Az XML (1. rész)

Gondolatok a hordozható adatokról, azaz hogyan készíthetünk mindenki által könnyen használható dokumentumokat.

Az XML a World Wide Web Konzorcium (W3C) ajánlása, mely egy sokkal régebbi SGML (Standard Generalized Markup Language = szabványos általánosított jelölőnyelv) elnevezésű ISO 8879/1986 szabványnak felel meg. Az SGML-megfelelőség azt jelenti, hogy minden XML-dokumentum egyben SGML-dokumentum is, de fordítva már nem igaz, azaz van olyan SGML-leírás, ami nem XML-megfelelő.

Az XML (Extensible Markup Language = bővíthető jelölőnyelv) adatleíró nyelv létrehozását a nyílt rendszerek térhódítása és az Internet tette szükségessé. A Java nyelv és az XML között érdekes hasonlóság figyelhető meg: a Java a futtatható formában hordozható programok nyelve, míg az XML a hordozható adatok készítésének az eszköze.

Az XML-dokumentumok (hasonlóan a Java-forráskódhoz) unicode-alapú szöveges karaktersorozatok, ahol az UTF-8 (tömörített unicode) az alapértelmezés, de a dokumentum elején ettől eltérő kódolási eljárást is választhatunk.

Az XML formátumú dokumentumok fokozatosan kezdik leváltani a jelenleg széles körben elterjedt ASCII-alapú szövegfájl-alkalmazásokat (például a beállítófájlokat). Ennek oka, hogy az ASCII-fájlok semmilyen leíró- és kezelőinformációt (metaadatot) nem tartalmaznak saját magukról (még az az alapvető tulajdonságuk sem őrződik meg, hogy a fájl készítője melyik kódlapot használta). Ezzel szemben az XML-alapú dokumentumok a benne tárolt adatok vagy hivatkozások (szöveg, kép stb.) értékein felül pluszinformációkkal is rendelkeznek, ugyanakkor leírásuk – hasonlóan a HTML nyelvhez – továbbra is rendelkezik azzal a kellemes tulajdonsággal, hogy szövegalapú.

Az XML-dokumentumokban az adatok értékein túl további címkéket és hivatkozásokat helyezhetünk el, amelyek utalnak az adat természetére, a dokumentum szerkezeti és tartalmi felépítésére, és a dokumentum érvényességének vizsgálatához is felhasználhatók.

Egy XML-dokumentum nem tartalmaz utalást arra nézve, hogy egy alkalmazás (például Internet Explorer 5, a továbbiakban IE5) hogyan jelenítse meg, de vannak kiegészítő W3C-ajánlások, amelyek pótolják ezt a hiányosságot.

Az XML-t úgy tervezték, hogy formális nyelvtanát könnyű legyen meghatározni, így az XML-adatfolyam elemzése egyszerű. Ez utóbbi tulajdonság az, ami miatt az XML ma szinte az egyetlen széles körben elterjedt, általános eszköz adataink hordozható formában történő tárolására és továbbítására.

Az XML ebben a megközelítésben a különféle jelölőnyelvek készítését leíró nyelv (metanyelv). XML-alkalmazásnak nevezzük, amikor XML segítségével készítünk el egy új jelölőnyelvet.

Ebben a cikksorozatban az XML összes lényeges vonatkozására kitérünk, melyek következő összetevőket jelentik:

- A jólformáltság feltételei.
- Az érvényességvizsgálat eszközei (DTD, Schema),
- A dokumentum elemzése és feldolgozása (Parse),
- A dokumentum megjelenésének meghatározása (CSS – stíluslap, XSL – bővített stíluslapleíró nyelv),

Első XML-dokumentumunk létrehozása

Annak érdekében, hogy az eddig leírtak ne hangozzanak túlságosan elvontnak, nézzünk egy példát! Legyen egy `WEBADATOK` adatbázisunk, amiben a `VKONYV` vendégkönyvtábla felépítése a következő:

```
NEV     VARCHAR(50)
EMAIL   VARCHAR(50)
DATUM   DATE
SZOVEG  VARCHAR(500)
```

Vizsgálódásunk pillanatában a `VKONYV` tábla a következő két bejegyzést tartalmazta:

Nyiri Imre	inyiri@mol.hu	2001.01.31	Üdvözletem a webmesternek!
Koller József	jkoller@mailbox.hu	2001.04.30	Tetszett a site :)

