XML (eXtensible Markup Language)

Bevezetés

- Az XML-t adatok tárolására és továbbítására tervezték.
- Mind ember és mind gép számára olvasható formátumban tárolja az adatot.
- Az XML szoftver- és hardverfüggetlen eszköz.
- Önmagában nem csinál semmit, nincsenek előre definiált tagek.

Egy egyszerű XML példa:

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

Az XML-t sokszor használják arra, hogy az adatot elszeparálják a megjelenítéstől.

XML fa

- Az XML dokumentumok egy fa struktúrát alkotnak (XML DOM).
- Minden XML dokumentumban pontosan **egy** gyökér található.
- Minden pontnak lehetnek gyerekei is.
- A fenti példában `<note>` a gyökér, melynek négy gyereke van (to, from, heading és body)

Az XML-ek első sorában az úgynevezett prolog sor található, mely speciális formátumú és opcionális. A lenti példában a használt XML verzióját (kezdetektől fogva 1.0) és a karakterkódolást adja meg (az alapértelmezett az UTF-8).

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

A prolog sor nem része az xml dokumentumnak, ezért nincs szükség lezáró tagre.

Egy XML dokumentum **jól formázott**, ha teljesülnek a következők:

- Nincs átlapoló tag-pár (pl: `<to> <from> </to> </from>`)
- Minden tagnek van lezáró párja
- Minden attribútum érték idézőjelek vagy aposztrófok között szerepel
- Pontosan egy gyökér elem található a dokumentumban

Karakter entitások

Mivel az XML-ben vannak speciális karakterek, ezeket valahogy máshogy kell írnunk ha egy ilyen karaktert szeretnénk adatként tárolni. Ezek a **karakter entitások**:

Karakter entitás	Megjelenés	Elnevezés
&lt;	<	kisebb mint
&gt;	>	nagyobb mint
&amp;	&	és jel
&apos;	'	aposztróf
&quot;	"	idézőjel

Például azt szeretnénk írni a `<heading>` tag-ek közé, hogy x < y, akkor azt a következőképpen tehetjük meg:

```
<heading> x &lt; y </heading>
```

Elemek

Minden XML elem a következőeket tartalmazhatja:

- szöveg (pl: `<to>` **szöveg** `</to>`)
- attribútum (pl: `<note id="attr"> </note>`)
- más elemek (pl: a note tartalmazza a to, from, heading, body tageket)
- a fentiek kombinációja

Az XML tagek lehetnek üresek, azaz nincs bennük se szöveg, se másik elem:

```
<heading></heading>
```

Ezt rövidebben is írhatjuk (**self-closing tag**):

```
<heading />
```

GYEREK ELEM VAGY ATTRIBÚTUM?

```
<note>
    <to>Jozsi</to>
</note>
```

A fenti XML részlet ekvivalens a következő kóddal:

```
<note to="Jozsi" />
```

Látható, hogy teljesen mindegy, melyik módot választjuk, azt adatot tárolni tudjuk valamilyen formában. Akkor mégis mikor melyiket használjuk? **Alapgondolat**: Az elemre vonatkozó meta-adatokat tároljuk attribútumokban a többit pedig gyermek elemként hozzuk létre. Tehát a to legyen inkább gyermek elem. Ha a note-ra vonatkozó meta-adatunk van (pl: id) akkor azt tároljuk attribútumként.

Névterek (namespaces)

Az XML névterek alapvető funkciója, hogy a névütközéseket elkerüljük. Ez akkor fordulhat elő, ha különböző XML-eket (különböző domain) akarunk egyesíteni. Például egy HTML table-t használó részlet:

```
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

Egy másik dokumentumban viszont a table egy bútordarabra vonatkozik:

```
<table>
    <name>African Coffee Table</name>
    <width>80</width>
    <length>120</length>
</table>
```

Ha ezt a két dokumentumot összefésülnénk, akkor névütközés lépne fel, mivel mindkettő dokumentumban van table, azonban teljesen más a jelentésük (ráadásul más elemeket is tartalmaznak).

```
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

A fenti példa még nem teljes, mivel névtereket már használunk benne, de még nem definiáltuk a "h" és "f" névtereket. A névtereket bármelyik elem xmlns attribútumában definiálhatjuk. A névterek definiálásának szintaxisa:

```
xmlns:prefix="URI"
```

A fenti példa helyesen a névtér definíciókkal kiegészítve így néz ki:

```
<root>
    <h:table xmlns:h="http://www.w3.org/TR/html4/">
        <h:tr>
            <h:td>Apples</h:td>
            <h:td>Bananas</h:td>
        </h:tr>
    </h:table>

    <f:table xmlns:f="http://www.w3schools.com/furniture">
        <f:name>African Coffee Table</f:name>
        <f:width>80</f:width>
        <f:length>120</f:length>
    </f:table>
</root>
```

A névtér definícióját az összes gyermek elem örökli. A névtereket a root elemnél is definiálhatjuk:

```
<root
    xmlns:h="http://www.w3.org/TR/html4/"
    xmlns:f="http://www.w3schools.com/furniture">
```

A névtereknél a az URI-nak nem kell létezőnek lennie, csak az egyediséget hivatott ellátni. A vállaltok azonban arra szokták használni, hogy a névteret leíró oldalra mutassanak. Megadhatunk alapértelmezett névteret (default namespace) is, melynek formája:

```
xmlns="URI"
```

Ilyenkor az összes gyerek elem ugyanabba a namespace-be fog tartozni.

