Az XML Schema deklarációs nyelv

oktatási segédlet

Mészáros Tamás

meszaros@mit.bme.hu

Budapesti Műszaki Egyetem

Méréstechnika és Információs Rendszerek Tanszék

Az XML Schema deklarációs rendszer

- Az alap XML szabvány deklarációs rendszerének (DTD) problémái:
  - nem XML szintaxisú
  - nem támogatja az XML Névtér szabványt
  - nincs igazi adattipizálás
  - komplex deklarációk készítését gyengén támogatja

- Megoldás: XML Schema szabvány (W3C, 2001)
  - XML formátumú dokumentum
  - gazdag típusválaszték
  - új típusok létrehozási lehetősége
  - új öröklési (egymásba ágyazhatósági) modellek
  - XML Névtér támogatás (kötelező használat)

XML Névterek

- XML Namespaces
  - probléma: több deklaráció összekapcsolása esetén névütközés
  - megoldás: névterek, melyek egyértelműen azonosítják a címkék szemantikáját
  - formai újítás: a címkék ellátása névtér előtaggal (prefix)
  - prefix deklaráció külső erőforrás megadásával
  - tipikus használat: XSL, Schema deklarációk

- Az XML Névterek alkalmazása
  - deklaráció: attribútum megadásával

    ```
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ```

  - alkalmazás: alapértelmezett névtér vagy explicit jelölt névtér

    ```
    <xs:schema>...</xs:schema>
    ```

A Schema és a DTD viszonya (példa)

DTD deklaráció

```
<!ELEMENT note (to, from, heading, body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

XML dokumentum

```
<?xml version="1.0"?>
<note>
<to>Juli</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Schema deklaráció

```
<?xml version="1.0"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSc
    hema">
<xs:element name="note">
    <xs:complexType>
      <xs:sequence>
          <xs:element name="to"
                      type="xs:string"/>
          <xs:element name="from"
                      type="xs:string"/>
          <xs:element name="heading"
                      type="xs:string"/>
          <xs:element name="body"
                      type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

Egy összetett Schema példa

```
<?xml version="1.0"?>
<xsd:schema
     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="purchaseOrder"
    type="PurchaseOrderType"/>
 <xsd:element name="comment" type="xsd:string"/>
 <xsd:complexType name="PurchaseOrderType">
  <xsd:sequence>
   <xsd:element name="shipTo" type="USAddress"/>
   <xsd:element name="billTo" type="USAddress"/>
   <xsd:element ref="comment" minOccurs="0"/>
   <xsd:element name="items"  type="Items"/>
  </xsd:sequence>
  <xsd:attribute name="orderDate"
     type="xsd:date"/>
 </xsd:complexType>
 <xsd:complexType name="USAddress">
  <xsd:sequence>
   <xsd:element name="name"   type="xsd:string"/>
   <xsd:element name="street" type="xsd:string"/>
   <xsd:element name="city"   type="xsd:string"/>
   <xsd:element name="state"  type="xsd:string"/>
   <xsd:element name="zip"    type="xsd:decimal"/>
  </xsd:sequence>
  <xsd:attribute name="country" type="xsd:NMTOKEN"
     fixed="US"/>
 </xsd:complexType>
```

```
<xsd:complexType name="Items">
  <xsd:sequence>
   <xsd:element name="item" minOccurs="0"
     maxOccurs="unbounded">
    <xsd:complexType>
     <xsd:sequence>
      <xsd:element name="productName"
     type="xsd:string"/>
      <xsd:element name="quantity">
       <xsd:simpleType>
        <xsd:restriction base="xsd:positiveInteger">
         <xsd:maxExclusive value="100"/>
        </xsd:restriction>
       </xsd:simpleType>
      </xsd:element>
      <xsd:element name="USPrice"
     type="xsd:decimal"/>
      <xsd:element ref="comment"   minOccurs="0"/>
      <xsd:element name="shipDate" type="xsd:date"
     minOccurs="0"/>
     </xsd:sequence>
     <xsd:attribute name="partNum" type="SKU"
     use="required"/>
    </xsd:complexType>
   </xsd:element>
  </xsd:sequence>
 </xsd:complexType>
 <!-- Stock Keeping Unit, a code for identifying
     products -->
 <xsd:simpleType name="SKU">
  <xsd:restriction base="xsd:string">
   <xsd:pattern value="\d{3}-[A-Z]{2}"/>
  </xsd:restriction>
 </xsd:simpleType>
</xsd:schema>
```

A Schema példa XML dokumentuma

