SZAKDOLGOZAT

Nádasdi Attila

Debrecen

2010

Debreceni Egyetem
Informatikai Kar

A szemantikus web és XML-alapú webes rendszerek kapcsolata

Témavezető:
Dr. Adamkó Attila
egyetemi adjunktus

Készítette:
Nádasdi Attila
programtervező informatikus

Debrecen
2010

Plágium – Nyilatkozat

Nyilatkozat a szakdolgozat készítésére vonatkozó szabályok betartásáról.

Alulírott Nádasdi Attila (Neptun-kód: PJL2CN) jelen nyilatkozat aláírásával kijelentem, hogy a *„A szemantikus web és XML-alapú rendszerek kapcsolata”* című szakdolgozat/diplomamunka (a továbbiakban: dolgozat) önálló munkám, a dolgozat készítése során betartottam a szerzői jogról szóló 1999. évi LXXVI. tv. szabályait, valamint az egyetem által előírt, a dolgozat készítésére vonatkozó szabályokat, különösen a hivatkozások és idézések tekintetében.

Kijelentem továbbá, hogy a dolgozat készítése során az önálló munka kitétel tekintetében a konzulenst, illetve a feladatot kiadó oktatót nem tévesztettem meg.

Jelen nyilatkozat aláírásával tudomásul veszem, hogy amennyiben bizonyítható, hogy a dolgozatot nem magam készítettem vagy a dolgozattal kapcsolatban szerzői jogsértés ténye merül fel, a Debreceni Egyetem megtagadja a dolgozat befogadását és ellenem fegyelmi eljárást indíthat.

A dolgozat befogadásának megtagadása és a fegyelmi eljárás indítása nem érinti a szerzői jogsértés miatti egyéb (polgári jogi, szabálysértési jogi, büntetőjogi) jogkövetkezményeket.

hallgató

Debrecen, 2010. 12. 12.

Tartalomjegyzék

Köszönetnyilvánítás

Szeretnék köszönetet mondani témavezetőmnek, Adamkó Attilának a szakdolgozatom készítése közben nyújtott hasznos tanácsaiért, valamint a Magyar Vendor Kft.-nek, akik hozzájárultak, hogy felhasználhassam a dolgozatomhoz készített alkalmazásban az Est.hu-ról származó adatokat.

1. Bevezetés

A W3C (World Wide Web Consortium) által kidolgozott *XML (eXtensible Markup Language, Bővíthető Jelölőnyelv)* napjaink meghatározó szabványává vált az információcsere és az adattárolás területén, köszönhetően egyszerűségének, platformfüggetlenségének, strukturáltságának, valamint modularitásának. Számtalan *XML-alkalmazás*[1], látta meg a napvilágot, a legkülönbözőbb felhasználási területeket átölelve, például:

- XHTML (eXtensible HyperText Markup Language): weboldalak tartalmának a leírásához;
- MathML: matematikai formulák leírásához;
- SVG (Scalable Vector Graphics): vektorgrafikák leírásához;
- XForms: webes űrlapok leírásához;
- SMIL (Synchronized Multimedia Integration Language): multimédiás prezentációk leírásához;
- VoiceXML: ember és számítógép közötti interaktív, hanggal vezérelt párbeszédek leírásához;
- OpenDocument: irodai programcsomagok dokumentumainak leírásához;
- XUL (XML User Interface Language): felhasználói felületek leírásához;
- KML(Keyhole Markup Language): földrajzi információk leírásához.

Az XML-dokumentumok manipulálására pedig jónéhány további nyelv került kifejlesztésre (pl. XQuery, XPath, XSLT), létrehozva ezáltal az XML-technológiák egy egész családját. Meg kell még említenünk a különböző API-kat (Application Programming Interface), pl.

1 Általános célú elemkészlet és dokumentumszerkezet együttese

DOM (Document Object Model), SAX (Simple API for XML), amelyek segítségével programozási nyelvekből, dinamikusan is feldolgozhatjuk az XML-dokumentumainkat. Láthatjuk tehát, hogy egy széles körben elterjedt és támogatott technológiáról van szó, amely fontos szerephez jut a szemantikus web elgondolásban is.

A *szemantikus webet* tulajdonképpen egy következtetések végrehajtására is alkalmas *metaadat* infrastruktúraként képzelhetjük el, és az az igény hívta életre, hogy számítógépeink ne csak olvasni tudják a weben található információt, hanem értelmezni is. Ennek egyik nagy gyakorlati haszna az volna, hogy sokkal hatékonyabbá tehetné a webes keresést. A metaadat pedig nem más, mint adat magáról az adatról, amely azt a cél szolgálja, hogy a gépek számára nyújtson információkat magáról a leírandó dologról. Napjainkban is sok helyen alkalmazzuk a metaadatokat, gondoljunk csak pl. az MP3 formátumú állományoknál használatos ID3 tagokra, amelyek segítségével előadót, címet vagy műfajt rendelhetünk a fájlhoz, megkönnyítve annak katalogizálást és kereshetőséget, azonban metaadatok hiányában ezek a tevékenységek szinte megoldhatatlanok lennének. A szemantikus webnél a metaadatok tárolása leginkább XML formátumban történik, ezért számos, az XML-nél már bevált technológia és eszköz alkalmazható a szemantikus webnél is, valamint kihasználhatók a formátumból származó előnyök is.

Dolgozatomban azt szeretném bemutatni, hogy az XML milyen szerepet tölt be a szemantikus web életében, milyen XML-alapú rendszerek a jellemzőek erre a környezetre, valamint néhány olyan technológiát is be szeretnék mutatni, amelyekkel akár a meglévő XML-alapú webes rendszerünket is be tudjuk kapcsolni a szemantikus web vérkeringésébe, elősegítve ezzel a világhálón található adatok gépi feldolgozhatóságát.

Az első részben egy általános áttekintőt szeretnék nyújtani a szemantikus webről és annak legfontosabb rétegeiről (XML, RDF, RDFS, OWL), aztán a következő részben bemutatok néhány, a szemantikus web gyakorlati alkalmazásában fontos szerepet játszó technológiát, amelyek szorosan kötődnek a XML-hez, majd legvégül ismertetem a dolgozathoz készített szemantikus webalkalmazást, a SEvent-et, amely segítségével igen sokoldalúan és hatékonyan tudunk keresni az Est.hu-ról kinyert, sport- és zenei események között.

2. A szemantikus web

A szemantikus web elképzelés a „web atyja”-ként is aposztrofált Tim Berners-Lee nevéhez köthető, aki valamikor 1998 táján vetette fel először ennek vízióját, majd később így fogalmazta meg ennek az alapgondolatát: *„A szemantikus web nem egy különálló web, hanem a mai web egy kiterjesztése, melyben az információk jól definiált jelentéssel rendelkeznek, jobb együttműködést lehetővé téve az emberek és a számítógépek között.”* [22] Tehát nem egy alapjaitól újjáépített webről lenne szó, hanem „csupán” egy kiterjesztésről, amely a jelenlegi webre épülne rá, kibővítve azt újabb képességekkel, ezáltal megtartva a kompatibilitást a régi webes rendszerekkel. Az új képességek közül az egyik legjelentősebb, hogy a gépeink képesek legyenek az információk szemantikájának a megértésére, ezáltal sokkal hatékonyabbá és egyszerűbbé téve a weben található információk feldolgozását.

