XML Túra (1. rész)
Alapok

Egy újabb rövidítés, mely talán még sokak számára nem világos, hogy mire is hivatott. A cikksorozat első részében lerántjuk a leplet erről az eszközről, megnézzük honnan jött és miért, valamint az alkalmazásaira is fordítunk majd egy kis figyelmet. Lássuk hát miről is szól az Extensible Markup Language, azaz az XML.

Mit takar a név?

Magyarul nevezhetjük ***kiterjeszthető leíró nyelvnek***. Ez az eszköz a ***W3C (World Wide Web Consortium)*** keze munkája, egy általános célú jelölő nyelv, melynek segítségével sajátos, speciális célú leíró nyelvek konstruálhatók. Az ***SGML*** nyelvek közé sorolhatjuk. Mivel igen alkalmas különböző adattípusok leírására, ebből is adódik, hogy elsődleges célja strukturált adatok megosztása az interneten keresztül (***RSS, SVG***).

A történet

A fejlesztés alapjai egy ember nevéhez fűződnek: ***Tim Bray***. Mielőtt a ***W3C***-hoz került volna, egy az ***IBM***, ***Oxford University Press*** valamint a ***University of Waterloo*** által támogatott projekten dolgozott. A projekt egy internetes szótár kialakítását tűzte ki célul. Nagy hangsúlyt fektettek azon módszerekre, melyeket az adatok indexelésére és tárolására alkalmaztak. Ebben a struktúrában beágyazott címkék adták meg az adat fajtáját, így született meg az ***XML*** elődje. Később ***Tim Bray*** lett a ***W3C XML*** specifikációjának szerkesztője.

Miért is volt szükség az XML-re?

Az internet fejlődésével egyre több, bonyolultabb struktúrájú adatot kívántak közzétenni. A ***HTML***-ben viszont egyre nehézkesebbé vált az ilyen bonyolult struktúrák ábrázolása. Szükségesség vált egy szabadabb nyelv megalkotása. Míg a ***HTML*** csak egy megadott elemhalmazt használhat fel, addig az ***XML***-ben magunk alkothatunk elemeket. Nem tartalmaz annyi megkötést, mint az ***SGML*** nyelvek általában, ezért könnyen megtanulható, programozható és olvasható. Az ***XML*** így lett a ***HTML*** kisegítője, együtt jelentős mértékben növelték a weblapok lehetőségeit.

Célkitűzések

Az ***XML*** sajátos filozófiával rendelkezik. A fejlesztők több dolgot vettek figyelembe munkájuk során. Mindenek előtt a könnyű használatot, hogy gyorsan elterjedjen a nyelv, és ezzel egyetemben a széleskörű felhasználhatóságot is szorgalmazták. Az opcionális elemeket próbálták minimálisra csökkenteni, mivel ezekből eredhet a legtöbb kompatibilitási probléma. A forráskód kinézeténél ügyeltek a világos és olvasható struktúrára. A nyelvet minél előbb el kellett készíteni, nehogy esetleg több (kisebb cégek próbálkozásaiból származó) nyelv terjedjen el ezzel a céllal. Nagy jelentőséget tulajdonítottak az olvashatóságnak és ezért nem is törekedtek nagyon a tömörségre.
A tervük – úgy tűnik – bevált. 1996 óta az ***XML*** futótűzként terjedt el a webfejlesztők, programozók körében és ma már igen sok helyen felüti a fejét. Nem egy mobiltelefon használja bizonyos adatok tárolására. Faszerkezetű struktúrája alkalmas bármilyen (logikusan csoportosított) adat tárolására, mint például táblázat, címjegyzék, üzleti tranzakciók vagy műszaki rajzok. Az adatok stíluslapok (***CSS***) alkalmazásaival könnyen megjeleníthetővé válnak. Egyedi dokumentumtípus létrehozásával kialakítható akár saját jelölő nyelv is: ***WML*** (Wap oldalak leírására), ***VML*** illetve ***SVG*** (vektorgrafika tárolására) valamit ***OFX*** (pénzügyi információcserére). A sort még jócskán lehetne folytatni. Az ***OpenOffice*** és az ***AbiWord*** is natív fájlformátumként az ***XML***-t alkalmazza. Népszerűsége mellett szól az is, hogy támogatja a ***UNICODE***-ot és ezáltal lehetővé teszi bármely információ bármely emberi nyelven történő közlését. Képes a legtöbb számítástudományi adatstruktúra ábrázolására, mint például ***fa***, ***lista***, ***rekord***. Könnyű olvashatóságát ***öndokumentáló*** tulajdonsága is javítja, struktúra és mező neveket ír le a hozzájuk tartozó esetleges értékekkel együtt. Egyszerű szöveges formátumként valósul meg, így következik, hogy ***platform független*** is tehát nem nagyon reagál a technológiai változásokra sem. Ez a tulajdonsága is igen kecsegtető, hiszen ki is szeretné, hogy a ma elkészített dokumentumai holnap már olvashatatlanok legyenek valami kompatibilitási probléma folytán.