```
<?xml version="1.0"?>
<purchaseOrder orderDate="1999-10-20">
    <shipTo country="US">
        <name>Alice Smith</name>
        <street>123 Maple Street</street>
        <city>Mill Valley</city>
        <state>CA</state>
        <zip>90952</zip>
    </shipTo>
    <billTo country="US">
        <name>Robert Smith</name>
        <street>8 Oak Avenue</street>
        <city>Old Town</city>
        <state>PA</state>
        <zip>95819</zip>
    </billTo>
    <comment>Hurry, my lawn is going wild!
     </comment>
    <items>
        <item partNum="872-AA">
            <productName>Lawnmower</productName>
            <quantity>1</quantity>
            <USPrice>148.95</USPrice>
            <comment>Confirm this is
     electric</comment>
        </item>
        <item partNum="926-AA">
            <productName>Baby
     Monitor</productName>
            <quantity>1</quantity>
            <USPrice>39.98</USPrice>
            <shipDate>1999-05-21</shipDate>
        </item>
    </items>
</purchaseOrder>
```

- a Schema deklaráció nem eredményez más XML dokumentumot
- az előny a dokumentum feldolgozásakor, ellenőrzésekor jelentkezik

A Schema deklarációs rendszer fogalmi alapjai

- Cél névtér (target namespace)
  - A deklarált nyelv névtere, amelynek a részei a megalkotott Schema dokumentum deklarációi (típusok, elemek, attribútumok, stb.)

- Minősített név (qualified name)
  - Azon nevek, amelyeket névtér segítségével értelmezünk.
  - Ilyenek pl. a minősítő előtaggal (prefix) ellátott nevek (xhtml:table).
  - A minősítés egyértelművé teszi, hogy a nevek milyen névtérhez tartoznak.

- XML dokumentum példány (instance document)
  - a deklaráció alapján elkészített XML dokumentum
  - a deklarációs cél- és az XMLSchema-instance névtérre támaszkodik

A Schema deklaráció felépítése

- XML deklaráció

```
<?xml version="1.0"?>
```

- Schema gyökérelem

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

- opcionális dokumentáció a teljes deklarációról

```
<xsd:annotation>
<xsd:documentation>
...
</xsd:documentation>
</xsd:annotation>
```

- deklarációk: elemek és attribútumok, típusok, stb.

A Schema és a deklarációs névterek

- A Schema szabványban rögzített címkék (a deklarációs nyelv elemei) az `XMLSchema` névtérben vannak rögzítve

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

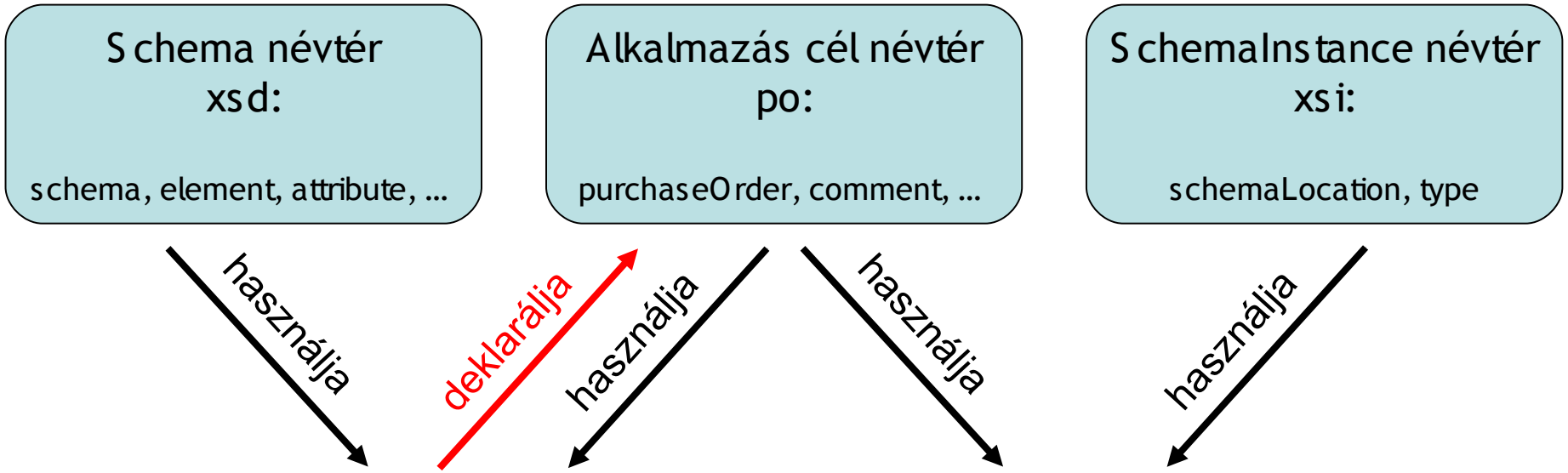
- A Schema dokumentum deklarálja az általa leírt cél névteret:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:po="http://www.example.com/PO1"
    xsd:targetNamespace="http://www.example.com/PO1">
```