Az XML jellegzetessége, hogy az adatokat függőségi viszonyukban képes megjeleníteni, így egy XML-adatfolyam egyben mindig egy fát is meghatároz. A `VKONYV` tábla tartalmát a következő XML-formában állíthatjuk elő:

```
<?xml version="1.0" encoding="WINDOWS-1250" ?>
<VKONYV>
  <VENDEG sorszam="1">
     <NEV>Nyiri Imre</NEV>
     <EMAIL>inyiri@mol.hu</EMAIL>
     <DATUM>2001.01.31</DATUM>
     <SZOVEG> dv zletem a webmesternek!</SZOVEG>
  </VENDEG>
  <VENDEG sorszam="2">
     <NEV>Koller J zsef</NEV>
     <EMAIL>jkoller@mailbox.hu</EMAIL>
     <DATUM>2001.04.30</DATUM>
     <SZOVEG>Tetszett a site :) </SZOVEG>
  </VENDEG>
</VKONYV>
```

A fenti XML-adatfolyamot vizsgálva láthatjuk, hogy eddig két vendég írt a vendégkönyvbe. Itt az ideje, hogy elgondolkodjunk azon, miért éppen ebben a formában hoztuk létre a VKONYV tábla tartalmát, hiszen megtehettük volna például így is:

```
<?xml version="1.0" encoding="WINDOWS-1250" ?>
<VKONYV>
  <VENDEG sorszam="1" NEV="Nyiri Imre"
     ➥DATUM="2001.01.31">
     <EMAIL>inyiri@mol.hu</EMAIL>
     <SZOVEG> dv zletem a WEB
     ➥mesternek!</SZOVEG>
```

Próbaűrlap

```
<!DOCTYPE UI><UI>
<class>frmTeszt</class>
<widget>
    <class>QWidget</class>
    <property stdset="1">
        <name>name</name>
        <cstring>frmTeszt</cstring>
    </property>
    <property stdset="1">
        <name>geometry</name>
        <rect>
            <x>0</x>
            <y>0</y>
            <width>342</width>
            <height>155</height>
        </rect>
    </property>
    <property stdset="1">
        <name>caption</name>
        <string>Teszt Form</string>
    </property>
    <widget>
        <class>QPushButton</class>
        <property stdset="1">
            <name>name</name>
            <cstring>btnOK</cstring>
        </property>
        <property stdset="1">
            <name>geometry</name>
            <rect>
                <x>20</x>
                <y>20</y>
                <width>104</width>
                <height>28</height>
            </rect>
        </property>
        <property stdset="1">
            <name>text</name>
            <string>OK</string>
        </property>
    </widget>
</widget>
</UI>
```

```
  </VENDEG>
  <VENDEG sorszam="2" NEV="Koller J zsef"
      ➥DATUM="2001.04.30">
    <EMAIL>jkoller@mailbox.hu</EMAIL>
    <SZOVEG>Tetszett a site :) </SZOVEG>
  </VENDEG>
</VKONYV>
```

E többféle lehetőség annak a következménye, hogy az XML-ajánlás az adatok értelmezését (szemantikáját) és a használható megjelölő jelölőtagok (címkék: VKONYV, NEV stb.) körét nem határozza meg, így annak kialakítása tervezési megfontolásokon alapul. A második megoldás jobban épít a címkék tulajdonságmegadási lehetőségeire, melynek következtében a leíró címkék száma csökkent. A fentiek alapján belátható, hogy a VKONYV tábla tartalmát sokféle XML-adatfolyamként elő lehet állítani. A tervezés során ki kell választani azt az XML-formátumot, amelyet az adott feladat szempontjából jónak tartunk, majd el kell készíteni hozzá azokat a nyelvtani szabályokat, melyek azt egyértelműen rögzítik (például hogy a továbbiakban mi a VKONYV első változatú XML-adatfolyamát szeretnénk használni). A nyelvtani szabályok megfogalmazására a DTD (Document Type Definition = dokumentumtípus-meghatározás) nyelv szolgál, amit a későbbiekben részletesen is leírunk. Példánkból arról is meggyőződhettünk, hogy az XML-leírás valóban nem tartalmaz olyan nyelvi elemet, ami a dokumentum kinézetét határozná meg, hiszen csak a dokumentum szerkezetét, értelmezését és adatait tartalmazta.
Az első XML-dokumentumot egy vkonyv.xml fájlban tárolva az IE5 alapértelmezettként faformában jeleníti meg, ahol a +/- jelekkel lehetséges a fa ágait kinyitni vagy becsukni. (Ez Netscape 6 alatt is hasonlóan működik, csak ott nincs lehetőség az ágak kibontására, illetve bezárására, a fa teljesen nyitott állapotban jelenik meg.) A fájl első sorában a „version=1.0” megadása kötelező, a kódolásé viszont választható, de ha nem adjuk meg, akkor az alapértelmezett érték UTF-8 lesz.

