2. Az XML alapjai

Az XML-elemek az adatokat értelemmel bíró összetevőkre osztják, és a szabvány alapvető részeit alkotják. A szabvány fennmaradó funkciói más perifériku szabványokkal együtt támogatószerepet játszanak, emellett az összetevők finomítására, annotálására, tárolására, vezérlésére, keverésére, kapcsolására, feldolgozására és formázására összpontosítanak.

Bevezetés

Az „XML" megnevezés az **„Extensible Markup language" (Kiterjeszthető jelölőnyelv)** kifejezés rövidítése (az „E" helyett jobban mutat az „X"). Ez a nyelv egyik kereskedelmi érdekeltséghez sem tartozik. A **W3C** (a ***World Wide Web Konzorcium***) fejlesztette ki, majd a korábbi jelölőnyelveken alapuló tapasztalat eredményeként fejlődött.

Az XML alkalmazási lehetősége szinte korlátlan. Két fő felhasználási területe azonban az adatcsere-formátum és a dokumentumszerkesztési formátum. E két alkalmazási mód gyakran kiegészíti egymást, mivel az adatok emberek által történő felhasználásra is formázhatók.

Az XML-adatformátum első ránézésre túlságosan leegyszerűsítettnek tűnhet, és az alapjait tényleg percek alatt elsajátíthatjuk. Azonban számos megszorítás és rejtett komplikáció van (ezekről a későbbi fejezetekben ejtünk szót). Ennek ellenére minimális XML-lel is sok mindent megtehetünk, és ráérezhetünk a nyelvre, ha röviden átnézzük a hátterét, néhány alapvető elvet pedig részletesen tanulmányozunk.

Szöveg

Az XML-adatformátumot mindig valamilyen mögöttes szövegformátum támogatja. Az XML tehát kiaknázza azokat a hatalmas előnyöket, amelyeket e megközelítés nyújt az információcsere számára. Az **ASCII** szöveg a szövegalapú információtárolás és -átvitel szinte egyetemes formátuma. Másolhatunk szöveget egyik számítógépes alkalmazásból egy másikba, hálózatokon keresztül vihetjük át már régóta a rendelkezésünkre álló eszközök segítségével. Bár az ASCII nem az egyetlen rendelkezésre álló szövegformátum, a legtöbb alternatíva csupán ennek a formátumnak egy jelentéktelenebb változata vagy továbbfejlesztése (további részletekért lásd a 29. fejezetet). Az alábbi újsághírt bármilyen szövegszerkesztővel könnyedén létrehozhatjuk, lemezre menthetjük, majd bármely más szövegszerkesztővel megnyithatjuk és módosíthatjuk:

```
            XML Standard Released
The XML standard was released today by the W3C.
This is an important new standard for data exchange
and document publishing ...
```

A szövegformátumok az egyszerűségüknek köszönhetik az általános támogatást. E túlzott egyszerűség azonban egyben az egyik legfőbb gyenge pontjukat is felfedi: az olyan formátumok, mint például az ASCII, nem képesek az adatokat jelentéssel bíró részekre bontani, legalábbis anélkül nem, hogy az félreérthetőséghez ne vezetne. Egy bekezdés végét például

jelezheti sorvégi kód, vagy esetleg egymás után következő sorvégi kódokra van szükség (üres sorokat létrehozva), hogy ezt a jelentést kifejezzük. A címsort jelölheti kezdő szóközökkel történő középre helyezés (a fenti ábra szerint) vagy nagybetűs szedés.

Jelölés

Az ASCII szöveg nyújtotta lehetőségeket kiterjeszthetjük azáltal, hogy egyszerűen további jelentésréteget adunk hozzá. Ezt úgy tehetjük meg, hogy a szövegben jelentést rendelünk bizonyos karakterekhez vagy karaktersorozatokhoz.

A vessző és a sorvégi kódok például jelentéssel bírnak a CSV-formátumban. Ezt a szabványt használjuk a beágyazott információegységek szövegsoron belüli egyértelmű elszigeteléséhez, egyszerűen úgy, hogy minden elemet vesszővel választunk el a szomszédaitól. Közöttük a sorvégi kódok és vesszők rácsos struktúrát hoznak létre. Ez tehát ideális formátum adatbázis-táblázatok vagy táblázatkezelők közt cserélendő táblázatos információk ábrázolására:

```
SGML standard released,1986,ISO
HTML 4.0 standard released,1997,W3C
XML 1.0 standard released,1998,W3C
```

E CSV-dokumentum táblázatba importálásával az alábbiakat hozhatjuk létre:

SGML standard released	1986	ISO
HTML 4.0 standard released	1997	W3C
XML 1.0 standard released	1998	W3C

A CSV-ben a vesszők és a sorvégi kódok nem részei az adatnak. Ezeket az adatkezeléshez használt **jelöléskonstruktumoknak** tekintjük. A 31. fejezet háttérinformációt nyújt az XML tervezését befolyásoló jelölőformátumokról, valamint a dokumentumformázással kapcsolatos megközelítésről.

A CSV és a hasonló alternatívák gyenge pontjai nyilvánvalóak. Egyrészt csak táblázatos információt tudunk ábrázolni, így minden sornak ugyanazt a típusú információt kell tartalmaznia. Másrészt minden oszlopnak előre meghatározott céllal kell rendelkeznie, így ha megismételhető elemekre van szükség, a megjelenés maximális számát előzetesen meg kell határozni. Ugyanígy, mivel az oszlopok elrendezése is kötött, a bejegyzések sorrendje nem változhat, ezért sohasem tekinthető jelentéssel bírónak. Végül, az adatfájlban az egyes oszlopok jelentése nincs azonosítva, így azt az adat címzettjei félreértelmezhetik, vagy idővel esetleg elfelejthetik.

E kérdések némelyikére választ adnak más jelölőnyelvek (például az **RTF** (*Rich Text format*)), amelyek a nyers adatok egyszerű átvitele helyett inkább a dokumentumok megjelenítését helyezik előtérbe. A jelölőnyelvnek ez a típusa karaktersorozatok, úgynevezett **jelölőelemek** segítségével nyújt többletinformációt az érintett adatokról. Az alábbi RTF példában félkövéren szedett szavakat tartalmazó bekezdést azonosítunk a „\par” és a „\b” jelölőelemek segítségével:

```
\par Ez a bekezdés félkövér szöveget is \b tartalmaz.
```

Az XML – melyet szokás „az általános okos ASCII” formátumként aposztrofálni – igen gyakran használ jelölőelemeket. Miközben könnyen lemásolja a CSV (bár nem annyira kompakt módon) és az RTF funkcionalitását, az előbbiekben felvázolt összes gyengeséget megcélozza.

