Az XML alapnyelv

dr. Paller Gábor

Az XML gyökerei

- 1969: egy IBM kutatási projekt kifejleszti a GML-t (Generalized Markup Language). A GML már rendelkezik azzal a képességgel, hogy dokumentumformátumok metanyelve legyen (tehát le lehessen benne írni dokumentumformátumokat).
- 1980: első ISO szabvány az SGML-ről (Standard Generalized Markup Language). 1983-ra az SGML ipari szabvány lett. Korlátozott elterjedtség jelentős bonyolultsága miatt.
- 1990-1992: Tim Berners-Lee az SGML-t használva hozza létre a legismertebb SGML-alapú dokumentumformátumot, a HTML-t.
- 1997-1999: A World Wide Web Consortium (W3C) a webre alkalmazza az SGML-t, amelyet túl bonyolultnak találnak közvetlen felhasználásra. Az eredmény az Extensible Markup Language (XML).
- 1998-2000: a Microsoft az XML-re építi a számítógépes rendszerek korlátlan összekapcsolhatóságát ígérő szabványát, amit Web Services néven népszerűsítenek.
- 1999-: a W3C az XML köré eszközök sorát építi fel az XML köré. Ezekkel bővebben megismerkedünk majd.

XML alapnyelv

- XML 1.0 specifikáció (4. kiadás): http://www.w3.org/TR/2006/REC-xml-20060816/
- Tetszőleges hierarchikus adatszerkezet leírására alkalmas
- Tartozik hozzá egy egyszerű adatformátum-leíró nyelv (DTD)

Szöveg és karakteradat

- Az XML dokumentum *szövegből (text)* áll.
- A szöveg szigorúan textuális.
  ```
  Char   ::=   #x9 | #xA | #xD | [#x20-#xD7FF] | [#xE000-#xFFFD]
  | [#x10000-#x10FFFF]
  ```
- A szöveg vagy *karakteradatot (character data),* vagy *markup-ot* tartalmaz.
- A karakteradat nem tartalmazhat & és < karaktert, valamint kompatibilitási okokból > karaktert sem. Ezeket minden esetben vezérlőszekvenciákká kell alakítani (& < >).
- Ezen felül az ' (') és a " (') szekvenciákat definiálták.
- Bármely Unicode karakter használható numerikus karakterformátummal: &#xxx; (xxx a karakter hexadecimális kódja). Ez csak a beviteli eszköz korlátozásainak feloldására való, a karakternek továbbra is meg kell felelnie a Char megkötéseinek.
- A szövegben lehetnek szakaszok, amelyekre nem vonatkoznak a karakteradatok megszorításai. Ezek neve CDATA szekció és ezek belsejét az XML elemző nem dolgozza fel. Példa:
  ```
  <![CDATA[<greeting>Hello, world!</greeting>]]>
  ```
- A CDATA szekciók tartalma továbbra is Char!
- A szövegben levő "fehér karakterek" sorsát az alkalmazás dönti el, az XML elemző csak a markup-nál (ld. később) távolítja el azokat.

Markup

- A markup a dokumentum szerkezetét írja le és hatással lehet a feldolgozására.
- A legfontosabb markup elemek a *tag*-ek. A egy nyitó-záró tag egy *elem-hez (element)* tartozó adatokat határolja.
  - Nyitó tag: <sometag>
  - Záró tag: </sometag>
  - Üres elem tag: <sometag/>
- Az XML metanyelv. A tag-eket és azok használatát mi definiáljuk (a definíció módjával még fájdalmas részletességgel foglalkozunk).
- A tag-ek szigorúan hierarchikus felépítésűek, egymásba ágyazás esetén a beágyazott tag-et előbb le kell zárni, mint a külsőt.

```
<sometag>
...
  <othertag> ... </othertag>
...
</sometag>
```

Attribútumok

- A tag-hez tartozó adatszerkezet elemei:
  - A tag *tartalma (content)*: a nyitó és záró tag között levő szöveg (beleértve a beágyazott markup-ot is).
  - A tag *attribútumai*, ha vannak.
- Példa:
  `<etag parm1="hello" parm2="hallo"> ... </etag>`
  Két attribútum (parm1, parm2)