Valamit valamiért

Azonban, mint minden más dolog esetén, a kényelemnek és kezelhetőségnek itt is akadnak árnyoldalai.

1. Lista Egy gyökérelem

```
<dolgozók>
  <dolgozó>Kovács
Elek</dolgozó>
  <dolgozó>Szabó Elemér
</dolgozó>
</dolgozók>
```

2. Lista Két gyökérelem már nem megengedett

```
<dolgozók>
  <dolgozó>Kovács
  ➥Elek</dolgozó>
  <dolgozó>Szabó Elemér
  ➥</dolgozó>
</dolgozók>
<dolgozók>
  <dolgozó>Sándor
  ➥Tamás</dolgozó>
</dolgozók>
```

3. Lista Attribútum nevek alkalmazása

```
...
<dolgozó telefon="555-634"
➥kor="45">Kovács
➥Elek</dolgozó>
...
```

Az olvashatóság első nagy ára a tárolási költség megnövekedése. Főleg fontos lehet ***PDA***-k és mobilok esetén, mely eszközök előszeretettel alkalmazzák az ***XML***-t videók és képek leírására. Tömörítéssel ez a probléma még orvosolható, mint ahogyan például azt az ***OpenOffice*** is teszi. Nincs lehetőség a dokumentum egyes részeinek közvetlen elérésére vagy frissítésére sem. Feldolgozásában problémát jelenthet, hogy nincs valami gazdag típus kezelése, például nem oldható meg XML-ben hogy a ***Pi*** közelítő értékét lebegőpontos számként tároljuk karakterekből álló karakterlánc helyett.

Helyes XML dokumentum

Láttuk miről is szól az ***XML***, és remélhetőleg több olvasónak is viszket már a tenyere az alkalmazására.
Az ***XML nem*** egy ***programnyelv***. Használatához nem kell nagy programozási ismeretekkel rendelkezni. Könnyen és gyorsan megtanulható. Túránkat folytassuk azzal, hogy mikor beszélünk helyes ***XML*** dokumentumról. ***Helyes XML dokumentumról van szó, ha az helyesen formázott és érvényes.***

Helyesen formázott dokumentumok

Egy XML dokumentum csak egy gyökérelemet tartalmazhat. Például tekintsük egy fiktív munkahely dolgozóinak listáját feldolgozó dokumentumot (*1. Lista*).
Ez a példa egy gyökérelemet tartalmaz, amelynek neve `dolgozók`.
Ha most egy új `dolgozók` elemet szúrnánk be a *2. Lista* szerint, akkor a dokumentumunk már nem lenne helyes.
A nem üres elemek mind ***nyitó*** és ***záró*** címkével kell hogy rendelkezzenek.
Az *1. Listában* láthatjuk hogy a `dolgozó` elemhez tartozó nyitó tag `<dolgozó>` és a záró tag `</dolgozó>`.