XSD (XML Schema Definition)

Az előzőekben bármit írhattunk egy-egy XML állományba, nem volt semmilyen megkötés az adatokra vonatkozóan. Az XSD segítségével az XML fájlok tartalmát **validálhatjuk**. Az XSD leírja az XML fájlok struktúráját. Maga az XSD is egy XML fájlban található, de .xsd kiterjesztést használ.

Alapok

Az XML gyökér elemben hivatkozunk az XSD-re:

```
<catalog name="My cds to sell" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation ="cd_catalog.xsd">
...
</catalog>
```

Az XSD fájl gyökéreleme mindig a schema elem:

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

Validálás a gyakorlatban

Adott az XSD állomány és adott egy XML állomány. Arra vagyunk kíváncsiak, hogy az XML állomány megfelel-e az XSD állományban leírtaknak. Ahhoz, hogy erre megkapjuk a választ szükségünk van valamilyen programra.

A gyakorlaton az `xmllint` -et fogjuk használni, ami Linux alatti program.

Telepítése:

```
sudo apt install libxml2-utils
```

Használat:

```
xmllint --noout --schema XSD_SCHEMA_FILE XML_FILE
```

További online eszközök

- http://www.utilities-online.info/xsdvalidation/#.XEh51lz0mUk
- https://www.freeformatter.com/xml-validator-xsd.html
- https://www.liquid-technologies.com/online-xsd-validator
- https://www.corefiling.com/opensource/schemaValidate/

Elemek

Egy elemet a következőképpen definiálhatunk (szabály egy XML elemre):

```
<xs:element name="xxx" type="yyy" />
```

Ahol az xxx a xml elem neve lesz.

Az yyy típus lehet:

- **egyszerű** típus (nem tartalmaz további elemeket)
- **összetett** típus (tartalmaz további elemeket is)

A note példához visszatérve a to, from, heading és body elemeket a következőképpen definiálhatjuk:

```
<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"/>
```

Az egyszerű és összetett típusok alapján szinonímaként használjuk az egyszerű/összetett elem elnevezéseket. Leggyakoribb egyszerű (beépített) típusok:

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

Az elemekhez megadható két módosító attribútum is. Az elemeknek lehet alapértelmezett értéke vagy fix értéke:

```
<xs:element name="color" type="xs:string" default="red"/>
<xs:element name="color" type="xs:string" fixed="red"/>
```

Az 1. sorban található definíció eredményeképpen, ha a color tag-ek közé nem rakunk majd értéket akkor annak alapértelmezett értéke legyen red. A 2. sorban található definíció azt mondja, hogy minden esetben legyen red az értéke, és más érték nem adható meg.

Attribútumok

Csak az összetett típusú elemeknek lehetnek attrbútumaik. Minden attribútumnak egyszerű típusúnak kell lennie. Az attrbútumok megadási módja:

```
<xs:attribute name="xxx" type="yyy"/>
```

xxx az attribútum neve, a típusa pedig a fent felsorolt egyszerű (beépített) típusok egyike. Egy-egy attribútumnak, hasonlóan az elemeknél látott default és fixed attribútumoknál, lehet default és fixed módosítója.

```
<xs:attribute name="lang" type="xs:string" default="red"/>
<xs:attribute name="lang" type="xs:string" fixed="red"/>
```

Az attribútumok **alapértelmezetten opcionálisan**, ahhoz hogy kötelezőek legyenek használjuk a use="required" attribútumot:

```
<xs:attribute name="lang" type="xs:string" use="required"/>
```

Megszorítások

A megszorítások segítségével az elfogadható értékeket korlátozhatjuk (elemekre és attribútumokra is megy). Példák: (az age elem tartalma 0 és 120 közé kell hogy essen):

```
<xs:element name="age">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="120"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Enum megszorítás (a car tartalma csak Audi, Golf és BMW lehet):

```
<xs:element name="car">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="Audi"/>
            <xs:enumeration value="Golf"/>
            <xs:enumeration value="BMW"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

A típusokat külön(az elemen kívül és nem beágyazva) is definiálhatjuk így több helyen is felhasználhatjuk őket:

```
<xs:element name="car" type="carType"/>
```

```
<xs:simpleType name="carType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Audi"/>
    <xs:enumeration value="Golf"/>
    <xs:enumeration value="BMW"/>
  </xs:restriction>
</xs:simpleType>
```