XML-alkalmazások

A fentiekben már említettük, hogy az XML segítségével megalkotott jelölőnyelveket XML-alkalmazásoknak nevezzük.
Mielőtt részletesen leírnánk az XML-adatfolyammal kapcsolatos alapvető tudnivalókat, tekintsünk ki egy kicsit a világba, és nézzünk néhány példát arra vonatkozóan, mi is készül napjainkban az XML segítségével. Alaposan körbenézve az egyes programok között, látható, hogy az XML szinte már alapvető beállító- és adattároló eszközzé vált.
A Linux GNOME gnumeric és az AbiWord dokumentumtárolási formájául az egyaránt XML-t választotta. AbiWordöt használva mentsünk egy dokumentumot a „Helló Világ” szöveggel. Nézzük meg egy szövegszerkesztővel a kapott fájlt (a megjegyzéseket kivettem belőle), értelmezése már nem okozhat sok nehézséget:

```
<?xml version="1.0"?>
<abiword version="0.7.11">
<section>
<p><c props="font-size:14pt">Hell&#xf3;
      ➥Vil&#xe1;g!</c></p>
</section>
</abiword>
```

A Linux alatti egyik legjobb grafikus elemkészlet a QT, amihez létezik egy QT Designer (a QT-elemkészlet és a QT Designer a Trolltech alkotása). A QT Designerben – a Delphihez vagy a Visual Basichez hasonlóan – grafikusan lehet felhasználói felületeket készíteni, majd azokat szabványos *.ui fájlokba menteni. Amikor a QT Designer használatával egy „Teszt Form” címsorú és „frmTeszt” nevű űrlapba „OK” feliratú, „btnOK” nevű gombot tervezünk, a *listán* látható XML-fájl jön létre.
A fenti két példán kívül most csak megemlítek néhány további, elterjedt XML-alkalmazást:

- Az XHTML nyelv: a HTML nyelvet írja le, illetve kissé módosítja annak érdekében, hogy a HTML XML-megfelelő legyen.
- Az XSQL nyelv: adatbáziskezelő műveletek XML-megfelelő leírására szolgáló jelölőnyelv.
- MathML (Mathematics Markup Language).

A szabványosított XML-alkalmazások köre egyre szélesebbé válik, amit a W3C honlapján folyamatosan figyelemmel követhetünk.
Az egyik legfontosabb XML-alkalmazás az XSL (Extended Style

Language = bővített stíluslapleíró nyelv), amivel – a CSS-t helyettesítve – az XML-dokumentumok képi megjelenítésének kinézetét határozhatjuk meg.
Az XML adatleíró képességének általános volta miatt egyre gyakrabban találkozunk XML-alapú beállítófájlokkal is. Mindenkinek ajánlom tanulmányozásra a Jakarta-Tomcat web.xml nevű fájlját, ami tartalmazza a webhely aktuális kialakításának beállítását.
Ezen a ponton abbahagyjuk az XML-megfelelő jelölőnyelvek felsorolását, hiszen végtelen sok ilyen létezik, illetve készíthető. Befejezésként érdemes kiemelni, hogy a Microsoft Office termékek következő változatai lehetővé fogják tenni a dokumentumok XML-megfelelő formátumban való tárolását. A szakemberek egyre inkább úgy tartják, hogy az Interneten a HTML nyelvet a közeljövőben az XML-megfelelő jelölőnyelvek fogják felváltani, melynek egyik előhírnöke az XHTML.

Az XML-adatfolyam felépítése (XML-fájlok)

Az XML-adatfolyamot formai és nyelvtani szempontból is vizsgálhatjuk. Ebben a pontban a formai elemzés szempontjait tekintjük át. Egy XML-dokumentumot jólformázottnak (well formed) nevezünk, ha a következőkben részletezett formai feltételeknek eleget tesz.

