O RAZŠIRLJIVEM OZNAČEVALNEM JEZIKU XML IN NJEGOVI UPORABI

Tomaž Mohorič, Marjan Krisper
Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, Tržaška 25, 1000 Ljubljana
{tomaz.mohoric, marjan.krisper}@fri.uni-lj.si

Izvleček

Razširljivi označevalni jezik XML nudi osnovo za kreiranje dokumentov in dokumentnih sistemov. Deluje predvsem v dveh smereh. Prva je skrb za sintakso pri označevanju dokumentov, druga pa skrb za deklariranje dokumentnih struktur. Nič manj pomembna ni uvedba imenskih prostorov, ki omogočajo obstoj različnih dokumentov, in njihovih interakcij v ustreznih imenskih prostorih. Morda je ena izmed najpomembnejših lastnosti prav sposobnost shranjevanja in izmenjave podatkov med uporabniki pod nadzorom definicij dokumentnih tipov in shem XML.

Abstract

Extensible Markup Language (XML) provides a foundation for creating documents and document systems. It functions in two directions mainly. The first one is the provision of syntax for document markup, and the second one is the provision of syntax for declaring the structures of documents. Not less important is an introduction of namespaces, enabling the coexistence of different XML documents and their interactions in corresponding namespaces. Maybe one of its strongest points is the ability for storing and exchanging data among users under control of Document Type Definitions and XML Schemas.

■ ■ ■

1. Uvod

Jezik XML (eXtensible Markup Language) je izpeljanka jezika SGML (Standard Generalized Markup Language), ki se uporablja za kreiranje in označevanje tehnične (in tudi druge) dokumentacije. Iz jezika SGML so, na primer, izvedeni tudi CML (Chemical Markup Language), VML (Vector Markup Language), MathML (Mathematical Markup Language) in še vrsta drugih sorodnih jezikov. XML je metajezik - jezik, s pomočjo katerega se definira konkreten označevalni jezik, neposredno uporaben za označevanje določene vrste dokumentov. Dokumenti XML imajo vlogo vsebnikov, v katere se shranjujejo podatki. Po tej plati so podobni tabelam v relacijski podatkovni bazi in objektom v objektno usmerjeni podatkovni bazi. Metajezik obsega množico sintaktičnih pravil za kreiranje konkretnega označevalnega jezika. Po svoji obliki je XML samoopisni jezik, ki z uporabo oznak pojasnjuje pomen podatkov v dokumentu. Oznake, s katerimi se dokument označi, se smejo izbrati poljubno, smiselno pa je, da hkrati odražajo tudi pomen podatka, tako da ga lahko razumeta hkrati človek in tudi računalnik, natančneje programska oprema.

2. Oznake

Vrednost podatka, ki ga želimo zapisati v dokument, se mora nahajati med dvema oznakama - začetno oznako in končno oznako (tag). Par oznak, med katerima se nahaja vrednost, imenujemo *element*:

```
<Ime>Peter</Ime>
<Priimek>Kovač</Priimek>
```

Tako predstavljata <Ime> in </Ime> začetno in končno oznako, Peter je pa vrednost elementa.

XML razlikuje med velikimi in malimi črkami, zato morata biti začetna in končna oznaka zapisani na enak način. Sintaktično pravilno zapisane oznake so tudi:

```
<ime>Peter</ime>
<PRIimek>Kovač</PRIimek>
```

V dokumentu se razen elementov lahko shranjujejo tudi atributi. Če naj bo v dokumentu shranjen element (npr. knjiga), ki mu pripada eden ali več atributov (npr. ISBN, leto izdaje), je zapis naslednji:

```
<Knjiga ISBN="961-6046-02-0" leto="1995"/>
```

V zgornjem zapisu je *Knjiga* element, *ISBN* in *leto* sta pa atributa. Omenjeni razpored vrednosti in atributov v oznaki se imenuje tudi "prazna oznaka", ker ne obsega tudi vrednosti elementa. Atribut je sestavljen iz para ime-vrednost, med njima pa stoji enačaj.