XML-dokumentumok

A **dokumentum** kifejezést főleg adatobjektumokkal vagy fájlokkal kapcsolatban használjuk, mivel az XML olyan korábbi szabványokon alapul, amelyeket elsődlegesen elbeszélő szöveg közzétételre való előkészítésére alkalmaztak. Ez a terminológia még ma is használatos, annak ellenére, hogy amikor az XML-t szoftveralkalmazások közti adatcserére használjuk, sohasem tároljuk adatfájlban, és sohasem tesszük közzé vagy mutatjuk be más módon az embereknek. Az egyszerűség kedvéért a könyvben a „dokumentum” kifejezést ezért az XML-adatobjektumok bármilyen módon történő tárolására vagy cseréjére használjuk.

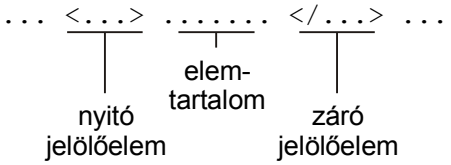
Természetesen még mindig gyakori, hogy az XML tárolt adatfájlokban jelenik meg, bizonyos esetekben pedig ezek a fájlok terjesztésre és közzétételre szánt elbeszélő szöveget tartalmaznak. Az XML-nek és elődeinek e „hagyományos” használata napjainkban is fontos alkalmazás, a könyv sok példájának szolgál alapjául.

Az XML lehetővé teszi, hogy a dokumentumokat olyan kisebb, jelentéssel bíró **elemekre** osszuk fel, amelyek felismerhetők és szükség esetén egyéni egységekként feldolgozhatók. Ez az elv áll az XML-szabvány középpontjában. Minden XML-dokumentumnak tartalmaznia kell legalább egy elemet, és az első elem mindig azonosítja és körülveszi a teljes dokumentumot. A dokumentumok zöme azonban több elemet is tartalmaz. Valójában az XML-dokumentumok gyakran nem állnak másból, kizárólag elemekből vagy elemek és szöveg keverékéből.

Hordozóelemek

A **hordozóelem** kifejezést olyan elem leírására használjuk, amely körülveszi az általa azonosított adatot. Az ilyen elem három részből áll: a **nyitó jelölőelemből**, a **záró jelölőelemből** és a jelölőelemek közötti részből. Ezt a részt **elemtartalomnak** nevezzük. Az elemtartalmat a két jelölőelem azonosítja és határolja is.

A nyitó jelölőelemet a körülötte lévő „<„ és „>„ jelölő karakterek azonosítják, a záró jelölőelem pedig nagyon hasonló, de a „</” sorozattal kezdődik. A tartalmat egyszerűen a két jelölőelem közti adat alkotja:



Elemnevek

A CSV-hez hasonló nyelveket ért kritika szerint ezek a nyelvek nem tartalmaznak információt az egyes információegységek jelentéséről. Máshol kell magyarázatot találni, vagy az adat elemzésére kell hagyatkozni a jelentés meghatározásához. Egyik megközelítés sem igazán kielégítő, ha összetett, hosszú élettartamú, vagy nagymértékben szétszórt adatokról van szó. Az XML-dokumentumokban a nevek magába az adatba ágyazódnak be, így sohasem vesznek el.

Az elemnevet gyakorlatilag a hordozóelem nyitó és záró jelölőeleme is tárolja. Ahogyan az alábbi példa is mutatja, egy adatformátum nevét a Név elembe zárva is azonosíthatjuk. Ebben a példában a Névelem tartalma az „XML” szöveg:

```
Az <név>XML</név> szabványt ma adták ki...
```

Az XML nyújtotta legnagyobb szabadság, amit a hagyományos jelölőnyelveken túl lehetővé tesz talán az, hogy nincsenek előre meghatározott elemnevek. Az XML-szabvány nem definiál „Name” elemet. Ehelyett a dokumentum szerzői céljaiknak megfelelő neveket alkothatnak. Amikor **XML-alkalmazásról** beszélünk, az XML egy adott használatát vitatjuk meg, amelyhez számos elemnevet definiáltunk. A WML például XML-alkalmazás, mely pl. definiál egy „card” elemnevet az alkalmazásra jellemző célból.

Az elemnév hossza nincs korlátozva, attól a nyilvánvaló ténytől eltekintve, hogy legalább egy karakterből kell állnia (bár a 6. fejezetben megvitatunk néhány javaslatot). A névben megengedett karaktereket illetően azonban van némi megszorítás. Az elemnévnek betűvel, alsó vonás karakterrel („_”), vagy kettősponttal („:”) kell kezdődnie (habár a kettőspont használatára vannak korlátozások), ezenfelül tartalmazhat számjegyeket és néhány más központozási karaktert („.” és „-”). Az érvényes nevek közé tartozik a „P”, az „X:123” és az „egyNagyonHosszúElemnév”.

A sorrend jelentése

Előfordulhat, hogy az utasításokat szigorúan követni kell, és az elbeszélő szövegnek (például egy könyv bekezdései) meg kell tartania az eredeti sorrendet, hogy értelme legyen. A szekvenciális környezet tehát rendkívül fontos. Szerencsére a szövegalapú adatfájlok tartalma rendelkezik implicit sorrenddel. Az adatot **adatfolyamként**, karakterek sorozataként dolgozzuk fel a dokumentum elejétől kezdve. A következő példák ezt az elvet szemléltetik. Az utasítások és a bekezdések is rendelkeznek implicit sorrenddel, amelyet az XML gond nélkül fenntart:

```
<művelet>kulcs a zárba</művelet>
<művelet>kulcs elfordítása</művelet>
<művelet>ajtó nyitása</művelet>
<művelet>be az ajtón</művelet>
<bekezdés>A következő bekezdésnek nincs értelme ezen
bekezdés nélkül.</bekezdés>
<bekezdés>Az előző bekezdés mondja el, mire való ez a
bekezdés. </bekezdés>
```

Ez ugyan triviálisnak és nyilvánvalónak tűnhet, ám bizonyos alternatív adattárolási technológiáknak (például a relációs adatbázisoknak) nem könnyű fenntartaniuk az egyéni dokumentum-összetevők sorrendjét.

Üres elemek

Az elemeknek nem kell hordozóknak lenniük, **helyfoglalókként** is funkcionálhatnak. A fontosabb dokumentumfunkciókat meghatározott pontokhoz kapcsolhatják a szövegben. Olyan értelemben foglalják a helyüket, hogy a szövegben az elemet megelőző szerkesztések a szöveg egyik oldalát tekintve sincsenek hatással az elem helyére.