Elemek (Elements)

Az elemek az XML-dokumentumok legalapvetőbb részei. Ilyen elem például a `<NEV>Koller J zsef</NEV>` részlet. Egy XML-elem általában a következő részeket tartalmazza:

- az elemet bevezető `<ElemCimke>` tagot, amit kisebb-nagyobb jelek között kell megadni;
- az elemhez tartozó adatot, például: Koller József (nem minden elem tartalmaz adatot);
- az elemet záró `</ElemCimke>` tagot.

A nyitó- és zárócímkék feladata, hogy megjelöljék és elnevezzék az általuk közrefogott adatot. Ezzel adataink természete és tartalma is nyomon követhetővé válik, így fontos, hogy a címkenevek kifejezők legyenek. Lényeges kiemelnünk, hogy a hagyományos XML-meghatározás nem ad olyan pontos adatábrázoló- és kezelőeszközt a kezünkbe, mint a programozásban megismert típus fogalma, azonban már létezik olyan W3C-ajánlás (XML Schema meghatározás), ami ennek a szigorú követelménynek is eleget tesz.
Időnként előfordul, hogy egy címke nem fog közre semmilyen adatot. Erre példa az, ha valakinek nincs levélcíme: `<EMAIL></EMAIL>`. Ezt a terjengős leírást az XML-ben a következő módon rövidíthetjük: `<EMAIL/>`. A címkenevekre vonatkozóan a következő szabályokat kell betartani:

- Csak betűvel vagy aláhúzásjellel kezdődhet.
- Tartalmazhat betűt, számjegyet, pontot, kötőjelet és aláhúzásjelet.
- A kis- és nagybetűk különbözőnek számítanak.

Megjegyzés

XML-dokumentumban a HTML nyelvben megszokott megjegyzést használhatjuk.

```
<!-
Ez egy XML megjegyzØs, ami formailag olyan, mint
a HTML megjegyzØs
-->
```

Ezeket a részeket az XML-feldolgozó (elemző, érvényességvizsgáló, megjelenítő) alkalmazások figyelmen kívül hagyják.

Tulajdonságok (Attributes)

Az XML-elemek tetszőleges számú tulajdonsággal rendelkezhetnek. A VKONYV XML második változatából tekintsük a következő részletet:

```
<VENDEG sorszam="2" NEV="Koller J zsef"
        ➥DATUM="2001.04.30">
```

Itt három tulajdonságot adtunk meg a `VENDEG` elem részére: sorszam, `NEV, DATUM`.
A tulajdonságok általában így néznek ki:

```
<elem nev1="ØrtØk1" nev2="ØrtØk2" ...>
...
</elem>
```

Egyedhivatkozások (entity references)

Az egyedek olyanok, mint a makróhelyettesítések. Minden egyednek egyedi névvel kell rendelkeznie. Legyen az `EgyedNev` egy egyednév, melynek meghívása a következő: `&EgyedNev;`, azaz `&` jellel kezdődik és `;`-vel fejeződik be. Ha megnézzük az AbiWord által készített XML-dokumentum `&#xf3;` részletét, ott egy, az „ó” betűre vonatkozó egyedhivatkozással van dolgunk.
Léteznek előre meghatározott egyedek, mint például a `gt` nevű. Az `&gt;` hatása olyan, mintha az XML-adatfolyamba egyből a `>` jelet írtuk volna. Az egyedek fogalma azt is lehetővé teszi, hogy bármilyen unicode karaktert helyezzünk a szövegbe. Ekkor az egyednév `#szÆm` vagy `#xszÆm` alakú, ahol az első a decimális, a második a hexadecimális megadási mód. A kis „ő” hexaunicode kódja 0x0151. Ezt mint egyedet a következő módon hívhatjuk meg: `&#x0151;`. Az XML-ajánlás lehetővé teszi, hogy saját egyedeket is létrehozzunk. Ez az `<!ENTITY myEgyed " rtØk">` szerkezettel lehetséges, amit `&myEgyed;`-vel hívhatunk meg és ugyanaz a hatása, mintha közvetlenül az `rtØk`-et írtuk volna le. Az egyedek meghatározása a jelölőnyelv nyelvtani szabályait meghatározó DTD-leírás része, ezért erre ott még visszatérünk.