Az üres elemek megjelölhetők ***önlezáró címkével***: `<üres/>`, ami persze megegyezik a következővel: `<üres></üres>`.
Minden attribútum értékének idézőjelek között kell szerepelnie, hogy szimpla (') vagy dupla (") az tetszőleges. Ami viszont lényeges, hogy szimpla csak szimplát, dupla idézőjel pedig csak duplát zárhat. Attribútumok alkalmazásánál érdemes beszédes neveket alkalmazni *3. Listát*.
A tag-ek egymásba ágyazhatók, de nem fedhetik át egymást. Tehát ha egy tag egy másik elemen belül kezdődik, akkor azon belül is kell lezáródnia.
A nevek kisbetű-nagybetű érzékenyek tehát figyeljünk oda, hogy például a: `<Példa>...</Példa>` helyes, amíg a `<Példa>...</példa>` helytelen használat.

Érvényes dokumentumok

Érvényes ***XML*** dokumentumok a felhasználó által definiált tartalmi szabályoknak megfelelő dokumentumok. Ilyen szabályrendszert alkothatunk akár ***DTD (Documentum Type Definition)*** segítségével is, melyet a dokumentumhoz csatolhatunk. Később még lesz szó a ***DTD***-ről és annak alkalmazásáról. A szabályok megszabják a helyes adatértékeket és azok helyeit. Például érvényességi szempontból hibás egy ***XML*** dokumentum, ha egy egész számként értelmezendő elem egy szöveges értéket tartalmaz, vagy éppen üres. Vannak olyan karakterek melyek nincsenek hatással az információ értelmezésére, ezeket nevezzük ***jelentés nélküli karaktereknek***. Ilyen karakter például ***tabulátor, szóköz, sortörés***.

1. ábra A dolgozok.xml megjelenítése Firefoxban

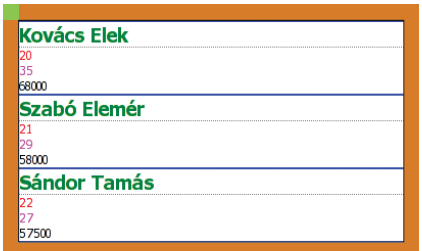
Gyakorlat

Ennyit az elméletről. Itt az ideje, hogy alkossunk is valamit. Egy ***XML*** dokumentumot bármilyen szövegszerkesztővel létre tudunk hozni. A lényeg hogy szöveges állományként legyen elmentve „***XML***" kiterjesztéssel. Gyúrjuk tovább az először említett példát. Pár dolog feltűnhet, amiről még nem tettünk említést, de a *4. Listában* már szerepel. Az első sor tartalmazza a dokumentum ***fejlécét***. Itt adhatjuk meg, hogy mely ***XML*** szabvány szerint dolgozunk (***version***) valamint, hogy milyen karakterkódolást alkalmazunk (***encoding***). A második sor egy egyszerű megjegyzést tartalmaz. Csak hogy ez is egyértelmű legyen: minden megjegyzést `<!-` és `-->` jelek közé illesztünk. A frissen alkotott ***XML*** forrásunk máris megjeleníthető valamely böngésző segítségével. Nagyon újat nem fog mutatni a megjelenítés,

4. Lista Első XML dokumentumunk: dolgozok.xml

```
<?xml version="1.0" encoding=
➥"ISO-8859-2" ?>
<!- XML Túra 1. rész -
➥Dolgozók listája ->

  <dolgozók>
    <dolgozó>
      <név>Kovács Elek</név>
      <osztály>20</osztály>
      <kor>35</kor>
      <fizetés>68000</fizetés>
    </dolgozó>
    <dolgozó>
      <név>Szabó Elemér</név>
      <osztály>21</osztály>
      <kor>29</kor>
      <fizetés>58000</fizetés>
    </dolgozó>
    <dolgozó>
      <név>Sándor Tamás</név>
      <osztály>22</név>
      <kor>27</kor>
      <fizetés>57500</fizetés>
    </dolgozó>
  </dolgozók>
```