Előfordulhat például, hogy a dokumentum közzétételekor oldaltörést kell beillesztenünk két adott szó közé, a töréspontot üres helyfoglaló elem jelölheti:

```
Az oldal itt fejeződik be <oldaltörés></oldaltörés> és a
következő oldal itt kezdődik...
```

Szöveg töréspont elé történő hozzáadása vagy eltávolítása miatt még nem kerül az oldaltörés más, kevésbé alkalmas helyre. Az első példa alább azt szemlélteti, hogyan kell az oldaltörésnek a „be” és az „és” szavak között megjelennie. A második példa azt igazolja, hogy az előtte álló szöveget érintő szerkesztések erre a törésre nincsenek hatással:

```
Az oldal itt fejeződik be <oldaltörés></oldaltörés> és a
következő oldal itt kezdődik...

A módosított oldal itt fejeződik be <oldaltörés>
</oldaltörés> és a következő oldal itt kezdődik...
```

Bár a jelölőelem nyitó és záró részének használata megengedett az üres helyfoglaló elemek esetében, semmi sem indokolja két jelölőelem használatát. Egyrészt nincs tárolandó adat, a záró jelölőelem tehát felesleges. Másrészt a záró jelölőelem jelenléte félrevezető, mivel azt sugallja, hogy értelmesen szöveget illeszthetünk ebbe az elembe. Az **üres elem** jelölőelem tömörebb és jobb alternatívát kínál. Ez egy „/>„-vel végződő jelölőelem:

```
Az oldal itt fejeződik be <oldaltörés/> és a következő
oldal itt kezdődik...
```

Speciális karakterek

Magától értetődik, hogy a relációs karakterek fontos szerepet játszanak az XML-jelölésben. Ha a dokumentum szövegében is megjelennének, ez megzavarná a dokumentumot olvasni és értelmezni próbáló szoftvert. Más programnyelvek részleteiben például gyakran megtalálhatók az alábbi karakterek:

```
<kód>if ( x < y ) { ... } </kód>
```

Az ilyen félreérthetőség elkerülése érdekében a jelentéssel bíró karaktereket biztosabb ekvivalensekkel kell helyettesítenünk, ha adatkarakterként használjuk őket. Ezt hagyományosan egy speciális karaktersorozattal, az úgynevezett **vezérlőkóddal** értük el (bár az XML-szabvány nem használja ezt a terminológiát). Az XML-ben az „**<**” (less than – kisebb mint) kód jelöli a „<„ karaktert, és a „**>**” (greater than – nagyobb mint) kód fejezi ki a „>„ karaktert.

Vizsgáljuk meg az XML-jelölést magyarázó, és ezért a szövegben példaként XML-jelölőelemeket tartalmazó XML-dokumentum létrehozásának problémáját. Egy elem kezdő jelölőelemének ábrázolásához a következő kódokra van szükség:

```
A &lt;név&gt; jelölőelem egy nevet határoz meg.
```

A szöveg megjelenítésekor a kódok visszaalakulnak az általuk jelölt karakterekké. A felhasználó XML-jelölőelemet lát:

A **<név>** jelölőelem egy nevet határoz meg.

A szoftverrészlet tehát a következőképpen van helyesen kódolva:

```
<kód>if ( x &lt; y ) { ... }</kód>
```

Jegyezzük meg, hogy az **XML-érzékeny** dokumentumszerkesztő a szerző helyett elvégezheti ezt a behelyettesítést. Az ilyen szerkesztők hasonlóak a hagyományos szövegszerkesztőkhöz, azonban megértik az XML-adatformátumot, és folyamatosan különbséget tesznek az adat és a jelölés között. El tudják rejteni ezt a pontot a szerző elől, aki a karaktereket a szokásos módon viheti be (és más módon is kényelmesen létrehozhat jelölőelemeket). Annak ellenére azonban, hogy ezek a karakterek megjelennek a képernyőn, a dokumentum mentésekor a szerkesztőknek a megfelelő kódokat ki kell írniuk a fájlba:

E kódok használata a problémára rögtön egy másik példát is felvet. Az „és” jel karakter („&”) ekkor fontos jelölőkarakter is. A megoldás ugyanaz. Ha a szövegben szükség van „és” jelre, annak *önmagát* kell vezérelnie, így az „**&**” (ampersand -”és” jel) kód jelöli:

```
<kód>if (( x &lt; y ) &amp;amp; ( y &lt; z ))
{...}</kód>
```

if ((x < y) & (y < z)) {...}

Elemzés

Az XML-adatot olvasó szoftvernek különbséget kell tennie a jelölés és a dokumentum tényleges tartalma közt. Meg kell értenie a jelölőkarakterek és a vezérlőkódok jelentőségét. Az adatok ilyen módon történő értelmezésének folyamatát **elemzésnek** nevezzük.

Az elemzés elvégzése hibákat fedhet fel a dokumentumjelölésben. A dokumentumok validálása a művelet végrehajtásának egyik fő oka. Olykor az **elemzőnek** mindössze annyi szerepe van, hogy elvégezze ezt a validálást. Az elemző azonban egy nagyobb alkalmazás modulja is lehet, és arra is használhatjuk, hogy a kért információt a dokumentumból az alkalmazás többi részébe továbbítsa.

Azt az elemzőt, amelyre a jelöléseket érintő alapvető ellenőrzések elvégzésére van szükség, **jól formázottság** elemzőnek nevezzük. Az alábbi XML-részlet nem jól formázott, mivel rossz relációs jelet használunk a kezdő jelölőelem teljesítéséhez, a szövegben pedig nem vezérelt „és” jel karaktert használunk. Ezeket a hibákat az elemző felismerné:

```
<bekezdés< Ez érvénytelen xml adat & ezért illegális.
/bekezdés>
```

Az elemző valójában az **XML-feldolgozó** egyik összetevője, amely entitáskezelést is végez (lásd 4. fejezet), habár előszeretettel használják az „elemző” szót a teljes csomag leírására (a könyvben mindvégig az XML-feldolgozó helyett használjuk).

Elemtípusok

Előfordulhat, hogy több elem ugyanazon a néven többször is előfordul a dokumentumban. Az alábbi szövegrészben a **Név** elem háromszor fordul elő:

```
Az <név>XML</név> szabványt ma adták ki.
A korábbi <név>SGML</név> és
<név>HTML</név> szabványokon alapul.
```

Mindhárom előfordulás ugyanazon **elemtípustól** származik. Egyszerűen *minden* azonos néven futó elem (egy dokumentumon belül) egy elemtípushoz tartozik. Adott típusú elem minden megjelenése annak az elemtípusnak egy **példánya**. Az alábbi példában a Név elemtípus három példányát láthatjuk.

A korábbi jelölőnyelvekkel ellentétben (az SGML kivételével) az XML valójában egy elemtípust sem definiál előre. Ebben az értelemben az XML-névben az „eXtensible” (kiterjeszthető) némileg félrevezető, hiszen nem létezik olyan lista az elemtípusokról, amelyet kiterjeszthetnénk. Ehelyett az elemtípusokat úgy választják ki, hogy megfeleljenek egy adott XML-alkalmazás igényeinek.