Feldolgozási utasítások (processing instructions)

A feldolgozási utasítások nem részei az XML-nek. A megjegyzésekhez hasonlóan nem részei a dokumentum szöveges tartalmának, de az XML-feldolgozókkal szemben követelmény, hogy ezeket az utasításokat továbbadják az alkalmazások számára.
Megadási módjuk: `<?target name?>`. Egy alkalmazás csak azokat az utasításokat veszi figyelembe, amelyek célját felismeri, az összes többit figyelmen kívül hagyja. A célt (target) követő adatok az alkalmazásnak szólnak, megadásuk nem kötelező.

A CDATA-szakaszok

Az ebben a szakaszban lévő CDATA, azaz karakteres adat értelmezés nélkül átadódik az XML-adatfolyamot fogadó alkalmazásnak. A szakasz írásmódja a következő:
`<![CDATA[Itt van a tetszıleges sz veg]]>`. A megadott szöveg a `]]` kivételével bármi lehet.

Az XML-adatfolyam érvényességének ellenőrzése

Az előző részben azt vizsgáltuk, hogy egy XML-dokumentum mikor jólformázott. A helyes formát nagyon könnyen ellenőrizhetjük különféle XML-feldolgozó programokkal, például az IE5 böngészővel.
A vkonyv.xml jólformázott, mert az IE5 elemezni tudta, és a faszerkezetet is megjelenítette.
A jólformázottság azonban még nem jelenti azt, hogy az XML-dokumentum helyes, hiszen ha mi a VKONYV adathalmazt az első formátumú XML-felépítésben szeretnénk látni, akkor például a második formátum nem érvényes, ezért az elemzőprogramok vissza fogják utasítani.

A mai XML-elemzők a jólformázottság mellett általában az érvényesség vizsgálatát is lehetővé teszik. Az XML egyik legnagyobb erőssége, hogy saját címkeneveket használhatunk, de a VKONYV példája arra is rámutat, hogy szükség ugyancsak van egy nyelvtani szabályokat meghatározó leírásra, amit az XML-elemző program az érvényesség ellenőrzéséhez feltud használni.
Az XML-elemek egy faszerkezetet építenek fel, amelynek természetesen mindig van egy gyökéreleme. A VKONYV XML karaktersorozatban a VKONYV elem a gyökér. Ez az a pont, ahonnan az XML-dokumentum nyelvtani elemzése indul (a formális nyelvtanok START-jának felel meg). A DTD nyelven megfogalmazott nyelvtani szabályok egy karaktersorozatot alkotnak, tárolásukra két lehetőség adódik:

- egy fájlban tároljuk őket (például vkonyv.dtd);
- az XML-dokumentum elején tároljuk őket.

Tegyük fel, hogy valaki elkészítette nekünk (hiszen mi még nem ismerjük az elkészítés módját) a VKONYV XML adatfolyam első változatának mint XML-dokumentumtípusnak DTD-ben megfogalmazott nyelvtanát, és ezt a vkonyv.dtd fájlban átadta nekünk. Ekkor XML-dokumentumunk első sora után a következőt lehet beszúrni:

```
<?xml version="1.0" encoding="WINDOWS-1250" ?>
<!DOCTYPE VKONYV SYSTEM "vkonyv.dtd">
<VKONYV>
  <VENDEG sorszam="1">
...
</VKONYV>
```

A beszúrt sor a `<DOCTYPE` résszel kezdődik, a következő szó pedig az XML-dokumentum gyökéreleme. A harmadik szó `SYSTEM` vagy `PUBLIC` lehet. A sor utolsó része azt adja meg, hogy melyik fájlban tárolódnak a nyelvtani szabályok.
Amennyiben a nyelvtant nem akarjuk külön fájlban tárolni (például azért, mert nagyon egyszerű), akkor az eredeti XML-adatfolyam elejét így kell kiegészíteni:

```
<?xml version="1.0" encoding="WINDOWS-1250" ?>
<!DOCTYPE VKONYV [Ide j n az a karaktersorozat,
    ➥amit egyØbkØnt a vkonyv.dtd-ben tÆrolnÆnk]>
<VKONYV>
  <VENDEG sorszam="1">
...
</VKONYV>
```

A beágyazott DTD-szabályokat tehát a fent látható módon `[ … ]` zárójelek között adhatjuk meg.