Pri poimenovanju oznak veljajo naslednja pravila:

- imena elementov se smejo pričeti s poljubno črko ali podčrtajem;
- po prvem znaku lahko sledijo črke, številke, pike, pomišljaji, podčrtaji ali dvopičja;

- imena elementov ne smejo vsebovati kontrolnih znakov kot so npr. (white space, cariage return, line feed, form feed);
- imena elementov se ne smejo pričeti z imenom *XML* ali *xml*, ker je to ključna beseda, ki se uporablja le v posebnih primerih, npr. pri deklaracijah dokumenta.

3. Deklariranje dokumenta XML

Dokument se prične z deklaracijo XML, ki obsega verzijo dokumenta XML, podatek, ali je dokument samostojen ali ne, ter način kodiranja:

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

Za naše kraje je primerno npr. kodiranje ISO-8859-2, ki je prava podmnožica kodiranja UTF-8. Določili *standalone* in *encoding* sta praviloma opcijski, vendar ju je v nekaterih primerih potrebno uporabiti. Določilo *standalone="yes"* pomeni, da je dokument povsem samostojen, določilo *standalone="no"* pa pomeni, da je dokument vezan na kak drug dokument, npr. na pripadajoči DTD (Document Type Definition).

Iz predhodno opisanih fragmentov se lahko sestavi samostojen dokument po imenu Revija. *Revija* je hkrati tudi korenski element, ki zajema vse druge elemente v dokumentu.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Revija>
    <Avtor>
        <Ime>Peter</Ime>
        <Priimek>Kovač</Priimek>
        <knjiga ISBN="961-6046-02-0" leto="1995"/>
    </Avtor>
    <Avtor>
        <Ime>Miha</Ime>
        <Priimek>Pečar</Priimek>
        <knjiga ISBN="0-201-42777-X" leto="1995"/>
    </Avtor>
</Revija>
```

4. Dobro oblikovani dokument

Dokument sodi med dobro oblikovane (well-formed) dokumente, če v njem ni sintaktičnih napak. Nekaj poglavitnih napotkov, ki omogočajo kreiranje dobro oblikovanih dokumentov:

- dokument se mora pričeti z deklaracijo XML;
- elementu, ki vsebuje podatke (vrednost), morata pripadati začetna in končna oznaka;
- element, ki ne vsebuje podatkov (vrednosti) pač pa le atribute, se ves nahaja v posamični oznaki, ki se prične z <... in konča z .../>; v tem primeru govorimo o prazni oznaki ali o elementu brez podatkov;
- dokument mora obsegati natanko en element, ki vsebuje vse ostale elemente - tak element se imenuje *korenski element*;
- element je lahko vgnezden v drugem elementu;
- vrednost atributa se mora nahajati v dvojnih narekovajih;
- obstajajo naslednje *znakovne entitete*, ki lahko po potrebi nadomestijo osnovne simbole:

Entiteta	Simbol	Pomen
&gt;	>	večji
&lt;	<	manjši
&amp;	&	konjunkcija (ampersand)
&apos;	'	apostrof
&quot;	"	dvojni narekovaj

Ker imata začetna in končna oznaka < ... > svoj rezervirani pomen, ki se uporablja pri zapisovanju oznak, je potrebno v primeru uporabe simbolov "večji" in "manjši" znotraj oznake nadomestiti simbol "večji"z znakovno entiteto ">" in simbol "manjši" z znakovno entiteto "<". Podobna pravila veljajo tudi za enojni in dvojni narekovaj ter konjunkcijo.

5. Veljavni dokument

Podobno kot pri drugih podatkovnih modelih tudi pri dokumentih XML obstajajo omejitve, ki omejujejo dopustno vsebino dokumenta. V dokument se smejo zapisati le tisti podatki, ki so v skladu s shemo dokumenta. XML pozna dve vrsti shem. Prva shema je že uveljavljena in v široki uporabi. Imenuje se DTD (Document Type Definition). Druga shema, ki je popolnejša od DTD in obsega več opcij za uveljavitev integritetnih omejitev, pa se imenuje XML Schema. Dokumenti, ki so zapisani v skladu s shemo, so veljavni (valid) dokumenti.

6. Elementi

Element je grupa, sestavljena iz enega ali več podelementov ali podgrup, obsega pa lahko tudi znakovne podatke ter ključni besedi EMPTY (prazen element) in ANY (katerikoli element). EMPTY pomeni, da element ne vsebuje podelementov ali znakovnih podatkov, pač pa lahko vsebuje atribute. ANY pomeni, da lahko element vsebuje nič ali več podelementov kateregakoli tipa, vključno z znakovnimi podatki.