Az elemnevek érzékenyek a kis- és nagybetűkre, a „név”, a „NÉV” és a „Név” tehát más-más elemtípusra utalna. Következésképpen a záró jelölőelemben megjelenő névnek pontosan ugyanannak kell lennie, mint a jelölőelem kezdő részében szereplő névnek. Bár a kis- és nagybetűkre való érzékenységnek köszönhetően különböző elemtípusokat definiálhatunk és használhatunk ugyanazzal a névvel (például „Név” és „név”), ez mégsem ajánlott, mivel csak kavarodást és számottevő hibalehetőséget eredményezne. Íme egy másik példa jól formázottsági hibára:

```
<rosszelem>EZ HIBÁS</ROSSZelem>
```

Figyelem: Az olvashatóság érdekében az elemnevek a könyvben mindig vegyesen kis- és nagybetűvel szedve jelennek meg, és általában nem felelnek meg a példákban bemutatott szedésnek. Ilyen esetekben a példák mutatják a helyes használatot. Ennek a megkülönböztetésnek akkor van jelentősége, amikor XML-en alapuló szabványokat vizsgálunk. Ha a szövegben „Xyz” áll, a példában pedig „<xyz>„ a „xyz” a helyes használat.

Megfelelő elemtípusok

Az XML-adatformátum legfőbb erőssége, hogy **önleíró** formátum. Ez a gyakorlatban egyszerűen annyit jelent, hogy az elemek általában a tartalmukat leíró névvel rendelkeznek. Az elemek önmagukat magyarázzák. „Név” elemtípusnak csak akkor kell léteznie, ha a dokumentumban neveket kell azonosítani. Még ebben az esetben is talán jobbak a specifikusabb alternatívák, mint például a „SzerzőNév”.

Az idézetet tartalmazó XML-dokumentum olyan elemtípussal rendelkezne, mint a Forrás és maga az idézet:

```
<idézetSzövege>A legbiztosabban úgy csinálhatsz majmot
valakiből, ha idézed őt!</idézetSzövege>

<forrás>Ismert idézetek – 123. oldal</forrás>
```

Az újsághíreket tartalmazó XML-dokumentum tartalmazna például Kelt és szerző elemtípust:

```
<szerző>J. Smith, fő tudósító</szerző>
```

Amikor minden információt félreérthetetlenül azonosítottunk, az adott közönség számára releváns dokumentumrészeket választhatunk és vonhatunk ki.

Most vizsgáljunk meg egy használati utasításból származó bekezdést, amely az északamerikai és brit közönségre vonatkozó információt tartalmaz:

```
<para>The <us>color</us><gb>colour</gb> green is used on
buttons in ACME <us>elevators</us><gb>lifts</gb> to
indicate the <us>first</us><gb>ground</gb> floor.</para>
```

> *Észak-amerikai változat*: The **color** green is used on buttons in ACME **elevators** to indicate the **first** floor.
>
> *Brit változat*: The **colour** green is used on buttons in ACME **lifts** to indicate the **ground** floor.

Az olyan nevek használatát, amelyek a tartalom jelentése helyett a megjelenést jellemzik, nem javasolják XML-körökben. Akik ismerik például a HTML-t, felismernék a „B” (félkövér) és az „I” (dőlt) jelölőelem neveket, amelyek semmit sem mondanak a tartalomról, kivéve, hogy hogyan kellene megjelennie Web böngészőn történő megjelenítéskor:

```
Ki tudja, hogy <B>ez</B> miért olyan fontos?
```

> Ki tudja, hogy **ez** miért olyan fontos?

Dokumentummodellezés

Amikor több dokumentum hasonló vagy azonos struktúrával rendelkezik, természetesen ugyanazokat az elemtípusokat kellene tartalmazniuk. Emellett olyan dokumentumcsoportnak kellene tekintenünk ezeket, amely ugyanazt a dokumentumosztályt vagy **dokumentummodellt** definiálja és ugyanannak felel meg.

Minden újsághír tartalmazna például egy helyet, egy forrást és az író nevét, így mindannyian ugyanannak a modellnek felelnének meg. A dokumentummodell XML-alkalmazás, például az XMLNews-Story technikai specifikációját nyújtja. Ez a modell számos elemtípust definiál, többek közt az alábbiakat:

- Hír (újsághír)
- Törzs (az újsághír törzse)
- Cím (az újsághír címe)
- Szerző (az újsághír írója)
- Kelt (az újsághír benyújtásának dátuma).

Az idézetekre is hasonlóan definiálhatunk modellt. E modellben az alábbi elemtípus-definíciók szerepelhetnek:

- Idézet
- IdézetSzövege
- Forrás
- SzerzőNév
- PublikációNév.

Modellezési szabályok

Kifejleszthetünk olyan szoftvert, amely értelmezi a modellnek megfelelő összes dokumentumot, valamint stíluslapokat hozhatunk létre a dokumentumok képernyőn való megjelenítésére vagy nyomtatásra történő formázásához. Azonban a programok megszakadnak, a stíluslapok pedig kudarcot vallanak, ha az általuk feldolgozott dokumentumok nem felelnek meg a várt és megértett modellnek. Ezért rendkívül fontos, hogy minden dokumentum, amiről azt tartják, hogy megfelel az adott modellnek, le legyen tesztelve, és meggyőződhessünk róla, tényleg így van.

A dokumentummodell nem más, mint szabályok összessége. Ezek a szabályok meghatározzák, hogy milyen elemeket használhatunk, valamint azt is specifikálhatják, hogy mely elemekre van szükség és melyek opcionálisak, hol lehet az elemeket beszúrni a dokumentumba, és milyen attribútumokat tartalmazhatnak az egyes elemek. Azonosíthatunk például „Könyv” és „Fejezet” nevű elemtípusokat, továbbá világossá tehetjük, hogy míg a fejezet beágyazódhat a könyvbe, ennek fordítottja sohasem lehet igaz.

Konfigurációs sémák

A dokumentummodellt felépítő szabályok elektronikusan kódolhatóak, hogy a számítógépes szoftverek olvasni tudják őket. Amikor a szabályokat adatfájlban tároljuk, a szoftver konfigurációs fájlként tudja őket kezelni. A szoftver először elolvassa a szabályokat, hogy „elsajátítsa” a dokumentummodellt, és így bármilyen problémát felismerhessen a modellen alapuló dokumentumokban.