Ennyi előzetes után tekintsük át röviden, hogyan lehet a DTD szabályrendszereket megfogalmazni.

Elemtípus-bevezetés (Element Type Declaration)

Az elemtípus-bevezetések azonosítják az elemek neveit és tartalmát. A DTD-ben ez az azonosítás nem éri el a programozási nyelvek típusfogalmának megfelelő részletezettségi szintet. A szemléletesség kedvéért nézzük meg a VKONYV XML adatfolyam első változata `<VKONYV>` elemének típusbevezetését:

```
<!ELEMENT VKONYV (VENDEG)* >
```

A fenti sor szavakban kifejtve: a `VKONYV` elem `VENDEG` típusú elemek sorozata, ahol a `*` jelöli, hogy ennek a sorozatnak 0 vagy több tagja lehet (a mi vkonyv.xml fájlunkban a sorozatnak most két tagja van).
A számosság kifejezésére a `*`-on kívül még a `+` (1 vagy több elem) és a `?` (0 vagy 1 elem) jelek szerepelhetnek. Amikor egy név mellett nincsenek írásjelek, akkor pontosan egyszer kell előfordulnia. Most nézzük meg, hogy a `<VENDEG>` elemre milyen nyelvtani szabályok állapíthatóak meg!

```
<!ELEMENT VENDEG (NEV, EMAIL?, DATUM, SZOVEG)>
```

Ez azt jelenti, hogy egy `VENDEG` elem a `NEV`, `EMAIL`, `DATUM`, `SZOVEG` elemek ebben a – és nem más – sorrendben vett szerkezetéből áll. Észrevehetünk itt egy fontos dolgot: ez a nyelvtani szabály már kizárja, hogy a második típusú VKONYV XML-folyam érvényes legyen, hiszen ott a `VENDEG` elem csak az `EMAIL` és a `SZOVEG` elemeket tartalmazhatja.
Az elemneveken felül egy különleges szimbólumot tartottak fenn a karakteres adat jelölésére, a `#PCDATA`-t. Ez teszi lehetővé, hogy például a `NEV` elem nyelvtani szabályát megadjuk:

```
<!ELEMENT NEV (#PCDATA)>
```

azaz a `NEV` elem egy elemzett karakterfüzér.

Már láttuk, hogy léteznek üres elemek is, azaz olyan címkék, amelyekhez nincs adat (tartalom) rendelve. Amennyiben a `<KEDVES_VENDEG/>` egy üres tag, ekképp adható meg hozzá a nyelvtani szabály:

```
<!ELEMENT KEDVES_VENDEG EPMTY>
```

Az `EMPTY` a kulcsszó, azt jelzi az elemzőprogramnak, hogy a `KEDVES_VENDEG` egy üres elem, így nincs további szerkezeti felépítése. Ebben a példában valószínűleg jelzőként szerepel.

Nézzünk egy másik üres elemet is: `<NEM_KEDVES_VENDEG/>`. Ekkor elképzelhető, hogy vendégkönyvünkben ezt a logikai tulajdonságot a következő XML-szerkezet segítségével fel akarjuk tüntetni:

```
<MINOSITES>                    <MINOSITES>
  <KEDVES_VENDEG/>   vagy        <NEM_KEDVES_VENDEG/>
</MINOSITES>                   </MINOSITES>
```

Itt a `<MINOSITES>` elem nyelvtani szabálya választható elemekből áll, amit a tartalom meghatározáskor a szűrő karakterrel fejezünk ki:

```
<!ELEMENT MINOSITES (KEDVES_VENDEG |
                    ➥NEM_KEDVES_VENDEG)>
```

A következő részben a tulajdonságlistáról, az egyedek létrehozásáról, a dokumentumok megjelenítéséről és az adatfolyam elemzéséről lesz szó.

Nyíri Imre
(inyiri@mol.hu) jelenleg a MOL Rt.-nél dolgozik. Informatikai vállalkozásában az Internet, a Linux és a Java programozás gyakorlati hasznosításával foglalkozik, ennek ellenére örök szerelme még mindig a C++. Kedveli a tudományos és a fantasztikus irodalmat, illetve filmeket (kedvencei: Solaris és 2001 – Űrodüsszeia). Szívesen sportol.