```
Grupa:      <!ELEMENT Avtor (Ime, Priimek, knjiga)>
Znakovni podatek:  <!ELEMENT Ime (#PCDATA)>
EMPTY:      <!ELEMENT knjiga EMPTY>
ANY:        <!ELEMENT uvod ANY>
```

Grupe so lahko zaporedja ali izbire podelementov (podgrup). Primer zaporedja je predstavljen že pri grupi, primer izbire pa je:

```
<!ELEMENT knjiga (prolog | poglavje| epilog)>
```

V grupo so lahko vgnezdene izbire in v izbiro so lahko vgnezdene grupe. Nad grupami, podgrupami in podelementi se lahko aplicirajo operatorji: ? (opcija), + (eden ali več), * (nič ali več).

Opcija: podelement knjiga lahko obstaja ali pa tudi ne
```
<!ELEMENT Avtor (Ime, Priimek, knjiga?)>
```

Eden ali več: podelement knjiga se pojavlja najmanj enkrat
```
<!ELEMENT Avtor (Ime, Priimek, knjiga+)>
```

Nič ali več: podelement knjiga se ne pojavlja ali pa se pojavlja enkrat ali večkrat
```
<!ELEMENT Avtor (Ime, Priimek, knjiga*)>
```

7. Atributi

Elementi lahko vsebujejo atribute. Na seznamu atributov ATTLIST se lahko nahaja eden ali več atributov, ki pripadajo posameznemu elementu. Naslednji zgled prikazuje atributni seznam, ki obsega dva atributa - ISBN in leto. Atributi so lahko po vrednosti opcijski, zahtevani ali fiksni. Opcijski atributi imajo lahko privzeto vrednost, fiksni atributi pa morajo imeti privzeto vrednost. Klasificirajo se v atribute brez privzete vrednosti, v atribute s privzeto vrednostjo ter obvezne in fiksne atribute.

```
<!ATTLIST knjiga
    ISBN CDATA #REQUIRED
    leto CDATA #REQUIRED
>
```

Vsakemu atributu pripada tudi tip. Tip je lahko znakovni ali od uporabnika definirani naštevni tip. Posebne vrste atributa sta ID in IDREF - atributi te vrste kažejo z enega elementa na drugega. Vrednost atributa IDREF na kažoči element je enaka kot vrednost atributa ID na pokazani element.

8. DTD za dokument Revija

S pomočjo znanja, pridobljenega v zgornjih vrsticah, je možno dešifrirati pomen DTDja - sheme - ki določa dopustno vsebino dokumentov.

