```
>
```

A következőkben a „term” elemhez rendelt attribútumokat látunk (azonosító, szerző, illetve egy opcionális forrás megjelölés):

```
<!ATTLIST term
     id        ID        #REQUIRED
     author    CDATA     #REQUIRED
     source    CDATA     #IMPLIED
>
```

A „name” illetve „definition” elemekhez rendelünk egy nyelv („lang”) attribútumot, ami opcionális és rögzített értékkészlettel valamint egy alapértelmezett értékkel rendelkezik:

```
<!ATTLIST name
     lang      (hu|en)   “hu”
>
<!ATTLIST definition
     lang      (hu|en)   “hu”
>
```

A korábbi példa az attribútumok használatával így tehető teljessé:

```
<term id=”term_1212” author=”76352”>
     <name>gyakori termékhalmaz</name>
     <name lang=”en”>large(frequent) itemset</name>
     <definition>Egy termékhalmaz gyakori, ha ...</definition>
     <definition lang=”en”>The large itemset ...</definition>
     <link id=”link_87632” href=”http://www.mit.bme.hu/...” />
     <ref id=”ref_234” href=”http://...” />
</term>
```

Entitások

Az XML dokumentumok felépítésénél az újra felhasználható tartalom készítésére alkalmas általános entitásokat már ismertettük. A DTD is rendelkezik egy speciális entitás típussal, az ún. paraméter entitással, amelyik a DTD belsejében definiálhat újrahasználható tartalmat. Mielőtt ezt ismertetnénk, nézzük az általános entitás deklarációs formáját:

```
<!ENTITY entitás_név „érték”>
```

Az entitás neve betűvel vagy aláhúzással kezdődhet és betűket, számjegyeket, elválasztó karaktert, aláhúzást és pontot tartalmazhat. Egy entitás értéke lehet tetszőleges XML tartalom (azaz címkékkel ellátott szöveg), de meg kell felelnie az XML szabványnak, azaz a címkéket helyesen kell alkalmaznia (jól formázottnak kell lennie). Ez garantálja, hogy a dokumentumokban helyettesítve őket a dokumentum jól formázott maradjon.

Definiálhatunk külső entitásokat is, melyek értéke egy megadott külső erőforrásban (pl. fájlban) található. Ezek definíciója az XML dokumentum típusdeklarációjánál megismert SYSTEM, illetve PUBLIC kulcsszavakkal történhet:

```
<!ENTITY entitás_név SYSTEM „entitás_URI”>
```

illetve

```
<!ENTITY entitás_név PUBLIC „entitás_publikus_azonosító”
„entitás_URI”>
```

A dokumentum típus deklarációkban alkalmazott paraméter entitások definiálásának szintaxisa alig tér el az általános entitásokétól:

```
<!ENTITY % entitás_név „érték”>
```

A százalékjel jelzi, hogy paraméter entitásról van szó, amit a DTD-ben fogunk alkalmazni. A paraméter entitás deklarációkat, illetve azok részletét tartalmazhatja, melyek egyszerűsíthetik a DTD készítését.

A paraméter entitások külön névtérben helyezkednek el az általános entitásoktól (hiszen másutt és más célra használjuk őket), így azonos nevű paraméter és általános entitások használata megengedett. A paraméter entitások nevének kiválasztása hasonló szabályok szerint történik, mint az általánosoké, tartalmuk azonban kevésbé korlátozott. Egy paraméter entitás tartalmazhat egy vagy több teljes deklarációt, illetve egy deklaráció egy részletét. Nem kezdődhetnek, illetve fejeződhetnek be tehát két deklaráció közepén.

Paraméter entitás is lehet külső, a SYSTEM, illetve PUBLIC kulcsszavakkal meghatározott erőforráshoz rendelt. Ez a paraméter entitások egy tipikus felhasználása: ilyenkor a dokumentum deklaráció felhasznál egy másik, külső fájlban deklarált dokumentumot.

```
<!ENTITY % entitás_név SYSTEM „entitás_URI”>
```

A paraméter entitások hasonlóképpen illeszthetők be a DTD megfelelő helyére, mint az általános entitások az XML dokumentumba:

```
&entitás_név;
```

Az & jelzi az entitás kezdetét, a pontosvessző pedig annak végét. A teljes karaktersorozat helyére beillesztődik a definiált szöveg, illetve külső erőforrás. A beillesztés után a DTD-t feldolgozó program ugyanúgy jár, mintha a beillesztett részlet eredetileg ott lett volna: elemzi azt és feldolgozza az ott talált deklarációkat.