2.1 Napjaink világhálójának hiányosságai

A webet alkotó dokumentumok nagyon heterogének (képek, videók, HTML oldalak, PDF, Word állományok stb.), és roppant mennyiségben vannak jelen, ráadásul ezen tartalmak jelentős része folyamatosan változik, van amelyik akár percenként is. Az indexelt weboldalak száma, 2010. november 2-án, a WorldWideWebSize.com[2] adatai szerint legalább 2,86 milliárd volt, de még ez is csak a jéghegy csúcsa, az ún. *sekély web (surface web)*. Létezik a világhálónak egy olyan része is azonban, amely nem elérhető a webes keresők számára, és ami nagyságrendekkel több adatot tartalmaz mint a sekély web, ezt nevezik *mély webnek (deep web)*. A mély webet olyan tartalmak alkotják mint pl.: dinamikusan előállított oldalak, nem szöveges állományok (pl. képek, videók), a regisztrációhoz kötött tartalmak (pl. fórumok, adatbázisok). A mély webből kinyert információk ráadásul sokkal specifikusabbak, pontosabbak, így relevánsabb találatként lehetne őket visszaadni, ezért is volna olyan fontos ezek feltérképezése.

2 http://www.worldwidewebsize.com/

A másik jelentős probléma a webes kereséssel a szemantika hiánya. Napjaink webes keresői, eltekintve egy-két egyelőre gyerekcipőben járó kezdeményezéstől (pl. Wolfram Alpha[3], Haika[4], Sindice[5]) nem tudják értelmezni a szavak jelentését, az számukra nem több mint egy karaktersorozat, és emiatt a keresések találatai között igen sok az irreleváns információ. A szemantika hiányából pedig következik a következtetés hiánya is, az a képesség, hogy meglévő információkból új információkat tudjunk levonni, és ez a képesség már egyfajta mesterséges intelligencia meglétét követelné meg a webes keresőktől. Gondoljunk csak arra például, ha egy keresőbe beírva az „emberszabású majom" kifejezést a kereső csak azokat az oldalakat adja vissza találatként, amelyikben szerepelnek ezek a szavak, de azokat amelyek az ebbe a csoportba tartozó konkrét fajokról (pl. csimpánz, orangután) szól és nem tartalmazza a keresőkifejezést már nem, míg ez a feladat egy átlagos műveltségű ember számára nem okoz túl nagy problémát.

Az előbbiekben megemlített problémákra egyfajta megoldási lehetőséget tudna nyújtani a szemantikus web elképzelés. A mély web feltárásában a metaadatok, a szemantika megragadásában pedig az *ontológiák*, valamint a következtetések végrehajtásának lehetősége nagy segítséget jelentene.

2.2. Ontológiák

Informatikai vonatkozásban az ontológia alatt egy alkalmazási terület jellemző fogalmait, kifejezéseit és a köztük fennálló kapcsolatok formális leírását értjük, aminek a célja a tudás reprezentálása és megosztása. Az ontológia fogalma, nem a szemantikus web sajátja, számos más területén is használatos pl.: mesterséges intelligencia kutatás, rendszertervezés, szoftvertervezés, orvosinformatika, könyvtártudomány, stb.

Megkülönböztethetünk ún. domén vagy szakterületi ontológiákat és felsőbb szintű vagy alapontológiákat. A domén ontológiák egy bizonyos szakterülethez kapcsolódnak pl.: póker, szívsebészet, míg a felsőbb szintű ontológiák jóval általánosabbak, így szélesebb körben al-

3 http://www.wolframalpha.com/

4 http://www.hakia.com/

5 http://sindice.com/

kalmazhatók. Felsőbb szintű ontológia például a Dublin Core[6], vagy a GFO[7] (General Formal Ontology), míg domén ontológiára példa lehet a FOAF[8] (Friend of a Friend), amely személyek, kapcsolataik, valamint tevékenységeik leírására szolgál, vagy a Disease Ontology[9], amely az emberi betegségek ontológiája.

Ontológiák szerkezeti felépítésében sok közös hasonlóságot lehet felfedezni, a leggyakoribb összetevői egy ontológiának a következők:

- *Egyedek:* példányok vagy objektumok
- *Osztályok:* halmazok, kollekciók, osztályok, objektumok típusai
- *Tulajdonságok:* jellemzők, paraméterek, amikkel egy objektum rendelkezik
- *Kapcsolatok:* az osztályok és az egyedek, hogyan tudnak kapcsolódni egy másikhoz
- *Fogalmak:* komplex struktúrák, amely bizonyos kapcsolatokból jönnek létre
- *Korlátozások:* formális leírása annak, hogy minek kell teljesülnie ahhoz, hogy egy kijelentést bemenetként elfogadjunk
- *Szabályok:* ha-akkor formában leírt állítások, amelyek a logikai következetesek végrehajtásához szükségesek
- *Axiómák:* mindig igaz kijelentések, segítségükkel ellenőrizhető az ontológia konzisztenciája
- *Események:* a kapcsolatok vagy tulajdonságok változásai

Az ontológiák létrehozásához különböző formális nyelvek állnak rendelkezésünkre, amelyek gyakran tartalmaznak következtetési szabályokat is. Ezek a nyelvek rendszerint deklaratívak, általában az elsőrendű vagy a leíró logikán alapulnak. A szemantikus webhez kapcsolódó *OWL (Web Ontology Language)* például egy leíró logikán alapuló nyelv, amelyet később részletesen is ismertetek.

---

6 http://dublincore.org/

7 http://www.onto-med.de/ontologies/gfo/

8 http://www.foaf-project.org/

9 http://do-wiki.nubic.northwestern.edu/

2.2. A szemantikus web alkalmazási lehetőségei

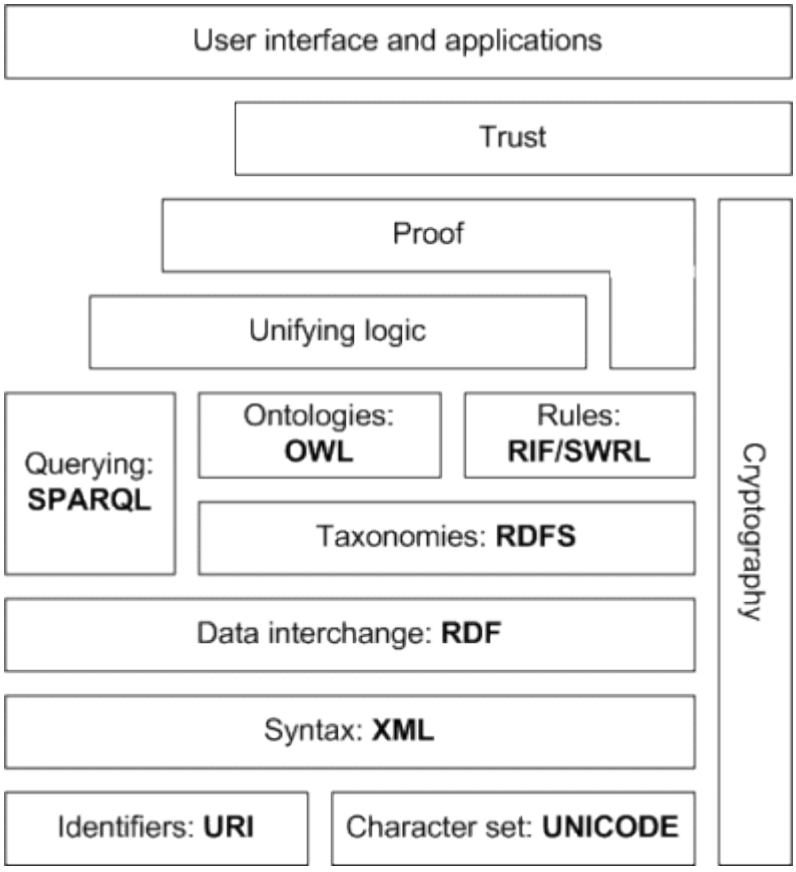
A következőkben, néhány példát szeretnék felhozni a szemantikus web gyakorlatban történő alkalmazási lehetőségeiből.