```
<!ELEMENT Revija (Avtor+)>
<!ELEMENT Avtor (Ime, Priimek, knjiga)>
<!ELEMENT Ime (#PCDATA)>
<!ELEMENT Priimek (#PCDATA)>
<!ELEMENT knjiga EMPTY>
<!ATTLIST knjiga
    ISBN CDATA #REQUIRED
    leto CDATA #REQUIRED
>
```

Pomen posameznih vrstic je naslednji:

1. *vrstica*:
   korenski element dokumenta je imenovan *Revija*, v dokumentu pa se lahko nahajajo podatki o enem ali več avtorjih;
2. *vrstica*:
   avtor ima *Ime*, *Priimek* in (je napisal) *knjigo*;
3. *in 4. vrstica*:
   enostavni elementi vsebujejo tekst, ki je poimenovan PCDATA (parsed character data), sestavljeni elementi vsebujejo druge elemente, lahko tudi tekst in druge elemente; *Ime* je tipa PCDATA, *Priimek* je tipa PCDATA; tekst tipa PCDATA je v bistvu katerikoli tekst, ki ni označevalni (markup) tekst;
5. *vrstica*:
   prazen element *knjiga*;
6. *vrstica*:
   lista atributov, ki pripadajo knjigi: *ISBN* in *leto* sta oba tipa CDATA (character data), atributa pa sta obvezna.

Omejitve, ki izhajajo iz DTDja, pa so naslednje:

- korenski element je *Revija*;
- *Revija* obsega enega ali več *avtorjev*;
- *Avtor* je sestavljen iz elementov *Ime*, *Priimek*, *knjiga*;
- *knjiga* obsega dva atributa *ISBN* in *leto*;
- elementa *Ime*, *Priimek* sta tipa PCDATA;
- elementa *ISBN*, *leto* sta tipa CDATA.

9. Vključenost DTDja v dokument

DTD je lahko vključen v dokument na dva načina. Lahko je vgnezden med deklaracijo XML (<?xml ... ?>) in podatkovnim delom dokumenta, lahko pa se DTD nahaja v posebni datoteki in je vključen v dokument s pomočjo reference. Naslednji primer prikazuje neposredno vključenost DTDja v dokument:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE Revija [
<!ELEMENT Revija (Avtor+)>
<!ELEMENT Avtor (Ime, Priimek, knjiga)>
<!ELEMENT Ime (#PCDATA)>
<!ELEMENT Priimek (#PCDATA)>
<!ELEMENT knjiga EMPTY>
<!ATTLIST knjiga
    ISBN CDATA #REQUIRED
    leto CDATA #REQUIRED>
```

```
]>
<Revija>
    <Avtor>
        <Ime>Peter</Ime>
        <Priimek>Kovač</Priimek>
        <knjiga ISBN="961-6046-02-0" leto="1995"/>
    </Avtor>
    <Avtor>
        <Ime>Miha</Ime>
        <Priimek>Pečar</Priimek>
        <knjiga ISBN="0-201-42777-X" leto="1995"/>
    </Avtor>
</Revija>
```