Több kísérletet tettek, hogy döntsenek arról, mi legyen a dokumentummodellezési szabályok kódolásának sémája, a kísérletek hátterében két fő kérdés áll. Először is, a középpontban a szabályok hatásköre áll, például, hogy a sablonok milyen mértékben korlátozhatják az attribútumok értékeit, és az elemek helye milyen mértékben korlátozható. A másik kérdés maguknak a szabályoknak a szintaxisa.

Az XML-szabvány egy ilyen (az SGML-től örökölt) sémát tartalmaz, amely jelölőelemek segítségével határozza meg a ***Document Type Definition-t*** (***Dokumentum típus definíció, DTD***). Ez azonban az XML-szabvány opcionális funkciója (lásd 5. fejezet). Amikor nem használunk DTD-t, a dokumentum érvényes maradhat, feltéve, ha jól formált.

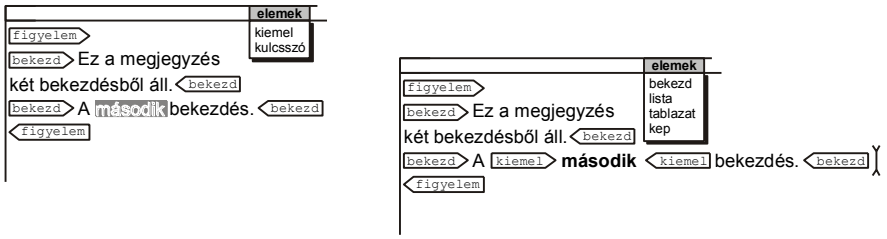
A DTD használata ellen szól az, hogy már léteznek fejlettebb alternatívák. Az XML kibocsátása óta jó néhány alternatíva merült fel, nemrégiben pedig **XML Schema** (14. és 15. fejezet) néven új szabvány született (a rövidség kedvéért a továbbiakban a „DTD" kifejezést használjuk a többi alternatíva helyett; ha a „tartalmazhat" kifejezés szerepel, DTD vagy séma használatára következtethetünk).

Ez az elv alátámasztja az XML-alkalmazás elgondolását, mivel minden XML-alkalmazást (például az XHTML-t, a WML-t, vagy az XMLNews-Story-t) egy vagy több DTD segítségével definiálunk (kiegészítő dokumentációval).

Validáló elemzők

A **validáló elemző** olyan szoftveralkalmazás, amely elolvas egy DTD-t, majd egy olyan dokumentumot is elolvas, amely állítólag megfelel a DTD-ben szereplő szabályoknak, és jelenti a talált eltéréseket. Hibát jelez például, ha a dokumentum tartalmaz Név elemet, és a DTD nem definiálja ezt az elemtípust.

Validálást a dokumentum szerkesztésekor is kezdhetünk. A korábban bemutatott **XML-érzékeny** szerkesztő lekérdezheti a DTD-t, és menüt készíthet az engedélyezett elemtípusokról, hogy a szerző az igényeknek megfelelően választhasson. A menüben megjelenő elemnevek a kurzorpozíció függvényében változhatnak. Az alábbi ábrák közül az elsőn a kijelölt szövegre csak a listában szereplő elemek vonatkoznak. A második ábrán a kurzor a bekezdésen kívül található, így más lista jelenik meg az opciókról:

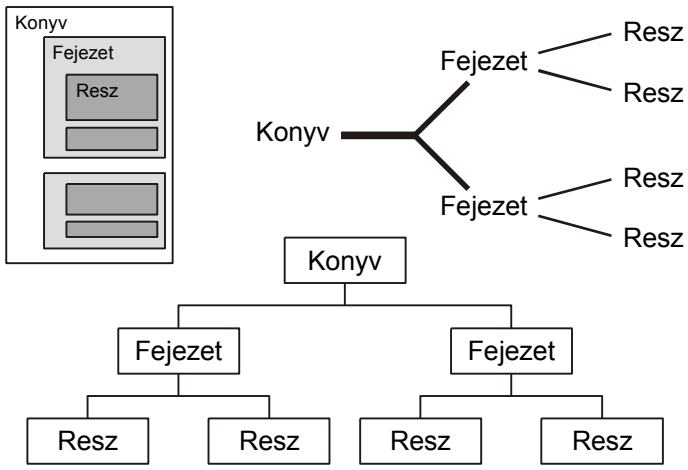


HTML megjegyzés: A DTD nem közvetlenül kapcsolódik a HTML-hez, mivel a megengedett jelölőelemek és a használatukat diktáló szabályok be vannak építve a **HTML-kompatibilis** szerkesztőkbe és Web böngészőkbe. A HTML egyes verzióira azonban léteznek DTD-k, elsősorban e szabványok dokumentálásához, de az SGML-felhasználók érdekében is (akik SGML-eszközökkel HTML-dokumentumokat hozhatnak létre, módosíthatnak, tárolhatnak vagy validálhatnak). A HTML új XHMTL változata XML-alkalmazás, tehát egy DTD definiálja (valójában több DTD).

Az elemhierarchia

A hordozóelemek egyik fő jellemzője, hogy gyakran más elemeket is tartalmazhatnak. A Könyv elemtől például azt várjuk, hogy Fejezet elemeket tartalmazzon, a Fejezet elemektől pedig azt, hogy Rész elemeket foglaljanak magukban. Ezt elem **hierarchiának** nevezzük

A dokumentum elemhierarchiát „doboz a dobozban" módszerrel vagy **fa** ágaiként ábrázolhatjuk. A faábrázolást bármilyen irányban rajzolhatjuk, de a legtermészetesebb nézet talán a balról jobbra vagy a fentről lefelé irány:



Végső soron egy teljes dokumentumot *egyetlen* elemnek kell határolnia. Ez az elem a fa gyökerében található, ezért nem hivatalosan **gyökérelemnek** nevezzük, bár a helyes megnevezése **dokumentumelem**. A fenti példában szereplő Könyv elem tehát dokumentumelem.