Prológ

- Az XML dokumentumoknak a következő prológgal kell kezdődnie:
```
<?xml version="1.0"?>
```
- Opcionálisan a prológ definiálhatja a szöveg karakterkódolását (UTF-8 az alapértelmezett):
```
<?xml version="1.0" encoding="ISO-8859-2" ?>
```
- A prológban még dokumentumtípus-definíció is lehet, amely megadja a dokumentum formátumleíróját (részleteket később)
  Példa:
```
<?xml version="1.0"?>
<!DOCTYPE greeting SYSTEM "hello.dtd">
```

Kommentek

- Az XML dokumentumba kommentek helyezhetők.
- A kommentet a <!– karaktersorozat nyitja és a --> karaktersorozat zárja.
- Komment nem tartalmazhat -- karaktersorozatot és csak Char-ból állhat. Viszont lehetnek benne XML elemek, például tag-ek.
- Az XML elemző figyelmen kívül hagyja a komment tartalmát, függetlenül a komment tartalmától.
- Példa:

```
<!-- declarations for <head> & <body> -->
```

Feldolgozásvezérlés

- Az XML dokumentumok tartalmazhatnak utasításokat az őket feldolgozó alkalmazásoknak.
- Maga az XML alapnyelv ilyeneket nem definiál, csak a prológ formátuma hasonló.
- A feldolgozásvezérlő utasítás (processing instruction) formája a következő:
  `<?instruction parameters ... ?>`
  Az "instruction"-t célnak (target) is nevezik.
- Példa: (az XML stylesheet-hez kapcsolódó kiegészítő specifikációból)
  `<?xml-stylesheet href="mystyle.css" type="text/css"?>`

Document Type Definition

- Ismétlés: az XML metanyelv. Arra való, hogy tetszőleges adatstruktúrákat tartalmazó dokumentumformátumokat ill. ezeknek a formátumoknak megfelelő dokumentumokat hozzunk létre.
- A dokumentumformátum definiálása:
  - Emberi nyelven írt specifikációval, amelyet aztán az XML dokumentum feldolgozása során a célra írt programmal kell ellenőrizni.
  - Gép által feldolgozható formátumleíró nyelvvel, amely lehetővé teszi az XML feldolgozónak, hogy maga ellenőrizze a dokumentum formátumát.
- Az XML alapnyelv az SGML formátumleírójának egyszerűsített változatát nyújtja (Document Type Definition, DTD).
- Ezen felül az XML-hez a lényegesen bonyolultabb adatszerkezeteket leírni képes XML Schema formátumleíró nyelvet is létrehozták (később jön a tantárgy folyamán).

Helyesen formált és érvényes dokumentumok

- Egy XML dokumentum lehet
  - Helyesen formált (well-formed), ha megfelel az XML általános nyelvi szabályainak (pl. prológ, a szöveg karakterei, speciális karakterek, tag-ek használata, stb.)
  - Érvényes (valid), ha ezen felül megfelel egy DTD-nek is.
- Az érvényesség megállapításához a DTD-t a dokumentumhoz kell kapcsolni. Lehetőségek:
  - DTD beágyazása a dokumentumba (a DOCTYPE első paramétere (greeting) a legfelső szintű tag-et határozza meg).

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE greeting [
  <!ELEMENT greeting (#PCDATA)>
]>
<greeting>Hello, world!</greeting>
```

  - DTD hivatkozása külső fájlból (URI hivatkozás lehetséges)

```
<!DOCTYPE greeting SYSTEM "hello.dtd">
```

  - Ezek keverése is lehetséges:

```
<!DOCTYPE rootElement SYSTEM "URIreference"[ ...
declarations ...]>
```

  - DTD hivatkozása rendszerfüggetlen azonosítóval (ha a PUBLIC második paramétereként megadott DTD fájl nem elérhető, a PUBLIC első paramétere alapján keres egy lokális DTD fájlt):