2. ábra CSS alkalmazásával Firefoxban

5. Lista Az adatok arculata CSS segítségével dolgozok.css

```
dolgozó
{
     display:    block;
     border:   1px solid blue;
     font-family: tahoma;
     width:  400px;
}

név
{
     display:    block;
     font-size:    16px;
     font-weight: bold;
     color:  green;
     border-bottom: 1px dotted
     ➥gray;
}

osztály, kor, fizetés
{
     display:    block;
     font-size:    10px;
}

osztály
{
     color:  red;
}

kor
{
     color:  magenta;
}
```

hiszen még nem definiáltunk semmilyen ***stílust***, mely szerint megformázhatná számunkra a böngésző az adatokat.

Stílus, forma, CSS

Amint már említésre került, az *XML* lehetőséget biztosít, arra hogy dokumentumunkhoz tetszés szerinti ***stíluslapot*** – akár többet is – hozzáfűzzünk. Ezek nélkül a böngésző csak egy ***fastruktúrát*** fog mutatni az állományból, melynek ágait általában be lehet zárni, vagy ki lehet fejteni tetszés szerint egy kattintással (*1. ábra*).
Az képkivágás tetején látható, amint a *Firefox* figyelmeztet is bennünket, hogy a megjelenített dokumentum nem tartalmaz stílusinformációt. Tüntessük hát el onnan azt a csúnya figyelmeztetést és vegyük át az uralmat dokumentumunk kinézete felett. Először létrehozunk egy *CSS* állományt mely tartalmazza a dokumentum és egyes eleminek a megjelenítési szabályait. A *CSS*-ről feltételezem nem először hall az olvasó. Ez a szabvány szintén a *W3C* ajánlása melyet főleg *HTML* dokumentumok formázására alkalmaznak. Egy sima szöveges állomány, melynek befűzését a dokumentumunk elején fogjuk megadni a böngésző számára.

Ha meg szeretnék szabni egy elem stílusát, akkor *CSS*-ben a nevével hivatkozunk rá, mint például ***osztály***. Aztán { és } jelek közé tesszük a formázás szabályait. Mivel a nem célom a *CSS* részletes magyarázata, ezért csak lényegretörően írok az egyes elemekről. A `display: block;` szabály adja meg az értelmezőnek, hogy az ezt követő elemek már másik sorba fognak kerülni. A `border: 1px solid blue;` adja meg hogy az elem körül egy 1 pixel vastag, folyamatos, kék vonal fog húzódni. A `font-size: 10px;` elég egyértelműen az elem betűtípusának méretét szabályozza 10 képpontra, míg a `color: red;` sor az elem betűjének színét pirosra állítja. Betűink vastagságát a `font-weight` tulajdonsággal tudjuk megadni, az elem szélessége pedig a `width` kulcsszó segítségével befolyásolható.
Már csak be kell kapcsolnunk a létrehozott stíluslapot a dokumentumunkba. Ez csak annyiból áll, hogy a fejléc alá beszúrunk egy hivatkozást:

```
<?xml version="1.0"
➥encoding="ISO-8859-2" ?>
<?xml-stylesheet
➥type="text/css"
➥href="dolgozok.css" ?>
...
```

Innen fogja tudni a megjelenítő (böngésző), hogy az adott dokumentumon mely stíluslap szerint jelenítse meg.

Ami következik

Természetesen az *XML*-t nem arra találták ki, hogy szépen formázott adatokat jeleníthessünk meg a segítségével. Ennél komolyabb célokat tűztek ki a számára. A következő részben lesz szó az említett *DTD*-ről, *DSO (Data Source Objectről)* azaz arról, hogy hogyan is hivatkozhatunk például *HTML*-ből *XML*-ben tárolt adatokra.

Radics Péter
(peter.radics@gmail.com)
Az ELTE-n tanulok programtervező matematikus szakon. Hobbim a kosárlabda, autóvezetés, web-design, programozás. Főleg webes alkalmazások fejlesztése érdekel. 4 éve megrögzött Linux felhasználó vagyok.