Az elemhierarchiák elrendezése

A beágyazott elemeket tehetjük ugyanazon sorba:

```
<könyv><fejezet><rész>...</rész></fejezet></könyv>
```

Az érthetőség kedvéért különböző sorokba is tehetjük őket:

```
<könyv>
<fejezet>
<rész>...</rész>
<rész>...</rész>
</fejezet>
<fejezet>
<rész>...</rész>
<rész>...</rész>
</fejezet>
</könyv>
```

Hogy még átláthatóbb legyen, általánosan bevett gyakorlat (legalábbis példadokumentumokban) a beágyazott elemek behúzása. Az alábbi példában rögtön láthatjuk az összes fejezet elejét és végét:

```
<könyv>
  <fejezet>
    <rész>...</rész>
    <rész>...</rész>
  </fejezet>
```

```
    <fejezet>
      <rész>...</rész>
      <rész>...</rész>
    </fejezet>
  </könyv>
```

A következő példa fiktív XML-alkalmazást mutat be idézetek kezelésére. Ebben a példában jól láthatjuk, hogy mind a közzététel, mind pedig a szerző adatai az idézet része:

```
<idézet>
  <idézetSzövege> A legbiztosabban úgy csinálhatsz majmot
  valakiből, ha idézed őt!</idézetSzövege>
  <forrás>
    <publikáció>Quick Quotations</publikáció>
    <szerző>
      <név>Robert Benchley</név>
      <született>1889</született>
      <meghalt>1945</meghalt>
    </szerző>
  </forrás>
</idézet>
```

Vegyes tartalom

Előfordulhat, hogy az elem szöveget és más elemeket is tartalmaz. Ezt **vegyes tartalomnak** nevezzük, bár bizonyos esetekben a tartalom kizárólag elem vagy kizárólag szöveg lesz. Az alábbi példában a bekezdés szöveget is és Név elemeket is tartalmaz:

```
<bekezdés>A <név>W3C</név> ma kiadta
az <név>XML</név> szabványt. </bekezdés>
```

A sorvégi kódok jelentéssel bírnak a szöveges tartalomban. A fenti példa bemutatja az ideális, vagy biztos módját annak, hogy hogyan formázhatjuk a vegyes tartalmú elem tartalmát. Egy olyan alkalmazásnak, amely ezt a szöveget megjelenítésre vagy nyomtatásra formázza, a „kiadta” szó utáni sorvégi kódot szóköz karakterrel egyenértékűnek kell tekintenie (a 8. fejezetben bővebb információt találhatunk a sorvégek és a szóköz jelentőségéről).

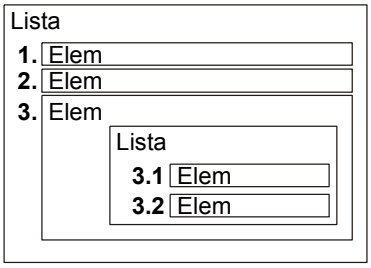
Elemtartalom

Az olyan elemről, amely közvetlenül nem tartalmaz szöveget, viszont más elemeket igen, azt mondjuk, **elemtartalommal** rendelkezik. A Könyv elem például általában nem közvetlenül tartalmaz szöveget, ehelyett tartalmazhat Cím, Bevezetés és Fejezet elemeket.

Hacsak nem DTD-t használunk, nem tudhatjuk biztosan, hogy az elem csak elemtartalommal rendelkezik. Az olvasó ember ésszerű következtetéseket vonhat le az elem nevéből, a szoftver azonban nem ilyen könnyen jut következtetésekre. Az, hogy az elemek között nincs tényleges szöveg, még nem jelenti azt, hogy nem is lehet. Ennek jelentősége számos tényezőtől függ, főleg azzal áll összefüggésben, hogy egy alkalmazás hogyan értelmezheti a sorvégi kódokat, továbbá a fejlett hiperszöveges hivatkozási rendszerekre is hatással lehet.

Rekurzió

Bizonyos hierarchiai struktúrák **rekurzívak** lehetnek. Ez annyit jelent, hogy egy elem közvetlenül vagy közvetve tartalmazhatja ugyanazon típus más példányait. A **beágyazott elem** kifejezést olyan elem leírására is használjuk, amely egy másik, ugyanolyan típusú elembe van beágyazva. Tipikus példa erre, amikor a lista több elemből áll, és az egyik elem további teljes részlistát tartalmaz. Néhány lista és elem-jelölőelem tehát beágyazott:



Ez azonban a végtelen rekurzió lehetőségéhez vezet, ami problémát okozhat a feldolgozó vagy közzétevő szoftver számára. Miután azt a DTD engedélyezte, nem tudjuk korlátozni a rekurzió mértékét:

```
<könyv>
  <fejezet>
    <lista>
      <elem>...</elem>
      <elem>
        <lista>
          <elem>...</elem>
          <elem>
            <lista>
              <elem>...</elem>
              <elem>...</elem>
              <elem>
                <lista>
                  ...
```

Környezetfüggő jelentés

Az olyan könyv, mint például ez, több címet tartalmaz. A könyv címe mellett minden fejezetnek, szakasznak és részszakasznak van címe. Minden használatra más-más elemtípust definiálhatunk, olyan nevekkel, mint a KönyvCím, FejezetCím, RészCím és a AlRészCím.

Ez a megközelítés azonban nehézkes és felesleges. A dokumentumszerzőknek nem kell ennyi elemtípust elsajátítaniuk (bár a DTP-szoftver és a szövegszerkesztő stíluslapjait ismerő olvasó számára nem idegen ez az elvárás).

A hierarchikus és rekurzív struktúrák jelenléte lehetővé teszi, hogy az elemek jelentését legalább részben meghatározza a dokumentumban elfoglalt helyük. A Cím elem tartalmát például másképp dolgozhatjuk fel vagy formázhatjuk, attól függően, hogy az elem közvetlenül egy könyvben, szerzőben, fejezetben, szakaszban, táblázatban vagy ábrán szerepel:

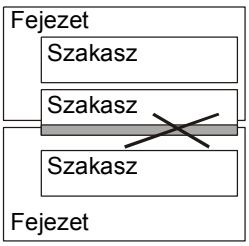
```
<könyv>
  <szerző><cím>Mr</cím>...</szerző>
  <cím>A könyv címe</cím>
  <fejezet>
    <cím>Fejezet címe</cím>
    <rész>
      <cím>Rész címe</cím>
      ...
      <táblázat><cím>Táblázat címe</cím>...</táblázat>
      ...
      <ábra><cím>Ábra címe</cím>...</ábra>
    </rész>
    ...
  </fejezet>
  ...
</könyv>
```

A fejezetcímek listájának célba vételével és kivonatolásával például tartalomjegyzéket hozhatnánk létre.

Szerkezeti megszorítások

A hierarchikus struktúrákat szigorúan be kell tartani. A dokumentum nem jól formált, ha a szerkezet valamilyen okból megtörik. Az elemnek vagy teljesen be kell ágyazódnia a másik elembe, vagy teljes mértékben kívül kell maradnia abból.