- **Ágensek:** Intelligens ügynökprogramok, melyek képesek a különböző forrásból származó információk automatikus begyűjtésére és integrálására. Például egy nyaralás megszervezésekor, lefoglalják a repülőjegyeket, kocsit bérelnek, szállást, programokat intéznek, mindezt önállóan és összehangoltan.
- **Szemantikus keresők:** Relevánsabb információkat kaphatunk segítségükkel keresőkérdéseinkre, kiaknázhatóvá válik a mély web tartalma.
- **Szemantikus e-learning:** Megvalósíthatóvá teszi az oktatási rendszerek közötti együttműködést, az oktatási anyagokat az egyén képességeihez szabottan, valamint az általa használt eszköz jellemzőitől függően lehet átadni és megjeleníteni.
- **Szemantikus bioinformatika:** A különböző formában jelenlévő tudományos adatok megosztása, összehangolása az élettudományok, egészségügy, gyógyszerkutatás területén. A W3C saját csoportot (Semantic Web Health Care and Life Sciences Interest Group) is létrehozott az ehhez a területhez kapcsolódó szemantikus web technológiák fejlesztésére, valamint létrejött a BioPAX (Biological Pathways Exchange) nyelv is, amely a biológiai folyamatokhoz kapcsolódó adatok integrálását, cseréjét, megjelenítését, elemzését hivatott elősegíteni.
- **Vállalati alkalmazás- és adatintegráció:** Egy nagyobb vállalatnál több alkalmazás is tárolhatja ugyanazt az adatot, ami redundanciát okozhat és inkonzisztenssé teheti azokat, illetve van amikor az adatok elosztottan vannak jelen a különböző rendszerekben, ilyenkor ezek összekapcsolása okozza a nehézséget. Ezekre a problémára próbál megoldást adni az alkalmazás- és adatintegráció. Egy ilyen feladatra kifejlesztett eszköz például a SILK (System Integration via Logic and Knowledge), melynek kifejlesztésében magyar kutatók is részt vettek.

- **Mobil-kooperatív számítástechnika:** Mobil számítástechnikai eszközök, vezeték nélküli hálózati kapcsolatának, ad hoc konfigurálásának támogatása. Célja, hogy azok a készülékek, amelyeket nem terveztek együttműködésre, fel tudják ismerni és ki tudják használni a másik nyújtotta szolgáltatásokat. Ehhez a területhez kapcsolódnak a W3C Composite Capability/Preference Profile-ja (CC/PP), és a WAP Forum User Agent Profile-ja (UAProf).

2.3. A szemantikus web architektúrája

A szemantikus web egy hierarchikusan felépülő rétegekből álló infrastruktúra, melyben minden réteg felhasználja az alatt lévők képességeit. A következőkben ezeket a rétegeket szeretném bemutatni, különös tekintettel az XML-re, és a vele szorosabb kapcsoltban levő technológiákra.



2.1. ábra: A szemantikus web architektúrája
http://semanticweb.org/wiki/File:Semantic-web-stack.png

- **URI** (*Uniform Resource Identifier*): *Egységes Erőforrás Azonosító,* amely az erőforrások egyértelmű azonosítására szolgál, segítségével egyértelmű állítások fogalmazhatók meg erőforrásokról.

- **Unicode:** Karakterkódolási szabvány, amely lehetővé teszi a világ bármely nyelvének az egységes kódolását, ez a szemantikus web globális méretéből adódóan fontos követelmény.
- **XML** (*eXtensible Markup Language*): A metaadatokat tartalmazó dokumentumok leggyakrabban XML-szintaxist használnak, mivel ez már egy bizonyított formátum az adatcsere területén.
- **RDF** (*Resource Description Framework*): *Erőforrás-leíró Keretrendszer*, melynek segítségével erőforrásokról tehetünk állításokat, ún. *hármasok (triples)* segítségével, gépek számára is feldolgozható formában.
- **RDFS** (*RDF Schema*): Az RDF Séma lehetővé teszi az RDF adatok csoportjainak, illetve tulajdonságainak a leírását.
- **SPARQL** (*SPARQL Protocol And RDF Query Language*): Deklaratív lekérdező nyelv RDF adatok számára.
- **OWL** (*Web Ontolgy Language*): *Web Ontológia Nyelv*, amely kibővíti az RDFS-t, fejlettebb módszereket biztosít az RDF állítások szemantikájának leírásához.
- **RIF/SWRL** (*Rule Interchange Format/Semantic Web Rule Language*): Ezek a technológiák biztosítják a szabályok alkalmazhatóságát a szemantikus weben. Akkor célszerű ennek a rétegnek a használata, ha olyan kapcsolatokat kell leírni, amire az OWL nyelvben implementált leíró logika nem képes.

Megvalósításra váró rétegek

- **Egyesítő logika** (*Unifying logic*): Ez a réteg nagyjából azt a célt szolgálná, hogy az alsóbb rétegek különböző modellszemantikáit egy közös modellbe foglalná össze, ezáltal egy egységes interfészt biztosítva az alkalmazásoknak, amelyek a szemantikus webhez akarnak kapcsolódni.
- **Bizonyítás** (*Proof*): Matematikai módszerekkel biztosítaná a következtetések helyességének ellenőrzését. Erre a feladatra egy lehetséges megoldást jelenthet, a még munkatervezet fázisában lévő *PML (Proof Markup Language)* nyelv.

- **Kriptográfia** (*Cyptography*): Ez a réteg lenne felelős azért, hogy meg tudjunk bizonyosodni, hogy az RDF kijelentések megbízható forrásból származnak. Egy lehetséges eszköz ehhez a digitális aláírás használata.
- **Bizalom** (*Trust*): Az adatok megbízhatóságának értékelésére szolgáló réteg.
- **Felhasználói felület és alkalmazások** (*User inteface and applications*): A legfelső réteg, amely az emberek és a szemantikus webalkalmazások közötti kapcsolatot biztosítaná**.**

A következő részekben a szemantikus web legfontosabb rétegeit, név szerint az XML, RDF, RDFS és OWL nyelveket részletesebben is be fogom mutatni.

2.3.1. XML

Az XML az *SGML (Standard Generalized Markup Language, Strukturált Általános Jelölőnyelv)* nevű ISO szabvány webre optimalizált, egyszerűsített változata, segítségével strukturált formában lehet adatokat tárolni, továbbítani. Az XML-t és a vele kapcsolatos legtöbb technológiát a W3C (World Wide Web Consortium) dolgozta ki és tartja karban.