- Schema deklaráció alapján létrehozott XML dokumentumok a cél névtérre és az XMLSchema-instance névtérre hivatkozhatnak

```
<?xml version="1.0"?>
<po:purchaseOrder
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:po="http://www.example.com/PO1"
    orderDate="1999-12-01">
```

Névterek, deklaráció és dokumentum példány



Schema deklaráció

```
<?xml version="1.0"?>
<xsd:schema
   targetNamespace="http://www.example.com/PO1"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:po="http://www.example.com/PO1">

<xsd:element name="purchaseOrder"
   type="po:PurchaseOrderType" />

...
```

XML dokumentum példány

```
<?xml version="1.0"?>
<po:purchaseOrder orderDate="1999-10-20"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-i
   xmlns:po="http://www.example.com/PO1"
   xsi:schemaLocation="http://www.example.com/PO

...
```

Hivatkozás a Schema deklarációra

- A Schema dokumentum a saját névterét is használhatja:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:po="http://www.example.com/PO1"
    xsd:targetNamespace="http://www.example.com/PO1"
    xsd:elementFormDefault="unqualified"
    xsd:attributeFormDefault="unqualified">
```

- A dokumentum példány megadja a névteret és a deklaráció elérését

```
<?xml version="1.0"?>
<po:purchaseOrder
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:po="http://www.example.com/PO1"
    xsi:schemaLocation="http://www.example.com/PO1 po1.xsd"
    orderDate="1999-12-01">
```

Schema deklarációk

- A Schema egy moduláris deklarációs rendszer
  (sok objektum-orientált koncepciót átvett)
  - egyszerűbb komponensekből (elemek, attribútumok) összetett komponensek építhetők
  - a beépített típusok szűkíthetők, bővíthetők
  - saját típusok is létrehozhatók

- A Schema a struktúra mellett az adattípusokat is pontosan rögzíti
  - a tipizálást meghatározó egyszerű típusok és a lehetséges struktúrákat leíró összetett típusok deklarációja szétválik
  - (maga a szabvány is két elkülönített részt szentel ezeknek)

- Globális és lokális deklarációk
  - közvetlenül a *schema* gyökérelem alatti, ún. *globális* deklarációk bárhol felhasználhatók, bármelyikük lehet gyökérelem az XML példányokban
  - a globális deklarációk gyermekei a lokális deklarációk

Egyszerű típusok deklarációja és származtatása

- Egyszerű típusok, amelyeknek nincs belső XML struktúrája
  - atom
  - lista
  - unió
- A Schema nyelv számos beépített egyszerű (atom) típussal rendelkezik

  `string, integer, float, boolean, date, time`, stb.
- Az egyszerű típusokból továbbiakat származtathatunk szűkítéssel
- A lista (xsd:list) egyszerű típussal rendelkező adatokat kapcsol össze
  - vannak beépített lista típusok
  - az egyszerű típusok felhasználásával további típusok is deklarálhatók
  - a listákra többféle megkötést is tehetünk (méret, minták, stb.)
- Az unió (xsd:union) egyszerű típusok egy halmaza, melynek elemei közül bármelyik szerepelhet a dokumentum példányban.
- Speciális egyszerű típus: anySimpleType (nem származtatható, speciális eseteket kivéve kerülendő az alkalmazása)

Egyszerű típusok deklarációja – példák