Pattern: Regex-el adhatjuk meg, hogy milyen formátumú adatot fogadunk el. A következő példa letter-nek csak egy angol kisbetűt fogad el:

```
<xs:element name="letter">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-z]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

További patternek:

```
<xs:pattern value="[A-Z][A-Z][A-Z]"/>
<xs:pattern value="[xyz]"/>
<xs:pattern value="[0-9][0-9][0-9][0-9][0-9]"/>
<xs:pattern value="([a-z])*"/>
<xs:pattern value="([a-z][A-Z])+"/>
<xs:pattern value="male|female"/>
<xs:pattern value="[a-zA-Z0-9]{8}"/>
```

String hosszának megszorítása:

```
<xs:length value="8"/>
<xs:minLength value="5"/>
<xs:maxLength value="8"/>
```

Összetett elemek

Az összetett elemek tartalmazhatnak további elemeket és/vagy attribútumokat. Négyféle egyszerű elem létezik:

- Üres elemek
- További elemeket tartalmaz
- Szöveget tartalmaz
- Szöveget és további elemeket is tartalmaz

Példák: Egy üres összetett elem (összetett, mert van attribútuma)

```
<product pid="1345"/>
```

További elemeket tartalmazó elem:

```
<employee>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</employee>
```

Csak szöveget tartalmazó elem (összetett, mert van attrbútuma)

```
<food type="dessert">Ice cream</food>
```

Szöveget és további elemet is tartalmazó összetett elem:

```
<description>
    It happened on <date lang="norwegian">03.03.99</date> ....
</description>
```

Összetett elemek definiálása

A simpleType helyett complexType használatos. A fenti employee xml-t leíró XSD a következő:

```
<xs:element name="employee">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

A **sequence** egy indikátor, mely megadja, hogy az elemeknek olyan sorrendben kell szerepelnie az XML-ben, mint ahogy az xsd deklarációban szerepelnek. Az összetett elemek típusa szintén megadható a type attribútumban, így újrafelhasználható lesz a típus:

```
<xs:element name="employee" type="personinfo"/>

<xs:complexType name="personinfo">
```

```
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Az összetett elemek típusai (összetett típusok) **kiterjeszthetnek** más összetett típusokat. Ehhez az extension szükséges, melyben megadjuk az alaptípust, ezután az új elemeket adjuk meg.

```
<xs:element name="employee" type="fullpersoninfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="fullpersoninfo">
  <xs:complexContent>
    <xs:extension base="personinfo">
      <xs:sequence>
        <xs:element name="address" type="xs:string"/>
        <xs:element name="city" type="xs:string"/>
        <xs:element name="country" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Üres Elemek

Olyan összetett elem aminek, csak attribútumai lehetnek. Példa:

```
<product prodid="1345" />
```

A hozzá tartozó XSD:

```
<xs:element name="product">
  <xs:complexType>
    <xs:attribute name="prodid" type="xs:positiveInteger"/>
  </xs:complexType>
</xs:element>
```

Indikátorok

Az indikátorok fajtái:

- Sorrendezés:
  - **all**: minden gyerek elem egyszer (a minOccurs attribútummal 0-ra vagy 1-re állíthatjuk az előfordulást) fordul elő, a sorrend nem számít.
  - **choice**: A megadott elemek közül csak az egyik fodulhat elő.
  - **sequence**: Minden elemnek elő kell fordulnia a megadott sorrendben.
- Előfordulás:
  - **maxOccurs**: Az elem maximális előfordulási száma. Ha végtelenszer is előfordulhat egy elem, akkor használjuk: maxOccurs="unbounded"
  - **minOccurs**: Az elem minimális előfordulási száma
- Csoportok: összetartozó elemek/attribútumok definiálása. A csoporton belül egy sorrendezésnek kell szerepelnie: all, choice vagy sequence.
  - **group**: Összetartozó elemek definiálása
  - **attributeGroup**: : Összetartozó attribútumok definiálása

Példák:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">

<xs:element name="persons">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="person" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="full_name" type="xs:string"/>
            <xs:element name="child_name" type="xs:string"
            minOccurs="0" maxOccurs="5"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

A fenti példában a persons elemen belül akárhány person elem szerepelhet. Minden person elem tartalmaz egy full nevet és 0, de legfeljebb 5 child_name elemet.

```
<xs:group name="persongroup">
  <xs:sequence>
```

```
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
    <xs:element name="birthday" type="xs:date"/>
  </xs:sequence>
</xs:group>

<xs:element name="person" type="personinfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:group ref="persongroup"/>
    <xs:element name="country" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

A group elem ref attribútumával meghivatkozhatjuk a korábban definiált group-ot. **Hasonlítsuk össze a típusok kiszervezését és a group-ok ref-es megadását. Vegyük észre a különbséget!**