```
<!DOCTYPE chapter PUBLIC "-//OASIS//DTD DocBook XML//EN"
"../dtds/chapter.dtd">
```

DTD: entitás deklarációk

- Az entitás (entity) egy makró-szerű konstrukció az XML-ben
- Lehet elemzett (parsed) vagy nem elemzett (unparsed). Az elemzett entitás kifejtésekor belekerül a szövegbe, tehát pl. a benne levő tag-eket az XML elemző feldolgozza.
- Entitások fajtái
  - Belső entitás (internal entity) - ha az entitás a DTD fájlban van deklarálva.
  - Külső entitás (external entity) - ha az entitás értéke egy másik, a külső entitás deklarációja által meghatározott fájlban tárolódik. Ez esetben a fájl tartalma az entitás hivatkozásakor beszúródik a szövegbe.
  - Paraméter entitás (parameter entity) - csak DTD-ben használható és ott azonnal kifejtődik.
- Példák:
  - Belső entitás:
    ```
    <!ENTITY Pub-Status "This is a pre-release of the specification.">
    ```
  - Külső entitás:
    ```
    <!ENTITY open-hatch SYSTEM "http://www.textuality.com/boilerplate/OpenHatch.xml">
    ```
  - Paraméter entitás:
    ```
    <!ENTITY % personcontent "quote">
    ```
  - Entitás hivatkozása szövegben (ill. DTD-ben, ha paraméter entitás):
    ```
    &Pub-Status;
    ```

DTD: külső entitások (2)

- Külső entitás: elemzett vagy nem elemzett
  - Elemzett: XML-nek kell lennie és az entitás kifejtésekor behelyettesítődik a szöveg adott helyén.
  - Nem elemzett: bármilyen típusú lehet (bináris is), de csak XML tag-ek attribútumaiban lehet felhasználni őket hivatkozásként (ha az atrribútum entity típusú)
- Példák:
  - Külső entitás XML tartalommal (elemzett)
    ```
    <!ENTITY chap1 SYSTEM "chap1.xml">
    ```
  - Külső entitás bináris tartalommal (nem elemzett):
    ```
    <!NOTATION gif SYSTEM "CompuServe Graphics
    Interchange Format 87a">
    <!ENTITY hatch-pic SYSTEM "../grafix/OpenHatch.gif"
    NDATA gif >
    ```

Elemek definíciója

- Az elemeket a bennük található tartalommal definiáljuk.
- Ez a tartalom további elemek és a rájuk vonatkozó megkötések lehetnek
  - Megköthető a benn foglalt elemek listája (szekvencia) vagy választási lehetőség (choice). Példák:
    - A "spec" nevű elem belsejében a "front", "body" és "back" elemeknek kell lenniük, ebben a sorrendben.
      ```
      <!ELEMENT spec (front, body, back)>
      ```
    - A "div1" nevű elem belsejében a "p", "list" vagy "note" elem található (egy és csak egy közülük)
      ```
      <!ELEMENT div1 (p | list | note)>
      ```

Elemek definíciója (2)

- Megkötések tehetők az elemek előfordulási számára. Példák:
  - A "spec" nevű elem belsejében egy darab "front" és egy darab "body" elemnek kell lennie. Ezen felül egy "back" elem is lehetséges (0 vagy 1 darab)
    ```
    <!ELEMENT spec (front, body, back?)>
    ```
  - A "div1" elem belsejében a "head" elem után tetszőleges számú "div2" elem lehet (0 vagy több)
    ```
    <!ELEMENT div1 (head, div2*)>
    ```
  - A "div1" elem belsejében a "head" elem után legalább egy vagy több "div2" elem lehet (1 vagy több)
    ```
    <!ELEMENT div1 (head, div2+)>
    ```
- A megkötések listája zárójelezhető. Példa: a "div1" elem belsejében először egy "head" elemnek kell jönnie, majd "p", "list" és "note" elemek következhetnek. Opcionálisan az utolsó elemek "div2" típusúak lehetnek, de el is maradhatnak.
  ```
  <!ELEMENT div1 (head, (p | list | note)*, div2*)>
  ```

Elemek definíciója (3)

- Kevert tartalom: az elem tartalma karaktersorozatok és beágyazott elemek keveréke lehet. A beágyazott elemek sorrendje tetszőleges, de típusuk előírható. Példa: a "p" elem belsejében karaktersorozatok, valamint tetszőleges számú és sorrendű "a", "ul", "b", "i" és "em" elemek lehetnek.