```
<xsd:simpleType name="myInteger">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="10000"/>
    <xsd:maxInclusive value="99999"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="listOfMyIntType">
  <xsd:list itemType="myInteger"/>
</xsd:simpleType>
<!-- példa: <listOfMyInt>20003 15037 95977 95945</listOfMyInt> -->

<xsd:simpleType name="SKU">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d{3}-[A-Z]{2}"/>
  </xsd:restriction>
</xsd:simpleType>
```

Komplex típusok deklarációja

- Komplex típus az, amelynek van beágyazott gyermek eleme vagy rendelkezik attribútummal.

- Összetett típusok létrehozhatók egyszerű típusokból
  - elemekhez attribútum hozzárendelésével (az elem tartalmának nem változik a típusa, de kapcsolódik hozzá egy attribútum)
  - elemek összekapcsolásával
    - sorozat (xsd:sequence)
    - elemekből álló struktúra (complexContent)
    - akár karakteres adatokkal vegyesen is („mixed” attribútum)
  - üres vagy tetszőleges tartalom beágyazásával
  - más komplex típusok szűkítésével

Komplex típusok deklarációja – példa

```
<xsd:element name="UKPrice">
   <xsd:complexType>
        <xsd:simpleContent>
               <xsd:extension base="xsd:decimal">
                      <xsd:attribute name="currency"
                                     type="xsd:string"/>
               </xsd:extension>
        </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>
```

Típusok elnevezése és anonim típusok

- Az általunk deklarált típusokhoz neveket rendelhetünk.
- Az elemek a type attribútummal hivatkozhatnak a típusnevekre.
- Lehetőségünk van anonim típusok alkalmazására is (tipikusan az egyszer használatos deklarációk esetében)
  - ebben az esetben nincs type attribútum
  - a típusdeklaráció a felhasználási helyén található

```
<xsd:element name="UKPrice">
  <xsd:complexType>
        <xsd:simpleContent>
...
        </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

Típusdeklarációk módosítói (facets)

- xsd:whiteSpace
  - „üres” (tabulátor, soremelés, újsor) karakterek kezelése
  - többszörös és záró szóközök „lenyelése”
- tipikus értékkészlet módosítók:
  - `xsd:enumeration`
  - `xsd:length`
  - `xsd:minLenght`
  - `xsd:maxLength`
  - `xsd:minExclusive (Inclusive)`
  - `xsd:maxExclusive (Inclusive)`
  - `xsd:pattern`

  a módosítók alkalmazhatósága és hatása függ a módosított típustól is
- A módosítók nem változtathatják meg a típust annak eredeti deklarációjával ellentétesen (pl. nullánál kisebb pozitív szám).

Típusdeklarációk módosítói (facets) - példák

```
<xsd:simpleType>
   <xsd:restriction base = "xsd:string">
        <xsd:length value = "2" />
   </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType>
   <xsd:restriction base = "xsd:string">
        <xsd:pattern value="[A-Z]{2}" />
   </xsd:restriction>
</xsd:simpleType>
```

Elemek deklarációja

- Deklaráció

```
<xsd:element name="..." type="...">...</xsd:element>
```

- A Schema tartalom modellek
  - a deklarált típusokra támaszkodva hozzuk létre őket
  - a Schema nyelv a DTD-nél szabadabb módon teszi lehetővé az elemek csoportosítását, illetve ezen csoportok elnevezését
  - az elemek összekapcsolásának „operátorai” (csoportosítások)
    - `choice`
    - `sequence`
    - `all`
    - `group`
  - a tartalom modellben hivatkozhatunk máshol deklarált elemekre

```
<xsd:element ref="comment" />
```