Egy szakasz például nem foghat közre két fejezetet:



Akik ismerik a HTML-jelölőelemeket, talán tudják, hogy a Web-böngészőknek nem jelentene gondot az alábbi részlet, amelyben a félkövér és dőlt szedésű szövegtartományok részben fedik egymást:

Az XML-dokumentumokban ez nem megengedett. A fenti szerkezetet tartalmazó dokumentumot nem tartanánk jól formáltnak. Ebben az egyszerű esetben az érvényessé tételhez csak át kell rendeznünk a jelölőelemek záró részét:

A következő példát azonban már nem ilyen egyszerű kijavítani:

Itt két külön elemre kell osztani a dőlten szedett szövegtartományt. Az egyik elemnek a félkövéren szedett elemen belül, a másiknak ezen kívül kell lennie:

```
<b>Félkövér és <i>dőlt</i></b><i> szöveg</i>.
```

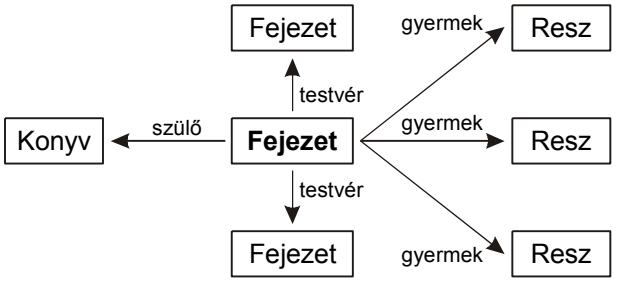
Ezek a megszorítások kényelmetlennek és feleslegesnek tűnhetnek, de szigorú, hierarchikus struktúra felállításához szükség van rájuk. A hierarchiák rendkívül hasznos struktúrák. Minden elemnek egyértelmű környezetfüggő helyet adnak a dokumentumban. Ez az XML-dokumentumrészletek megtalálásánál, vezérlésénél és kezelésénél hasznos (ahogyan a későbbi fejezetekben látni fogjuk).

E megszorítás leküzdésére azonban üres elempárokat használó trükköket dolgoztak ki (lásd 6. fejezet).

Terminológia

Gyakran egy adott elemről kell beszélnünk az XML-dokumentumban, és más, közeli elemekhez kell kötnünk. Az elemek közti kapcsolat leírásakor gyakran a családfa terminológiáját vesszük át (ez az analógia világosan megfelel a faszerkezetnézetnek).

Egy adott Fejezet elem szemszögéből nézve például a szomszédos Fejezet elemek **testvérek,** a Könyv elem a **szülő**, és a benne található szakaszok a Fejezet elem **gyermekei**:

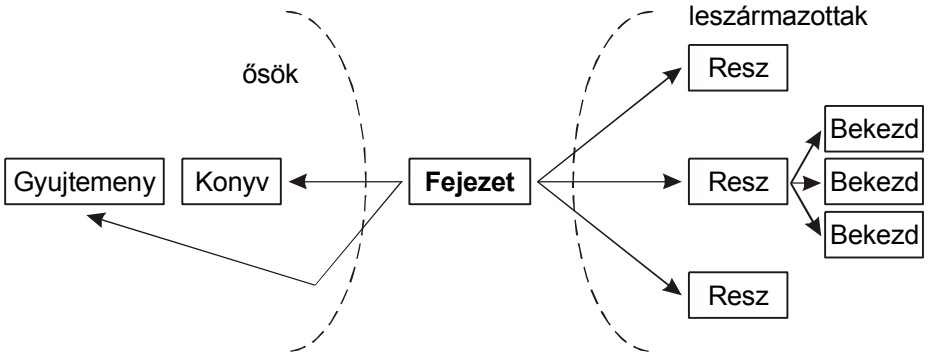


Ezt az elvet illusztrálhatjuk példa XML-dokumentummal, amely megfelelő elemneveket tartalmaz a „cél” nevű elemre vonatkozóan:

```
<szülő>
  <testvér>...</testvér>
  <cél>
    <gyerek>...</gyerek>
```

```
      <gyerek>...</gyerek>
      <gyerek>...</gyerek>
    </cél>
    <testvér>...</testvér>
  </szülő>
```

Az analógiát tovább vezetve, a Fejezet elem által közvetlenül vagy közvetve határolt összes elem annak az elemnek a **leszármazottja – descendant** (a gyermekeit is beleértve), a Könyv elemet pedig annak **elődjeként – ancestor** (és egyben szülőjeként – parent) írhatjuk le. Ha a Könyv elem ugyanazon XML-dokumentumban egy könyvgyűjtemény része lenne, a Gyűjtemény elem szintén elől lenne:



Az alábbi, megfelelő elemneveket tartalmazó példa is ezt az elvet szemlélteti:

```
<felmenő>
  <felmenő>
    <cél>
      <leszármazott>...</leszármazott>
      <leszármazott>
        <leszármazott>...</leszármazott>
        <leszármazott>...</leszármazott>
        <leszármazott>...</leszármazott>
      </leszármazott>
      <leszármazott>...</leszármazott>
    </cél>
  </felmenő>
</felmenő>
```

A családfa elvén alapuló terminológia azonban korlátozott. Egyrészt a többes számú „szülők” szónak nincs értelme, mivel az XML-elemeknek csak egy szülőjük lehet. Ugyanígy, az olyan elemet, amelynek nincsenek gyermek elemei, nem „gyermektelen”, hanem „**levél**” elemnek nevezzük (éppúgy, ahogy az olyan elemet, amelynek nincs szülője, „gyökér” elemnek hívjuk).

Attribútumok

Egy elem a nevén felül további információt is nyújthat a tartalmára vonatkozóan. Egy adott Bekezdés elem tartalmának célközönségét például szabályozhatja biztonsági szint érték, és minden bekezdést egy adott szerzővel társíthatunk. Ezt az „információ az információról” jelenséget **metaadatnak** nevezzük, és egy **attribútumban** tároljuk. Az attribútum névvel és értékkel is rendelkezik. Az alábbi példában a bekezdést titkosnak jelöljük a nyitó jelölőelembe beágyazott attribútum segítségével:

```
<bekezdés biztonságiSzint=”titkos” >
  ...
</bekezdés>
```

Az attribútumokat üres jelölőelembe is beágyazhatjuk. Egy kép helyőrzőjének például attribútumra lenne szüksége a beszúrandó kép azonosításához:

```
A hajó ilyen <image file=”hajo.gif” />.
```

Egyetlen elem több attribútumot is tárolhat. A következő bekezdés rejtett, továbbá azonosított szerzővel rendelkezik:

```
<bekezdés biztonságiSzint=”titkos” szerző=”J. Smith”>
  ...
</bekezdés>
```

Az attribútumok felépítése

A jelölőelem kezdő részében vagy az üres jelölőelemben minden attribútumnak tartalmaznia kell az attribútum nevét, értékét, és a fent bemutatott további jelölő karaktereket. Az egyenlőségjel („=”) elválasztja a nevet az idézőjelben szereplő értéktől.

HTML megjegyzés: A legtöbb HTML felhasználó ismeri a ”<hr noshade>” vagy a ”<ol compact>” rövid formát. Az attribútum név nincs jelen, az attribútum csak egyetlen megengedett (például a „compact”) értékkel rendelkezik. Ha az érték nincs jelen, az elemnek burkolt értéke van, például „shaded” vagy „not compact”, így a jelenléte egyszerű kapcsoló utasításként működik. Mivel az XML megköveteli az attribútum név, az egyenlőségjel és a határoló idézőjelek meglétét, ez a technika nem használható.