2.3.1.1. Az XML előnyei és hátrányai

Előnyök:

- Egyszerű, emberek és gépek számára is olvasható szöveges formátum.
- Platform- és gyártófüggetlen.
- Logikailag ellenőrizhető formátum.
- Öndokumentáló formátum.
- Támogatja az Unicode-ot.
- Ábrázolható vele a legtöbb, informatikában használatos adatstruktúra.
- Az SGML miatt, már nagy tapasztalat áll rendelkezésre kezelését illetően.

Hátrányok:

- Bőbeszédű, a redundáns adatok tárolása miatt nagy méretű.
- Sok a felesleges SGML hagyaték.
- Nem támogatja az adattípusokat, így pl. a számokat is szövegként kell tárolni.
- Nem hierarchikus adatstruktúrák modellezése nehézkes.

2.3.1.2. XML-dokumentumok felépítése

Egy XML-dokumentum általában két fő részből tevődik össze, a *fejrészből* (*prolog*) és a *dokumentumelemből (gyökérelem)*.

A fejrész tartalmazza az XML-deklarációt, amely az XML 1.1 verzió használat esetén kötelező: `<?xml version="1.1"?>` A fejrész ezen kívül még tartalmazhat a karakterkódolásra és a *dokumentumtípus-deklarációra* vonatkozó információkat vagy *feldolgozó utasításokat* is egyebek mellett. A dokumentumtípus-deklaráció, amely a dokumentum típusát, tartalmát, szerkezetét határozza meg, elhelyezhető egy külső fájlban, de akár magában a dokumentumban is. A feldolgozó utasítások az alkalmazásoknak szólnak, amelyek a XML-dokumentumot felhasználják, és ezt számukra az XML-feldolgozó továbbítja.

Az XML-szintaxis egy hierarchikusan felépülő dokumentumot határoz meg, a hierarchia csúcsán a dokumentumelem foglal helyet. Lényegében az elemek egy fa struktúrát alkotnak, ezért is nevezik a dokumentumelemet más néven gyökérelemnek, amelynek elhelyezése a dokumentumban kötelező, viszont pontosan csak egy ilyen elem fordulhat elő.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE catalog SYSTEM "catalog.dtd">

<catalog>
  <cd>
        <title>Empire Burlesque</title>
        <artist>Bob Dylan</artist>
        <country>USA</country>
        <company>Columbia</company>
        <price>10.90</price>
        <year>1985</year>
    </cd>
    <cd>
        <title>Hide your heart</title>
        <artist>Bonnie Tyler</artist>
        <country>UK</country>
        <company>CBS Records</company>
        <price>9.90</price>
        <year>1988</year>
    </cd>
</catalog>
```

2.1. példa: XML-dokumentum

Elemek

Egy XML-dokumentumban a lényegi információt az *elemek* hordozzák. Egy elem tipikusan egy *nyitócímkéből*, a *tartalomból* és egy *zárócímkéből* áll pl.: `<elem>tartalom</elem>`. A nyitó- és zárócímkékben megadott elemneveknek meg kell egyezniük, egyébként a névadásra nincs szigorú megkötés, ezeket a neveket nevezzük egyébként az *elem típusának*. A tartalom lehet egyszerű, ilyenkor az értéke valamilyen karakteres adat, vagy összetett ilyenkor a tartalmat az elembe beágyazott elemek alkotják. Létezik még az ún. *üreselem*, melynek nincs tartalma, ez megadható egy nyitó- és egy zárócímkével pl.: `<elem></elem>` vagy egy üreselem címkével pl.: `<elem />`.

Az elemek nyitócímkéje tartalmazhat *attribútumokat (tulajdonságokat)*, amelyek név-érték párok és az elem típusához adhatnak további kiegészítő információkat, segítségükkel tömörebbé és átláthatóbbá is válnak dokumentumaink, pl.:

```
<a href="http://www.w3.org/TR/xml11/">XML 1.1</a>
```

2.3.1.3. Jólformázott és érvényes XML-dokumentumok

Egy XML-dokumentumot *jólformázottnak* nevezzük, ha teljesíti az alábbi kritériumokat:

- A dokumentumnak pontosan egy legfelső szintű eleme (gyökérelem) van, a többi elem ebbe van beágyazva.
- Minden nyitó címkéhez tartoznia kell egy megfelelő záró címkének.
- Az elemeket csak úgy lehet egymásba ágyazni, ha nem fedik át egymást.
- A nyitócímkében található elemtípus nevének pontosan meg kell egyeznie a hozzá tartozó zárócímkében található típusnevével. Az elemtípusok nevei kis- és nagybetű érzékenyek, tehát az „ELEM” típusnév nem egyezik meg az „elem” névvel.
- Minden hivatkozott, elemzett egyed jólformázott.

A jólformázottság azért fontos kitétel egy XML-dokumentummal szemben, mert ennek hiányában az alkalmazások nem tudnák feldolgozni azokat. A jólformázottság ellenőrzéséhez szükséges szabályokat az XML-specifikáció határozza meg.

Ahhoz, hogy egy XML-dokumentum *érvényes* is legyen már szigorúbb feltételeknek kell megfelelni, mintha csak a jólformázottság lenne a szempont. Érvényes akkor lehet egy jólformázott dokumentum, ha az XML-dokumentum szerkezete és tartalma mindenben megfelel a magában a dokumentumban, vagy egy külső állományban meghatározott szabályoknak. Ezek a szabályok többféleképpen is megfogalmazhatók a különböző *sémanyelvek* segítségével.

Az érvényesség ellenőrzése, vagyis, hogy az adott dokumentum lehet-e a sémában megfogalmazott dokumentumosztály egy példánya, az XML-feldolgozó feladata. Az egyik legrégebbi és legegyszerűbb sémanyelv a DTD (Document Type Definition), amely egy nyelvtan alapú nyelv. Korszerű sémanyelvek pl.: XML Schema (objektumorientált), RELAX NG (nyelvtan alapú), Schematron (szabály alapú).

2.3.1.4. XML-névterek

Abban az esetben, ha különböző forrásból származó XML-adatokat szeretnénk egy dokumentumon belül egyszerre használni, az az elemnevek és attribútumnevek névütközéséhez vezethet. Az elemneveknek egy dokumentumon belül, az attribútumneveknek egy elemen belül

egyedieknek kell lenniük, viszont két azonos név esetén nem tudnánk megkülönbözteti őket. Erre a problémára nyújt megoldást az XML-névtér mechanizmusa. Ha az azonos neveket különböző névterekhez rendeljük, majd a névtér nevével minősítjük azokat, már egyszerűvé válik a megkülönböztetésük. Ahhoz, hogy névtereket tudjunk használni egy XML-dokumentumban, először is deklarálni kell azokat, valamelyik elem nyitócímkéjében, ehhez hasonló módon:

```
<catalog xmlns:book="http://mydomain.com/book
    xmlns:cd="http://mydomain.com/cd">
    ...
</catalog>
```

Tehát ez egy speciális attribútum hozzáadásának fogható fel, amely rendelkezik egy `xmlns` nevű előtaggal, ami után következik a *névtérelőtag* kettősponttal elválasztva, majd pedig a névtér neve, – ami egy URI – az attribútumok értékének megadásánál ismertetett módon. Az URI szerepe a névtér egyediségének biztosítása. Deklaráció után tudjuk alkalmazni a névtérrel minősített neveket, `névterelőtag:lokális_rész` formában, pl.: `book:TITLE`. A névtér hatásköre az az elem, amelyben deklaráltuk és az azt tartalmazó elemek. Létezik a „lyuk a hatáskörben” jelenség, tehát egy érvényben lévő deklaráció felülírható. Lehetőség van *alapértelmezett névtér* megadására is, tehát azok a nevek, amelyek nincsenek minősítve egyéb névtérrel, azok automatikusan ebbe a névtérbe kerülnek, azonban ez a hatáskörbe tartozó attribútumnevekre nem vonatkozik, azokat mindig minősíteni kell. Alapértelmezett névtér deklarálása:

```
<catalog xmlns="http://mydomain.com/book>
```

Láthatjuk tehát, hogy a névterek használata roppant modulárissá és egyben újrafelhasználhatóvá teszi XML-dokumentumainkat.

2.3.1.5. Relatív URI-k alkalmazása

Az XML-dokumentumokban lehetőségünk van egy bázis URI megadására az `xml:base` attribútum segítségével, amit később az értelmező felhasznál a relatív címek feloldására. A bázis URI-k deklarálása és hatásköre a névtereknél ismertetett módon alakul. Ha nem adunk

meg explicit bázis URI-t, akkor is alkalmazhatunk relatív URI-kat, ilyenkor a bázis URI az aktuális dokumentum URI-ja lesz. Példa bázis URI-k használatára:

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

<catalog xml:base="http://en.wikipedia.org/"
    xmlns:xlink="http://www.w3.org/1999/xlink">
  <cd>
      <title>Empire Burlesque</title>
      <artist>Bob Dylan</artist>
      <country>USA</country>
      <company>Columbia</company>
      <price>10.90</price>
      <year>1985</year>
      <wikipedia xlink:href="wiki/Bob Dylan">Wikipedia</wikipedia>
    </cd>
</catalog>
```

2.1.3.6. XML-dokumentumok transzformációja

A W3C által kidolgozott *XSL (eXtensible Stylesheet Language Family)* nyelvcsalád célja az XML-dokumentumok transzformációjának és megjelenítésének elősegítése. Az XSL-t alkotó nyelvek: *XPath (XML Path Language)*, *XSLT (eXtensible Stylesheet Language Transformations*), *XSL-FO (XSL Formatting Objects)*. Az XSL megszületése annak a gondolatnak köszönhető, hogy érdemes az adatok tárolását és a megjelenítését különválasztani, így az XML-dokumentumban nem kell törődni azzal, hogy milyen eszközön lesz megjelenítve a dokumentum tartalma, azt nyugodtan rábízhatjuk a *XSL-stíluslapokra*.

XSL-FO

Az XSL-FO – amely maga is egy XML-alkalmazás – leírja, hogy egy XML-dokumentum, hogyan jelenjen meg emberek számára olvasható formában. Funkcióját tekintve hasonlít a CSS-re (Cascading Style Sheet), viszont egy XSL-FO-dokumentum rendszerint nem csak a formázási utasításokat tartalmazza, hanem az eredeti dokumentum adatait is. Az XSLT és XSL-FO kiegészítik egymást, gyakran egy stíluslapon belül van kombinálva a két nyelv. Az XSL-FO-dokumentumokat FOP-nak (Formatting Object Processor) nevezett alkalmazások konvertálják át a végső, megjelenítésre kerülő formátumra, ami lehet PDF (Portable Document Format), PS (PostScript) vagy RTF (Rich Text Format) is akár.

XPath

Az XPath-nak létezik egy régebbi 1.0-s és egy újabb 2.0-s változata. A 2.0-s verzió jóval fejlettebb, nagyobb tudású, bár a legtöbb esetben az 1.0 által nyújtott lehetőségek is elegendőnek bizonyulnak.

Az XPath működéséhez egy saját adatmodellt alkalmaz, amely az XML-dokumentumok logikai szerkezetére épül, az XPath 2.0-ban ezt a modellt *XDM-nek (XQuery/XPath Data Model)* nevezik. Az adatmodell egy fa szerkezetet határoz meg, ebben a fában lehet az XPath segítségével többek között csomópontokat kijelölni vagy mintaillesztést végrehajtani.

Az XPath egy tömör, nem XML szintaxist alkalmaz, ezeket nevezzük *XPath-kifejezéseknek*. A kifejezések kiértékelése az adott aktuális helyzet függvényében történik, amelyet az XSLT vagy XPointer értelmező határoz meg. A kifejezések érteke a következő típusok közül kerülhet ki: lebegőpontos szám, sztring, logikai, csomóponthalmaz.

XSLT

Az XSLT ajánlásból, ugyanúgy mint az XPath-ból, létezik egy 1.0-s és 2.0-s verzió. Az XSLT 1.0-t a XPath 1.0-val, míg a XSLT 2.0 leginkább az XPath 2.0-val való együttműködésre tervezték, valamint a használt adatmodelljük is közös az azonos verziószámú XSLT és XPath nyelveknek.

Az XSLT-t XML-dokumentumok más XML-dokumentumokká transzformálására alkották meg, de elég gyakran az XML-dokumentumok (X)HTML formában való megjelenítésére is alkalmazzák. A transzformációkat XSLT-stíluslapok formájában lehet megadni, amelyek maguk is az XML-szintaxist követik. A stíluslapokban *sablonok* segítségével tudjuk megadni, hogy a forrásdokumentum illeszkedő részére milyen átalakítás hajtódjon végre. Az eredeti dokumentumból szükséges csomópontok kijelölésére az XSLT XPath-kifejezéseket használ. A stíluslapokat az XSLT-értelmező dolgozza fel, amely aztán előállítja a kívánt céldokumentumot az eredeti megváltoztatása nélkül. A stíluslapokat leginkább egy külön állományban hozzuk létre, majd azt csatoljuk az átalakítani kívánt dokumentumhoz, de lehetőség van a stíluslap forrásdokumentumba való beágyazására is.

2.3.2. RDF

A W3C által kidolgozott *RDF (Resource Description Framework, Erőforrás-leíró Keretrendszer)* segítségével erőforrásokhoz tudunk metaadatokat társítani, elősegítve ezzel az információk gépi feldolgozhatóságát. Korábban is már sokféle metaadat megtalálható volt a weben, gondoljunk csak pl. a HTML `meta` elemére, vagy az MP3 állományokhoz csatolható ID3 címkékre. A metaadatok sokfélesége viszont problémás is abban a tekintetben, hogy mindegyik valami egyedi megoldást kíván a feldolgozás szempontjából. Az RDF előnye abból származik, hogy a metaadatokat egységes módon tudjuk hozzárendelni az erőforrásokhoz, és itt erőforrás alatt szinte bármit érthetünk, ami rendelkezik URI-val. Mivel az RDF egy univerzális keretrendszer, ezért a különböző forrásokból származó RDF adatokat egységesen tudják kezelni és kombinálni az őket felhasználó alkalmazások. Mielőtt jobban elmélyednénk a RDF világában, tisztáznunk kell az URI-k szerepét.