Globális és lokális elemdeklarációk

- A globális deklarációk közvetlenül a gyökér (xsd:schema) elem alatt vannak.
  - nem hivatkozhatnak más globális elemekre (`ref` attribútum)
  - nem használhatják a `minOccurs` és `maxOccurs` módosítókat
  - a lokális deklarációkban hivatkozhatunk rájuk, és további szűkítést is alkalmazhatunk (pl. `minOccurs` és `maxOccurs`)
  - dokumentum példányok gyökér elemei lehetnek

- A nem globális deklarációk lokálisak.
  - más deklarációk belsejében helyezkednek el, érvényességük is arra terjed ki

Attribútumok deklarációja

- Deklarációjuk

```
<xsd:attribute name="..." type="...">...</xsd:attribute>
```

- Különböző módosítókat (Schema attribútumokat) alkalmazhatunk
  - `default`
  - `fixed`
  - `use = (optional | prohibited | required) : optional`
  - `...`
- Az attribútumok csoportokba szervezhetők
  - a dokumentumpéldányokban együtt használt attribútumok összefogása

```
<xsd:attributeGroup name="GarmentGroup">
  <xsd:attribute name="size" type="xsd:string"/>
  <xsd:attribute name="color" type="xsd:string"/>
  <xsd:attribute name="manufacturer" type="xsd:string"/>
</xsd:attributeGroup>
```

- Az elemekhez hasonlóan globálisan és lokálisan is deklarálhatjuk őket.

Dokumentálás (annotációk)