Legalább egy szóköz kell, hogy álljon az elemnév és az első attribútum között, valamint az attribútumok között. Opcionálisan szóközök is állhatnak az egyenlőségjel mellett:

```
<téma kulcsszavak = ”XML SGML” id   =   ”x123”  >
```

Attribútumnevek

Az attribútumnév érzékeny a kis- és nagybetűkre. Éppúgy, mint az elemneveknél, figyelnünk kell a név pontos írására, hiszen a „Típus” eltér a „típus” vagy a „TÍPUS” attribútumtól. Az attribútumnevekre ugyanazok a megszorítások érvényesek, mint az elemnevekre.

Ahogyan az elem pontos jelentése részben a dokumentumstruktúrában elfoglalt helyétől is függhet, az attribútum pontos jelentése az azt tartalmazó elemtől is függ. Egy Key nevet viselő attribútumnak például más jelentése lenne a Song nevű elemben, és más a Password nevű elemben:

```
<Password Key="x123yz" ... />
<Song Key="C" ... />
```

Néhány attribútumnév az XML-szabvány részére van fenntartva, vagy a jövőben lesz majd lefoglalva kiegészítő szabványok számára. Ezek az attribútumnevek minden esetben „xml:"-tal kezdődnek. A szabvány például az „xml:lang" nevet olyan attribútum részére tartja fenn, amely az elem tartalmában használt emberi nyelvről tartalmaz információt, és ezt minden (érdekelt) XML-kompatibilis alkalmazás elismeri (előfordulhat, hogy a felhasználó csak az adott nyelvnek megfelelő elemek tartalmát kívánja látni, vagy a helyesírás-ellenőrző alkalmazás a nyelv változásakor válthat a szótárak közt).

Attribútumértékek

Az attribútum értékét idézőjelek határolják, mivel szóközök fordulhatnak elő benne, és máskülönben lehetetlen volna felismerni egy érték végét, ha további attribútumok állnak utána. Vizsgáljuk meg a következő kezdő jelölőelemet:

```
<téma kulcsszavak="XML SGML" id="x123">
```

A Kulcsszavak attribútum jelenleg „XML SGML" értékkel rendelkezik, az Id attribútum pedig „x123" értékkel. Idézőjelek nélkül a feltételezés az lenne, hogy a Kulcsszavak attribútum értéke „XML SGML id=x123", és ha a szóközt értéket lezáró karakterként értelmeznénk, a Kulcsszavak attribútum feltételezett értéke csupán „XML" lenne.

Bár általában idézőjelet használunk, használhatunk helyette aposztrófot is. Akkor kell például aposztrófot használnunk, ha idézőjelet tartalmazó értéket szeretnénk a szöveg részeként behatárolni:

```
<csavar átmérő="2"">
```

Ugyanígy az sem probléma, ha aposztróf szerepel az idézőjellel elhatárolt attribútumértékekben:

```
<oszlop átmérő="2"">
```

Ha az értékben mindkét idézőjeltípust használjuk, vezérlőkódot kell alkalmaznunk (idézőjel esetén „**"**", aposztróf esetén „**'**"). Az, hogy melyiket kell használnunk, a határolóknál alkalmazott karaktertől függ:

```
<oszlop átmérő="2"5&quot;">
<oszlop átmérő="2&apos;5"">
```

HTML- és SGML-megjegyzés: Fontos megjegyeznünk, hogy a körülfogó idézőjeleket kötelező kitenni, és az attribútum névnek mindig meg kell jelennie.

Az attribútum értékében bármely tabulátort, kocsivisszát, vagy új sort a szóközkarakterrel megegyezőnek tekintünk, és szóközre fordítunk le (néhány jellemzőtípus esetén a térköz további beállításait is elvégezzük DTD alkalmazásakor). A CR (kocsivissza), majd az LF (új sor) kombinációját egyetlen szóközre fordítjuk le. Az alábbi példák tehát egyenértékűek:

```
név="John Smith"
```

```
név="John
Smith"
```

Az XML felhasználási területei

Az XML-t sokszor és sokféleképpen használhatjuk. Itt nem tudjuk mélységeiben leírni a rengeteg XML-alkalmazást, de nézzünk néhány példát az általános használatra az alkalmazási területek szélességének szemléltetésre.

Az XML korábbi, a kiadói iparra koncentráló technikákból született, és folytatja e terület kiszolgálását. Használhatjuk részben strukturált dokumentumok, például referenciamunkák, képzési útmutatók, kézikönyvek, katalógusok, tudományos folyóiratok és jelentések jelölésére. Sok más szűk piaccal rendelkező termék közt az XML-t szabadalmak, dolgozatok, éves beszámolók és kutatómunkák jelöléséhez használhatjuk.

E háttér ellenére az XML-t teljesen más problémára kínált megoldásként dobták piacra. A legizgalmasabb új alkalmazás – az, amely a legnagyobb érdeklődést váltotta ki a sajtóban – megoldást jelent összetett adatok szoftveralkalmazások közti átvitelére, elsősorban a weben keresztül. Az XML-adat önleíró jellege alapvető fontosságú ehhez az alkalmazáshoz.

Az XML-t emellett ideális adatformátumnak tekintik a konfigurációs fájlok számára. Kihasználhatjuk azt a tényt, hogy az XML-érzékeny szerkesztőeszközök irányítani és segíteni is tudják azokat, akiknek adott célból szoftvert kell konfigurálniuk.

Akkor is alkalmazhatunk XML-t, ha a szöveg nem, vagy csak kevés szerepet játszik az alkalmazásban. XML-t használunk grafikai információ ábrázolására is. Megszületett az **SVG** (*Skálázható vektorgrafika, Scalable Vector Graphics*) nevű szabvány, és széles körű támogatásra tett szert. A többi „vektor" formátumhoz hasonlóan közvetlenül csak vonalakból, görbékből és szövegstringekből felépíthető képekre alkalmas. A következő elem megmondja az SVG-alkalmazásnak, hogy rajzoljon egy két hüvelyk széles és egy hüvelyk magas piros téglalapot:

```
<rect style="fill:red" width="2in" height="1in" />
```

Az XML-t még multimédia adatformátumként is használják. Segítségével multimédiás prezentációkat készíthetünk, lehetővé téve olyan dokumentumok létrehozását, amelyek utasítják a **SMIL** (*Szinkronizált multimédia integrációs nyelv, Synchronized Multimedia Integration Language*) lejátszót a bemutató lejátszásának módjáról.