2.3.2.1. Az URI-k szerepe a szemantikus weben

Az *URI (Uniform Resource Identifier)*, azaz *Egységes Erőforrás-azonosító*, ahogy a neve is sugallja, az erőforrások egyértelmű azonosítását biztosítja. Az RDF-ben például arra használjuk, hogy egyértelmű állításokat fogalmazhassunk meg általa erőforrásokról. Ha két RDF kijelentés ugyanazt az URI-t használja, még ha a világháló eltérő részén találhatók is, biztosak lehetünk benne, hogy ugyanarról a dologról tesznek állításokat. URI-ja bárminek lehet, és ehhez nem kell feltételen elérhetőnek lennie az interneten. Az URI-k egy részhalmazát alkotják az *URL-ek (Uniform Resource Locator, Egységes Erőforrás-helymeghatározó)* és *URN-ek (Uniform Resource Name, Egységes Erőforrás-név)* is. URL-ekkel (pl. `http://sevent.dyndns.org`, ami egy weboldalt azonosít) azonosítjuk az interneten elérhető erőforrásokat, míg az URN-ek (pl. `urn:isbn:0-520-02356-0`, ami egy könyvet azonosít) perzisztens, elhelyezés-független azonosítókat takarnak.

Az URI-k lehetnek *relatívak* és *abszolútak* is. Az erőforrások azonosítása csak abszolút URI-k használata esetén egyértelmű. Relatív URI-k használata esetén szükség van egy bázis URI-ra, amely segítségével fel tudjuk oldani őket, vagyis kiegészítjük őket abszolút URI-vá. A relatív URI-k előnye abban áll, hogy az URI-k változásakor elég csak a bázis URI-t módosítani, nem kell minden egyes erőforrásra való hivatkozást egyenként átírni.

Az URI-kat el lehet látni opcionális *erőforrásrész-azonosítóval*, amelyet az URI-tól egy „#” jel választ el (pl.: `http://www.example.org/index.html#section2`), amely az URI-val azonosított erőforrás egy bizonyos részét azonosítja. Erőforrásrész-azonosítót csak akkor használhatunk a szabvány szerint, ha az URI-val azonosított erőforrás a weben keresztül elérhető.

2.3.2.2. Az RDF adatmodellje

Az RDF specifikációja egy halmazelméleten alapuló adatmodellt definiál a metaadatok leírásához, amely az alábbi halmazokat különbözteti meg:

- *Erőforrások (Resources):* Azon dolgok összessége, amelyekre egy kijelentés vonatkozhat. Az erőforrásokat *RDF URI-hivatkozások (URIref)* azonosítanak, ezek olyan URI-k, amelyekhez opcionálisan erőforrás-azonosító is kapcsolható, illetve tartalmazhatnak Unicode karaktereket is. Erőforrás lehet pl. egy weboldal, vagy egy kép a weboldalon, de nem kell feltétlen a weben elérhetőnek lennie.
- *Tulajdonságok (Properties):* Az erőforrásokhoz kapcsolódó jellemzők, amelyeknek szintén erőforrásoknak kell lenniük. A tulajdonságoknak van jelentésük, meghatározható milyen értékeket vehetnek fel és milyen erőforrásokhoz kapcsolhatók, valamint, hogy milyen kapcsolatban vannak más tulajdonságokkal.
- *Literálok (Literals):* Fix, előre meghatározott karaktersorozatok.
- *Kijelentések (Statements): Hármasok* vagy *tripletek (triples)*, amelyek *alanyból (subject)*, *állítmányból (predicate)* és *tárgyból (object)* állnak. Az alany tetszőleges RDF-erőforrás, az állítmány egy tetszőleges RDF-tulajdonság, míg a tárgy egy tetszőleges RDF-erőforrás vagy literál lehet.

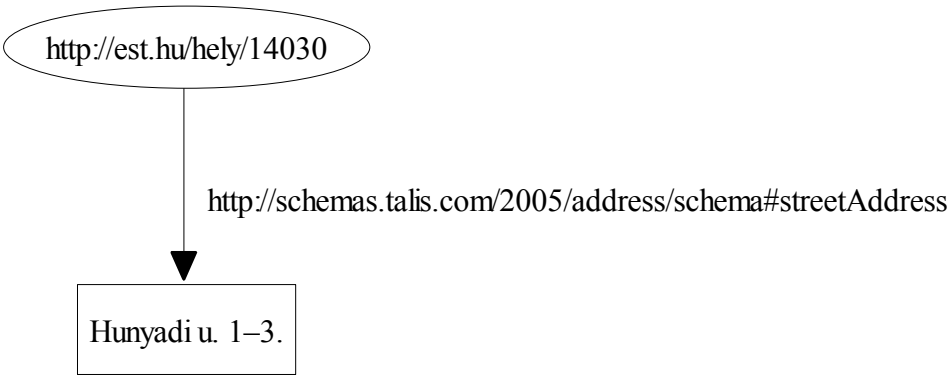
Az RDF-adatmodell szemantikája szerint a hármasokban megfogalmazott kijelentések igazak. Az RDF tulajdonképpen bináris relációkat fogalmaz meg erőforrásokkal kapcsolatban.

2.3.2.3. Az RDF gráfmodellje

Az RDF-kijelentéseket egy gráffal lehet a legszemléletesebben modellezni. Az alanyt és a tárgyat egy-egy csomópontként, míg az állítmányt egy irányított élként ábrázoljuk, ahol az él

az alanyt és a tárgyat köti össze és tárgy irányába mutat. Az RDF URI-hivatkozással rendelkező csomópontokat egy ellipszissel, a literál értékkel rendelkezőket egy téglalappal jelöljük, míg az éleket az állítmány RDF URI-hivatkozásával címkézzük a gráfban.

Vegyük azt a természetes nyelven leírt kijelentést, hogy „A debreceni Kölcsey Központ címe Hunyadi u. 1-3”. Ezt a kijelentés a következő gráffal ábrázolhatjuk:

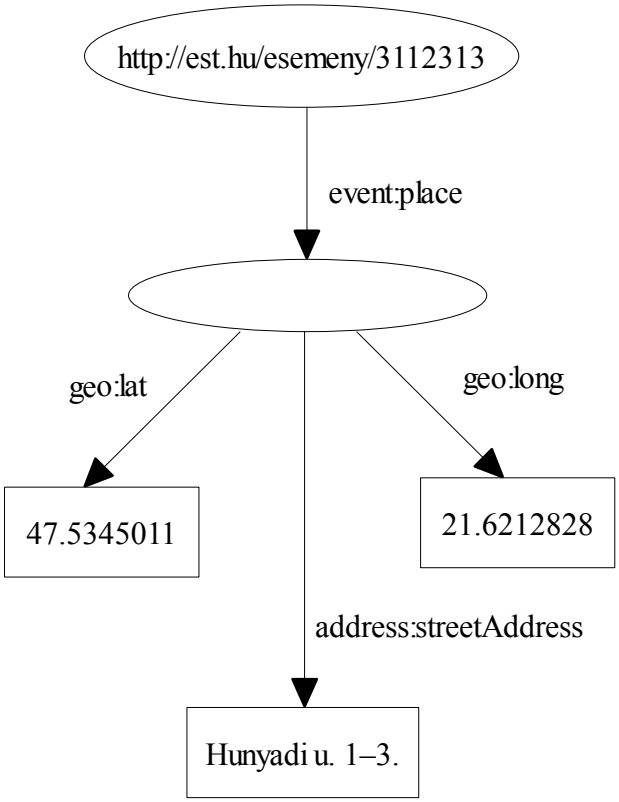


2.2. ábra: Egy egyszerű RDF-kijelentés gráfmodellje

A fenti gráfban:

- Alany: A „http://est.hu/hely/14030” URI-val azonosított „Kölcsei Központ”.
- Állítmány: Az „http://schemas.talis.com/2005/address/schema#streetAddress” RDF URI-hivatkozással azonosított RDF-tulajdonság, amely egy „streetAddress” nevű erőforrás-azonosítóval is rendelkezik.
- Tárgy: Egy literál, amely az állítmányban megfogalmazott RDF-tulajdonság értékét tartalmazza, amely nem más, mint a „Hunyadi u. 1-3.”.