A paraméter entitások a lehetőségek széles körét tárják a DTD készítők elé. Van azonban néhány fontos dolog, amit szem előtt kell tartanunk. Az XML elemzők az entitás behelyettesítése után egy szóközt illesztenek a behelyettesített szöveg végére, így az alábbi példa hibásan fog működni:

```
<!ENTITY % dátum „év, hó, n”>\
<!ELEMENT dátum (%dátum;ap)>  <!-- tilos, nem lesz nap belőle -->
```

Elnevezések és nem elemzett entitások

Az elnevezések (notation) lehetővé teszik típusok (fájl típusok, adattípusok, stb.) deklarációját. Deklarációjuk hasonló a külső entitásokéhoz:

```
<!NOTATION név SYSTEM „URI”>
```

illetve

```
<!NOTATION név PUBLIC „azonosító” „URI”>
```

Eltérően minden más külső hivatkozástól, az elnevezések deklarációja megengedett URI azonosító nélkül is:

```
<!NOTATION név PUBLIC „publikus_azonosító”>
```

Ennek oka, hogy a megadott URI nem az erőforrás beszerzése céljából van, hanem csak azonosítja az (amihez az azonosító is elegendő). Az URI, illetve a publikus azonosító, illetve az elnevezés által definiált típus a dokumentumot olvasó ember számára nyújt elsősorban információt, illetve a fájlt feldolgozó alkalmazás használhatja ezeket. Az XML elemzők számára csak azt jelzi, hogy például egy vele ellátott entitás nem elemzett.

A nem elemzett entitások (unparsed entity) lehetővé teszik nem XML alapú tartalomra történő hivatkozást. Ezen entitásokat kiegészítjük egy NDATA résszel, ahol egy korábban definiált elnevezést adunk meg:

```
<!ENTITY név SYSTEM „URI” NDATA elnevezés>
```

Például:

```
<!ENTITY kép SYSTEM „kép1.jpg” NDATA image/jpeg>
```

Elnevezéseket felhasználhatunk attribútumokban is a NOTATION típus megadásával:

```
<!ATTLIST kép
     href      CDATA     #REQUIRED
     típus     NOTATION  (image/gif | image/jpeg) #REQUIRED
>
```

Ezen deklaráció alapján egy kép elem két attribútummal rendelkezik, például:

```
<kép href=”kep1.jpg” típus=”image/jpeg” />
```

Ez az elemző számára nem sokat mond (azon kívül, hogy szintaktikailag helyes), azonban a fájlt feldolgozó alkalmazás a típus információt felhasználhatja a kép megjelenítésére.

Szóközök és egyéb szeparáló karakterek

A szóközök és egyéb „üres” karakterek (white space) használata sok bonyodalmat okoz az XML dokumentumokban, ezért röviden kitérünk rájuk. Az ilyen karakterek közé tartozik a szóköz (space), soremelés (line break), sor eleje (carriage return), és a tabulátor (tab) karakter. Ezeket tipikusan a dokumentumok formázására használjuk. Bár a legtöbb esetben használatuk (illetve jelenlétük) nem okoz gondot, azonban figyelnünk kell bizonyos szabályokra, illetve az elemzők tulajdonságaira.

Ahol az üres karakterek kötelezőek (különösen a deklarációkban, például elválasztandó az attribútumok nevét, típusát és alapértelmezett értékét), ott meg kell jelenniük – bármelyiküknek. Van, ahol használatuk tiltott (például a címkék neve és a nyitó < jel között, illetve üres elemek belsejében). Más elemeket tartalmazó elemek belsejében még akkor is megengedett az üres karakterek használata, ha az elem szabad szöveget nem tartalmaz (azaz csak elemet).

Legnehezebb akkor a helyzet, amikor egy elem belsejében vegyes tartalom van (MIXED). Ilyenkor az XML feldolgozók minden üres karaktert külön feldolgoznak, még akkor is, ha azok a dokumentum készítőjének szándéka szerint csak formázás miatt kerültek a szövegbe. Ez alól a soremelés jel a kivétel, amit az alkalmazás figyelmen kívül hagyhat. Az XML 1.0 szabvány egy speciális attribútumot, az xml:space nevűt ajánla annak jelzésére, hogy egy elem (és az összes benne található további) hogyan tartalmaz üres karaktereket:

```
<!ATTLIST c-kód
     xml:space (default | preserve)     ‘preserve’
>
```

Karakterkészlet, nyelv

Az XML 1.0 szabvány az „Universal Character System” (ISO 10646) nemzetközi szabványt használja a dokumentumokban használt karakterkészlet leírására. Ez a szabvány magába foglalja az Unicode 2.0 formátumot [5] is. Általában csak ez utóbbit alkalmazzák a dokumentumok.