DTD pa se lahko iz praktičnih razlogov shrani v posebni datoteki, na primer zato, da je dosegljiv tudi preostalim dokumentom, ki so osnovani na tem istem DTDju. Pri naslednjem zgledu je DTD shranjen v datoteki *Revija1.dtd*:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE Revija SYSTEM "Revija1.dtd" >
<Revija>
    <Avtor>
        <Ime>Peter</Ime>
        <Priimek>Kovač</Priimek>
        <knjiga ISBN="961-6046-02-0" leto="1995"/>
    </Avtor>
    <Avtor>
        <Ime>Miha</Ime>
        <Priimek>Pečar</Priimek>
        <knjiga ISBN="0-201-42777-X" leto="1995"/>
    </Avtor>
</Revija>
```

10. Shema XML za dokument Revija

Shema XML igra podobno vlogo kot DTD in je integritetna omejitev na dopustno vsebino dokumenta Revija. Pri relacijskih podatkovnih bazah se z jezikom DDL definirajo relacijske sheme za tabele skupaj z integritetnimi omejitvami. Dokument XML obsega enega ali več elementov, ki vsebujejo atribute, ostale elemente in tekstovne zapise. Z uporabo jezika Shema XML lahko avtor definira strukturo in dovoljene podatkovne tipe v dokumentih. V nadaljevanju je predstavljena shema za dokument Revija. Vsaki shemi praviloma pripada eden ali več primerkov te sheme – konkretnih dokumentov. Uporabniki praviloma izpolnijo le s shemo definirani obrazec, v katerem se potem nahajajo podatki, ki jih želijo shraniti, posredovati drugim uporabnikom in podobno, shema, ki preverja pravilnost izpolnjevanja obrazca, pa ostaja ves čas ena sama.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/
XMLSchema" elementFormDefault="qualified">
    <xsd:element name="Avtor">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="Ime"/>
                <xsd:element ref="Priimek"/>
                <xsd:element ref="Knjiga"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Ime" type="xsd:string"/>
    <xsd:element name="Knjiga">
        <xsd:complexType>
            <xsd:attribute name="ISBN" type="xsd:string"
            use="required"/>
            <xsd:attribute name="leto" type="xsd:short"
            use="required"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Priimek" type="xsd:string"/>
    <xsd:element name="Revija">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="Avtor" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Tako kot DTD je tudi Shema XML sama zase veljaven dokument. Iz nje lahko izluščimo, da mora za vsak dokument, ki se ujema s shemo, veljati:

- korenski dokument je *Revija*;
- število avtorjev v reviji je neomejeno;
- ime in priimek avtorja sta tipa *niz* (string);
- atributa knjige *ISBN* in *leto* sta obvezna in oba tipa *niz*.

11. Imenski prostor

Imenski prostor (XML Namespace) je zbirka imen tipov elementov in imen atributov. Namen imenskih prostorov je razreševanje problemov poimenovanja v dokumentih XML, ki vsebujejo tipe elementov in atribute iz različnih jezikov XML. Nabor oznak, ki se uporabljajo pri enem konkretnem jeziku, se namreč lahko razlikuje od nabora oznak pri drugem konkretnem jeziku. Primer dveh konkretnih, med seboj razlikujočih se jezikov, ki uporabljata enake oznake vendar v različnih pomenih, je na primer naslednji:

Jezik A

```
<?xml version="1.0"?>
<Naslov>
    <Ulica>Tržaška 25</Ulica>
    <Mesto>Ljubljana</Mesto>
    <Država>Slovenija</Država>
    <PŠT>SI-1000</PŠT>
</Naslov>
```

Jezik B

```
<?xml version="1.0"?>
<Strežnik>
    <Ime>PC02</Ime>
    <Naslov>300.2.76.14</Naslov>
</Strežnik>
```

Vsak izmed omenjenih dveh jezikov je drugačen. Pri obeh nastopa oznaka *Naslov* kot ime tipa elementa, pa vendar je pomen omenjenih oznak različen. Razlikovanje nastopa pri modelu, pomenu in interpretaciji tipa elementa v aplikaciji. Pa vendar se lahko omenjena dva dokumenta združita v enega, če se dokumente označi s prefiksi, kot kaže naslednji sestavljeni dokument:

```
<Oddelek>
    <Ime>FRI</Ime>
    <nsl:Naslov xmlns: nsl ="http://fri.uni-lj.si">
        <nsl:Ulica>Tržaška 25</nsl:Ulica>
        <nsl:Mesto>Ljubljana</nsl:Mesto>
        <nsl:Država>Slovenija</nsl:Država>
        <nsl:PŠT>SI-1000</nsl:PŠT>
    </nsl:Naslov>
    <str:Strežnik xmlns: str ="http://fe.uni-lj.si">
        <str:Ime>PC03</ str:Ime>
        <str:Naslov>123.45.67.8</str:Naslov>
    </str:Strežnik>
</Oddelek>
```

Imenski prostor se v slednjem primeru identificira s pomočjo identifikatorja URI (Universal Resource Identifier). Vsak tip elementa ali ime atributa pa se v prostoru XML enolično identificira z dvodelnim imenom, s pomočjo URI in s pomočjo svojega lokalnega imena. Imenski prostori se potem deklarirajo z atributom *xmlns* (XML Namespace), ki poveže poljuben prefiks z določenim imenskim prostorom. Imenski prostori namreč nimajo druge funkcije, kot zgolj poskrbeti za dvodelni sistem poimenovanja tipov elementov in atributov.

Pomen jezika XML ne sega le na področje kreiranja strani v spletu, pač pa je uporaben tudi kot univerzalni format, ki aplikacijam različnih vrst omogoča nemoteno izmenjavo podatkov. Med drugim podpira tudi podatkovne transformacije, kot so XSLT (Extensible Stylesheet Language Transformations) in CSS (Cascading Style Sheets). Odlike jezika XML so preprostost, razširljivost, povezljivost (interoperability) in odprtost. XML se lahko uporablja na različnih platformah v povezavi s široko paleto orodij v različnih programskih okoljih, pogosto v povezavi z Javo. Podpira vrsto standardov za kodiranje znakov, kot je na primer UTF-8. Razširljivost se kaže v dveh smereh - omogoča kreiranje shem XML in DTD, hkrati pa je jezik razširljiv z dodatnimi funkcijami, kot so stilni listi, povezovanje in reference. Standard XML je tudi popolnoma odprt in dosegljiv v spletu.

Literatura

[1] Elliotte Rusty Harold (1999): XML Bible, IDG Books Worldwide, Inc.

[2] Erik T. Ray (2001): Learning XML, O'Reilly & Associates

[3] Kevin Williams, Michael Brundage, Patrick Dengler, Jeff Gabriel, Andy Hoskinson, Michael Kay, Thomas Maxwell, Marcelo Ochoa, Johnny Papa, Mohan Vanmane (2000): Professional XML Databases, Wrox Press Inc.

[4] Mark Birbeck, Michael Kay, Stephen F. Mohr, Jonathan Pinnock, Brian Loesgen, Steven Livingstone, Didier Martin, Nikola Ozu, Mark Seabourne, David Baliles (2000): Professional XML, Wrox Press Inc.

[5] Eric van der Vlist (2000): Using W3C XML Schema, http://www.xml.com/pub/a/2000/11/29/schemas/part1.html

[6] Lars Marius Garshol (1999): Introduction to XML, http://www.stud.ifi.uio.no/člmariusg/download/xml/xml_eng.html

[7] Michael J. Young (2000): Step by Step XML, Microsoft Press

[8] Ronald Bourret (2000): Declaring Elements and Atributes in an XML DTD, http://www.rpbourret.com/xml/xmldtd.htm

[9] Ronald Bourret (2000): XML Namespaces FAQ, http://www.rpbourret.com/xml/NamespacesFAQ.htm

[10] Simon St. Laurent (2001): XML: A Primer, M&T Books

[11] Steven Holzner (2000): Inside XML, New Riders Publishing

[12] Tim Bray (1999): XML Namespaces by Example, http://www.xml.com/pub/a/1999/01/namespaces.html

[13] Yasser Shohoud (2000): XML's Grand Schema, XML Magazine, Vol 1, No3

◆

Dr. Tomaž Mohorič je diplomiral leta 1970, magistriral leta 1973 in doktoriral leta 1986 na Fakulteti za elektrotehniko. Od leta 1987 je docent za področje računalništva in informatike. Področje njegovih raziskav se nanaša na podatkovne baze, razvoj informacijskih sistemov in v zadnjem času na področje razširljivega označevalnega jezika (XML).

◆

Dr. Marjan Krisper je predstojnik katedre za informatiko na Fakulteti za računalništvo in informatiko Univerze v Ljubljani in od ustanovitve leta 1992 predstojnik Laboratorija za informatiko. Bil je soustanovitelj prve slovenske računalniške revije BIT in Revije za razvoj RR. Je član več znanstvenih in strokovnih združenj, med drugim ustanovitveni član AIS (Association for Information Systems), Slovenskega društva INFORMATIKA, Društva za umetno inteligenco in INFOS-a. Vodi številne projekte razvoja informacijskih sistemov in uvajanja metodologij razvoja v največjih sistemih v gospodarstvu, državni upravi in javnem sektorju.