- Emberek és alkalmazások számára is készíthetjük őket.
- A deklarációk dokumentálása a deklaráció elején történik.

```
<xsd:complexType>
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
           empty anonymous type with 2 attributes
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
           …
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

- Az annotációkban célszerű rögzíteni a nyelvet (xml:lang attribútum)
- A jó dokumentáció nemcsak a leendő felhasználókat segíti, de a deklarációk későbbi módosítását (a felidézést) is.

Schema dokumentumok összekapcsolása

- Külső deklarációs dokumentum betöltése
  - ekvivalens a dokumentum tartalmának bemásolásával

    `<xsd:include schemaLocation="….xsd" />`
  - ugyanazt a cél névteret alkalmazzuk a deklarációkban

- Külső deklarációk betöltése és újradefiniálása
  - a bemásolás után lehetőségünk van a deklarációk módosítására

    `<xsd:redefine schemaLocation="….xsd">...</xsd:redefine>`
  - ebben az esetben is ugyanazt a cél névteret alkalmazzuk a deklarációkban

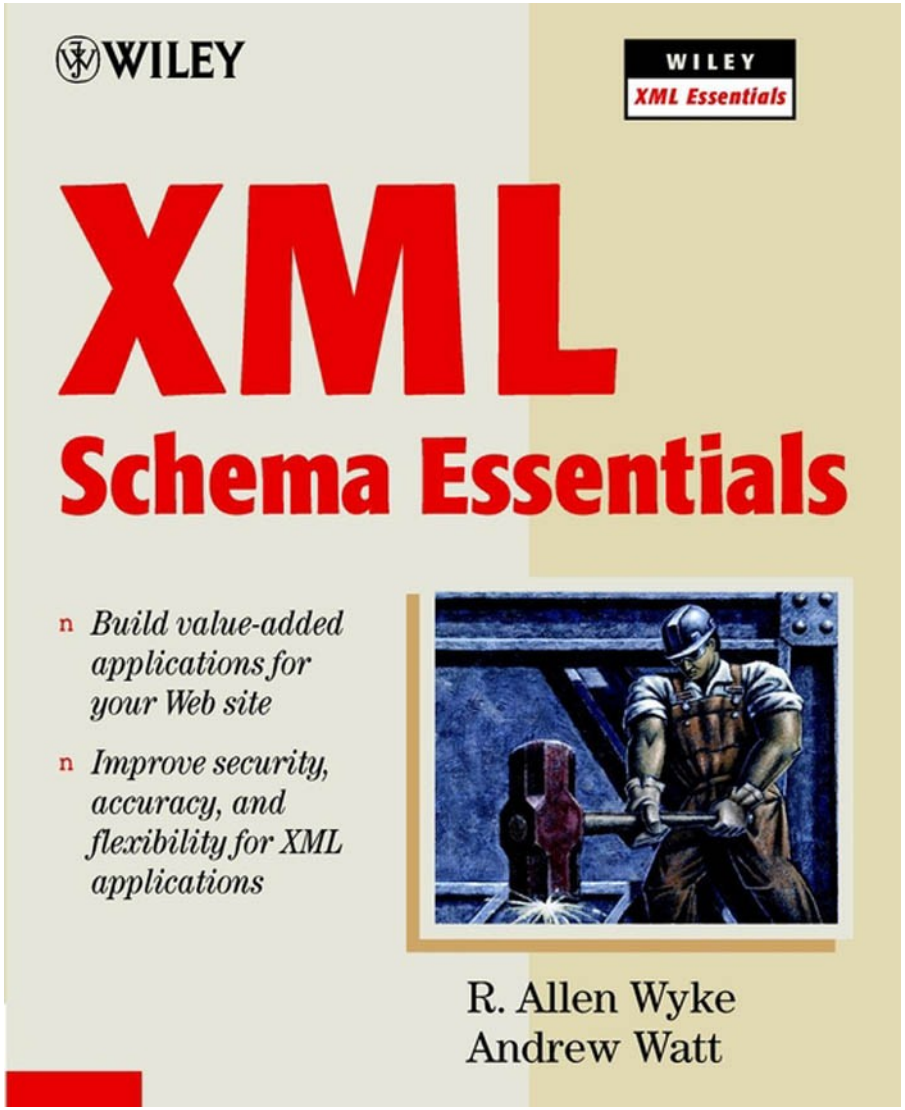
- Más névterek deklarációit is betölthetjük
  - ekvivalens a dokumentum tartalmának bemásolásával

    `<xsd:import nameSpace="..." schemaLocation="….xsd">`
  - az új névtér deklarációit felhasználhatjuk a dokumentumunkban
  - alapvetően típusok újrahasznosítására használjuk az import mechanizmust

A Schema deklaráció előnyei (összefoglalás)

- A programozási nyelvekben megszokott típusválasztékot kínálja (többet is)
  - számok, dátumok, URI, boolean, időintervallumok
  - értékkészlet minta szerinti rögzítése
- Komplex, felhasználói típusok is létrehozhatók
  - elemeket strukturálhatunk, tipizálhatunk
  - csoportosíthatjuk is őket az egyszerűbb felhasználás érdekében
  - speciális szövegstruktúrákat (listákat) is használhatunk
- Nyitott, finomítható dokumentum struktúrák
  - objektum-orientált dokumentum deklarációnak is nevezik
  - az egyszerű típusok esetében is (értékkészlet meghatározása)
  - szűkítés: a meglevő struktúra elemeinek elhagyása
  - kiterjesztés: új elemek hozzáfűzése egy meglevő struktúrához

Ajánlott irodalom



XML dokumentumok minősítése

- Jól formázott dokumentum (kötelező követelmény)
  - betartja a szabvány szintaktikai előírásait
  - eredménye, hogy könnyebb nyelvi elemzőt írni
- Érvényesség
  - a dokumentum megfelel a deklarációjának
  - enélkül is elemezni tudjuk a dokumentumot
  - hozzájárul a dokumentum tartalmi megértéséhez
  - a tárolt információ kinyeréséhez elengedhetetlen
- Schema érvényesség
  - a dokumentumnak van Schema deklarációja
  - tipizált deklaráció és tartalom
  - lehetővé válik a pontosabb deklaráció és feldolgozás a programozási nyelvek típuskészletét felhasználva