Az XML 1.0 nem használja az összes Unicode karaktert, egyes karakterek (leginkább a 32 kódszám alatti vezérlő karakterek) használata kifejezetten tiltott.

Az XML feldolgozók számára kötelező az UTF-8 és UTF-16 formátumok elfogadása, ugyanakkor opcionális számos más (pl. a magyar karakterek is tartalmazó ISO-8859-2) karakterkészlet használata.

A dokumentum, pontosabban egy elem nyelvének meghatározására az xml:lang attribútum használható (hasonlóan az xml:space attribútumhoz).

```
<!ATTLIST előadás
     xml:lang NMTOKEN #IMPLIED
>
```

Jól formázott és érvényes XML dokumentumok

Az XML 1.0 szabvány kétféle megfelelést definiál: jól formázott (well-formed) és érvényes (valid) dokumentum.

Egy jól formázott dokumentum megfelel a szabványban rögzített összes szintaktikai szabálynak, de nem köteles megfelelni a dokumentum típus deklarációnak. Egy érvényes dokumentum megfelel a jól formázottság összes szabályának és ezen kívül a dokumentum deklarációban meghatározott struktúra követelményeinek is.

Az XML és az SGML viszonya

A két technológia viszonyát leegyszerűsítve megállapíthatjuk, hogy az XML az SGML egyszerűsített változata. Általánosságban elmondható róla, hogy az SGML szabvány azon részeit nem vette át, amelyek általában igen komplexek, esetiek és ritkán használtak. A leegyszerűsítés

előnye, hogy XML dokumentumok egyszerűbben készíthetőek és könnyebb a feldolgozásukra képes programot írni. Az egyszerűsítések mögötti célkitűzés az volt, hogy könnyítsék az alkalmazások és dokumentumok készítését, azok elterjedését a web rendszereiben.

Az egyszerűsítések ellenére az XML nagyon közel áll az SGML-hez. Az XML dokumentumok feldolgozhatóak SGML rendszerekkel, az SGML szövegszerkesztő, megjelenítő, tároló és feldolgozó eszközök alkalmasak ilyen dokumentumok kezelésére.

A különbségeket részletesen az [1] irodalom tartalmazza.

A W3C tevékenysége az XML technológia területén

A W3C tevékenységét az XML és kapcsolódó szabványainak kialakításában részletesen a W3C XML Activity Statement [3] ismerteti. A W3C tevékenysége során létrehozta az XML szabványt [2], majd ehhez kapcsolódóan az XML Namespaces, illetve az XML Style Sheet, valamint XML Linking specifikációkat. Ezt követően elkezdték egy XML Protocol kidolgozását is.

A W3C tevékenysége az elkövetkező években elsősorban a nagyszámú XML alkalmazás készítése során felmerült problémák, a különböző szabványok közötti kisebb inkonzisztenciák, illetve a már folyamatban levő szabványosítás befejezésére koncentrálódik.

A tevékenységi körök alapvetően a következő munkacsoportokba szerveződnek:

- Az **XML Core Working Group** az XML 1.0 specifikáció karbantartásával foglalkozik. Ezen kívül az XML Syntax, XML Fragment Interchange, XML Inclusion, és az XML Information Set specifikációk kidolgozása is a feladatai közé tartozik.
- Az **XML Schema Working Group** feladata a dokumentumok struktúrájának, tartalmának és szemantikájának kidolgozása. A munkacsoport kidolgozott egy követelményrendszert XML-alapú dokumentum típusok definiálására, illetve elkezdte az XML Schema specifikáció kialakítását.
- Az **XML Linking Working Group** XML dokumentumokban ágyazható hipertext linkek tervezésével foglalkozik. Általuk kidolgozott specifikáció az XML Pointer Language (XPointer), az XML Linking Language (XLink), illetve az XML Base.
- Az **XML Query Working Group** elsődleges feladata XML-alapú web dokumentumok hatékony lekérdezését biztosító technológiák kidolgozása. A munkacsoport az XML Query Requirements dokumentumban összefoglalta az ilyen megoldásokkal szemben támasztott követelményeket, illetve az XPath és XQuery specifikációkat.
- Végezetül az **XML Coordination Group** feladata az XML terület munkacsoportjainak koordinálása.

Hivatkozások

[1] W3C “Comparison of SGML and XML”, http://www.w3.org/TR/NOTE-sgml-xml-971215

[2] W3C “Extensible Markup Language (XML) 1.0”, http://www.w3.org/TR/REC-xml

[3] W3C “Extensible Markup Language (XML) Activity Statement”, http://www.w3.org/XML/Activity

[4] Tim Bray „Annotated XML Specification”, http://www.xml.com/axml/testaxml.htm

[5] “Unicode 2.0”, Unicode Consortium, http://www.unicode.org