```
<!ELEMENT p (#PCDATA | a | ul | b | i | em)*>
```

Elemek definíciója (4)

- Egy elem lehet kötelezően üres. Ekkor az elemnek nem lehet tartalma. Példa: csak a
 vagy
</br> illeszkedik.
  `<!ELEMENT br EMPTY>`
- Dönthetünk úgy, hogy az elem tartalmának ellenőrzését kivonjuk a DTD validálás alól. Ekkor az elemnek bármilyen helyesen formált tartalma lehet, de a benne szereplő elemeknek mind szerepelnie kell a DTD-ben. Példa:
  `<!ELEMENT container ANY>`

Attribútumok definíciója

- Az elem attribútumait külön nyelvi szerkezetben definiáljuk, majd az elemhez kötjük.
- Attribútumok típusai:
  - Felsorolás: az attribútum által felvehető karaktersorozatok felsorolása. Példa: (bullets | ordered | glossary)
  - CDATA: karakteres adat: az attribútum értéke egy tetszőleges karaktersorozat.
  - ID: karaktersorozat, amely az XML dokumentumban csak egyetlen attribútumnak lehet értéke (minden ID típusú attribútumnak más értéke kell legyen).
  - IDREF: karaktersorozat, amely megegyezik egy ID attribútum értékével.
  - IDREFS: IDREF felsorolás (elválasztókarakter: szóköz).
  - ENTITY: nem elemzett külső entitás az érték
  - ENTITIES: nem elemzett külső entitások listája (elválasztókarakter: szóköz)
  - NMTOKEN: azonosító
  - NMTOKENS: azonosítók listája
  - NOTATION: az attribútum értékének meg kell felelnie egy NOTATION azonosítónak (pl. gif, 13. dia)

Attribútumok definíciója (2)

- Az attribútum lehet kötelező vagy opcionális.
  - Kötelező attribútum: #REQUIRED
  - Opcionális attribútum: #IMPLIED
  - Előírt értékű: #FIXED. Az attribútum értékének kötelezően egy megadott értéknek kell lennie.
- Példák:
  - A "termdef" elemnek két attribútuma van: az "id" nevű, amely kötelezően megadandó és ID típusú valamint a "name" nevű, amelyik tetszőleges karaktersorozat lehet és opcionális.

```
<!ATTLIST termdef
          id      ID      #REQUIRED
          name    CDATA   #IMPLIED>
```

  - A "list" elemnek egy attribútuma van, melynek neve "type". Ennek értéke "bullets", "ordered" vagy "glossary" lehet, az "ordered" érték alapértelmezett.

```
<!ATTLIST list
          type    (bullets | ordered | glossary)
"ordered">
```

Feltételes szekciók

- A DTD egyes szekcióinak jelenléte feltételekhez köthető. Ha a feltétel INCLUDE, akkor a feltételes szekció megjelenik a DTD-ben, ha IGNORE, akkor nem.
- Példa: minthogy a "draft" paraméter-entitás van INCLUDE-ra állítva, a "book" elem tartalmazhat "comments" elemeket is.

```
<!ENTITY % draft 'INCLUDE' >
<!ENTITY % final 'IGNORE' >
<![%draft;[
<!ELEMENT book (comments*, title, body, supplements?)>
]]>
<![%final;[
<!ELEMENT book (title, body, supplements?)>
]]>
```

Gyakorlat

- Egy dokumentumban könyvek adatait tároljuk (0 vagy több könyv).
- Egy könyvnek címe, kiadási éve, kiadója, szerzői vannak (egy vagy több). Ezen felül rövid szöveges összefoglaló is megadható opcionálisan. Ezek az adatok ebben a sorrendben vannak megadva minden egyes könyvnél.
- Minden könyvnek ezen felül egyedi azonosítója is van.
- Tervezzen XML adatszerkezetet a könyv adatainak leírására és készítse el az adatszerkezetet validáló DTD-t!

books.xml

```
<?xml version="1.0"?>
<!DOCTYPE books SYSTEM "books.dtd">
<books>
  <book id="id1">
    <title>My first XML book</title>
    <published>2001</published>
    <publisher>&lt;XML&gt; Publishing, Inc.</publisher>
    <author>John Doe</author>
    <author>Richard Doe</author>
    <abstract>XML for dummies</abstract>
  </book>
  <book id="xy2z">
    <title>Advanced XML book</title>
    <published>2002</published>
    <publisher>&lt;XML&gt; Publishing, Inc.</publisher>
    <author>Richard Doe</author>
    <abstract>Hairy details about XML you never wanted
to know</abstract>
  </book>
</books>
```

books.dtd

```
<!ELEMENT books (book)*>
<!ELEMENT book
(title,published,publisher,author+,abstract?)>
<!ATTLIST book
          id ID #REQUIRED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT published (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT abstract (#PCDATA)>
```