A valós világban azért ettől jóval bonyolultabb kijelentések is előfordulnak, például amikor valamilyen strukturált tulajdonságértéket akarunk ábrázolni. Tegyük fel, hogy adva van egy esemény, amely rendelkezik helyszínnel, viszont a helyszín alkotó elemeket (pl. város, cím, koordináták) külön-külön is kezelhetővé szeretnénk tenni. Ebben az esetben a kapcsolat az esemény és az egyes helyszínt alkotó komponensek között már nem lesz bináris a helyszín reláció esetén, viszont RDF-ben csak bináris kapcsolatokat tudunk kezelni. Szerencsére minden magasabb aritású kapcsolatot fel lehet bontani bináris kapcsolatokra, az RDF-ben erre az egyik megoldás az ún. *üres csomópont* használata.



2.3. ábra: Üres csomópont használata

A 2.3. ábrán a névtérelőtagokhoz tartozó RDF URI-hivatkozások:

- place: http://purl.org/NET/c4dm/event.owl#
- geo: http://www.w3.org/2003/01/geo/wgs84_pos#
- address: http://schemas.talis.com/2005/address/schema#

A 2.3. ábrán látható példában az üres csomópont használatára láthatunk példát, amiben egy esemény (konkrétan egy Koncz Zsuzsa koncert) helyszínéről teszünk kijelentést, ahol a helyszín egy összetett tulajdonságérték (helyszín címe, helyszín koordinátái) és nem azonosítható RDF URI-hivatkozással. Az ilyen modellek szerializált alakjában, viszont szükséges az üres csomópontok azonosíthatósága, amire alkalmazhatunk például lokális azonosítókat.

2.3.2.4. RDF/XML

Az RDF nem nem határoz meg konkrét szintaxist az RDF-gráfok leírásához, viszont rendelkezik egy XML-alapú szintaxissal, amely alkalmazható ezen célokra és az alkalmazások közötti adatcserét is támogatja. A következőkben ezt a szintaxist szeretném bővebben ismertetni.

Az `rdf:RDF` elem

Az RDF/XML-be szerializált RDF-gráf gyökéreleme legtöbbször az `rdf:RDF` elem, ahol az `rdf` névtérelőtag a `http://www.w3.org/1999/02/22-rdf-syntax-ns#` névteret azonosítja, amely az RDF specifikáció részét képzi. Az `rdf:RDF` elem tartalmazza általában a névtér-deklarációkat, de ez nem kötelező. Ha `rdf:RDF` elemen belül csak egy legfelső szintű `rdf:Description` elemet található, az `rdf:RDF` elem el is hagyható.

Csomópontelemek, tulajdonságelemek, tuljadonság-attribútumok

Az `rdf:Description` *csomópontelem*, az RDF-gráf olyan alany- vagy tárgycsomópontjainak a leírására szolgál, amelyek vagy üres csomópontok vagy RDF URI-hivatkozással azonosíthatók. Az RDF-gráf állítmány éleit a megfelelő alanyt reprezentáló `rdf:Description` elembe ágyazva önálló elemként, ún. *tulajdonságelemként*, vagy akár ún. *tulajdonság-attribútumként* is ábrázolhatjuk, ez utóbbit csak abban az esetben, ha az RDF-tulajdonság értéke karakter literál.

Megosztott alany alkalmazása

Gyakran előforduló eset, hogy kijelentésekben egy alanyról több állítást is megfogalmazunk, azaz egy alanycsomópontból több állítmányél is kiindul. Ezekben az esetekben az RDF/XML-ben lehetőség van rövidített szintaxist alkalmazni, vagyis az alanycsomópontot reprezentáló elembe beágyazhatjuk gyermekelemekként a hozzá kapcsolódó tulajdonságelemeket.

Üres tulajdonságelemek

Ha egy tárgycsomópontból nem indul ki egyetlen állítmányél sem és az ezt reprezentáló elem „`<rdf:Description about="…`” alakú, akkor ez a szerkezet rövidíthető. A tárgyat rep-

rezentáló csomópontelemet elhagyjuk és a szülő tulajdonságelemet kibővítjük egy `resource` attribútummal, amelynek értékéül adjuk a tárgycsomópont RDF URI-hivatkozását.

Lássunk egy példát is az eddigiekre:

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

<!-- Névtér deklarációk -->
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:event="http://purl.org/NET/c4dm/event.owl#"
    xmlns:time="http://purl.org/NET/c4dm/timeline.owl#">

    <!-- Megosztott alany és tulajdonság-attribútum -->
    <rdf:Description rdf:about="http://est.hu/esemeny/3114047"
        rdfs:label="Horváth Gábor Trió">

        <!-- Tulajdonságelem -->
        <rdfs:comment>r'n'b, pop, filmzene</rdfs:comment>
        <event:place>
            <rdf:Description rdf:about="http://est.hu/hely/27236">
            </rdf:Description>
        </event:place>

        <!-- Üres tulajdonságelem -->
        <event:place rdf:resource="http://est.hu/hely/1111"/>
        <event:time>

            <!-- Üres csomópont -->
            <rdf:Description>
                <time:start>2010-11-24T22:00:00</time:start>
            </rdf:Description>
        </event:time>
    </rdf:Description>

</rdf:RDF>
```

Az `xml:lang` attribútum

A karakter literál tartalom nyelvének megállapítására az RDF/XML az `xml:lang` attribútumot biztosítja számunkra. Csomópontelemben vagy tulajdonságelemben használható a benne foglalt tartalom nyelvének jelzésére. Az XML-literálokat tartalmazó tipizált literálokra nincs hatással ez az attribútum. Például:

```
<gn:alternateName xml:lang="hu">Budapest</gn:alternateName>
```

Literálok

Az RDF/XML lehetőséget biztosít *tipizált literálok* alkalmazására tulajdonságelemekben, az RDF-tulajdonságok értéktípusának a meghatározásához. A karakter literál-csomópontnál ismert szintaxist kell követni, kiegészítve azt egy `rdf:datatype="`*`datatypeURIref`*`"` attribútummal, ahol a *`datatypeURIref`* az adattípusra mutató RDF URI-hivatkozás. Például:

```
<time:start
rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">
2010-11-24T22:00:00</time:start>
```

Az RDF/XML lehetővé teszi *XML-literálok* megadását is a tipizált literáloknál ismertetett esetekben, de csak a tulajdonságelem tartalmaként, amit egy `rdf:parseType:"Literal"` attribútummal jelzünk.

Az rdf:ID attribútum

Az RDF/XML-ben abszolút RDF URI-hivatkozások helyett relatívakat is alkalmazhatunk az `rdf:ID` attribútum segítségével. Az `rdf:ID`-t az `rdf:about` helyett használhatjuk egy csomópontelemben. A csomópontelem relatív címét megkapjuk az `rdf:ID` értékét kiegészítjük az elé beszúrt „#” jellel, tehát az `rdf:ID="id1"` megegyezik az `rdf:about="#id1"`-val. Ráadásul az `rdf:ID` használatának az az előnye is megvan, hogy ellenőrzi a megadott relatív RDF URI-hivatkozás egyedi-e, az implicit vagy explicit megadott bázis URI hatáskörén belül.

Üres csomópontok

Az üres csomópontok azonosítására az `rdf:nodeID` attribútum kínál lehetőséget, amely értékeként adható meg egy *ürescsomópont-azonosító*. Az ürescsomópont-azonosító egy lokális azonosító, és a dokumentumon beül egyedinek kell lennie. Ott, ahol hivatkozzuk az üres csomópontot, az `rdf:about` és az `rdf:resource` attribútumok helyett az `rdf:nodeID` attribútumot kell használnunk az ürescsomópont-azonosítóval.

Tipizált csomópontelemek (példányok)

RDF-ben lehetőség van, hogy egy erőforrásról kijelenthessük az egy osztály példánya, ezt RDF/XML-ben az `rdf:type` attribútum segítségével tudjuk megfogalmazni. Lehetőségünk van egy rövidített szintaxis alkalmazására is, amelyben az `rdf:Description` elem nevét lecseréljük az `rdf:type` értékekére, minősített név alakban.

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:event="http://purl.org/NET/c4dm/event.owl#"
    xmlns:time="http://purl.org/NET/c4dm/timeline.owl#">

    <!-- Tipizált csomópontelem rövidített megadása -->
    <event:ZeneEvent rdf:about="http://est.hu/esemeny/3114047">
        <rdfs:label>Horváth Gábor Trió<rdfs:label>
        <rdfs:comment>r'n'b, pop, filmzene</rdfs:comment>
        <event:place rdf:resource="http://est.hu/hely/27236"/>
        <event:time rdf:nodeID="ido_3114047"/>
    </event:ZeneEvent>

    <!-- Tipizált csomópontelem -->
    <rdf:Description rdf:nodeID="ido_3114047">
        <rdf:type rdf:resource="time:Interval"/>
        <time:start>2010-11-24T22:00:00</time:start>
    </rdf:Description>

</rdf:RDF>
```

Magasabb rendű kijelentések

A való életben gyakran teszünk kijelentések kijelentésekről, például: „Peti azt mondja, hogy Zoli megette a sárgaborsó főzeléket”. Legyen az alanyunk „Peti”, az állítmány a „mondja” míg a tárgy a „Zoli megette a sárgaborsó főzeléket”, ami maga is egy kifejezés, jó lenne tehát szétbontanunk és egy külön erőforrással azonosítani, amit aztán meg tudnánk adni az eredeti kijelentésünk tárgyaként. Az előbb ismertetett folyamatot *tárgyiasításnak*, más szóval *reifikációnak* nevezzük, ennek segítségével az összetett kijelentések szemantikáját is meg tudjuk őrizni. A reifikiált kijelentés az `rdf:Statement` osztály egy példánya kell hogy legyen. A reifikált kijelentés alanyát az `rdf:subject`, az állítmányát az `rdf:predicate`, míg a tárgyát a `rdf:object` tulajdonságelemekkel tudjuk megadni az RDF/XML-ben. A felhozott példa így néz ki RDF/XML-ben:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:ex="http://example.org/reif/">

<rdf:Statement rdf:nodeID="id_1">
    <rdf:subject rdf:resource="ex:Zoli"/>
    <rdf:predicate rdf:resource="ex:megette"/>
    <rdf:object>sárgaborsó főzelék</rdf:object>
</rdf:Statement>

<rdf:Description rdf:about="ex:Peti">
    <ex:mondja rdf:nodeID="id_1"/>
</rdf:Description>

</rdf:RDF>
```

Konténerek

Az RDF biztosít olyan beépített típusokat, tulajdonságokat, amelyek segítségével a dolgokat csoportba rendezhetjük, ezeket hívjuk konténereknek. Egy konténer eleme lehet erőforrás vagy literál is. Az RDF a következő típusú konténereket nyújtja számunkra:

- `rdf:Bag`: Elemek olyan csoportja, ahol nincs rendezettség és egy elem többször is előfordulhat. Üres is lehet.
- `rdf:Seq`: Egy rendezett `rdf:Bag`, ahol számít a sorrend is. Üres is lehet.
- `rdf:Alt`: Nem rendezett elemek csoportja, ahol egy elem többször is előfordulhat és az elemek egymás alternatívái. Nem lehet üres, legalább egy elemet tartalmaznia kell.

A konténerek egyes elemeit az `rdf:li` vagy az `rdf_`*n konténertag-tulajdonságokkal* tudjuk megadni, ahol *n* egy nullától nagyobb egész szám lehet.

Kollekciók

A kollekciók elemek zárt csoportosítását teszik lehetővé, vagyis egy kollekciónak csak azok az elemei, amiket explicit megadunk, míg ez a konténereknél nem áll fent. Egy kollekció tulajdonképpen egy listával egyenértékű. Az RDF a következő komponenseket biztosítja a kollekciók létrehozásához: az `rdf:List` típust, illetve az `rdf:first`, a kollekció első elemét, az `rdf:rest`, a kollekció maradék részét, valamint az `rdf:nil`, a kollekció végét jelölölő elemeket.

Az rdf:parseType="Collection" attribútum segítségével egyszerűbb formában is leírhatjuk kollekcióinkat. Annak a tulajdonságelemnek, amely tartalmazza ezt az attribútumot, megadható csomópontelemek egy halmaza, amelyek a kollekció elemeit fogják képviselni.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:event="http://purl.org/NET/c4dm/event.owl#"
    xmlns:time="http://purl.org/NET/c4dm/timeline.owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xpath10="http://example.org/REC-xpath-19991116/">

<!-- Konténer -->
<event:ZeneEvent rdf:about="http://est.hu/esemeny/3089014">
    <rdfs:label>Afterparty</rdfs:label>
    <event:time>
        <rdf:Seq>
            <rdf:li rdf:nodeID="time_1"/>
            <rdf:li rdf:nodeID="time_2"/>
            <rdf:li rdf:nodeID="time_3"/>
        </rdf:Seq>
    </event:time>
</event:ZeneEvent>

<!-- Kollekció -->
<rdf:Description rdf:about="http://www.w3.org/TR/1999/REC-xpath-
19991116">
    <xpath10:editor>
        <rdf:List>
                <rdf:first>James Clark</rdf:first>
                <rdf:rest>
                    <rdf:List>
                        <rdf:first>Steve DeRose</rdf:first>
                        <rdf:rest rdf:resource="rdf:nil"/>
                    </rdf:List>
                </rdf:rest>
        </rdf:List>
    </xpath10:editor>
</rdf:Description>

</rdf:RDF>
```

2.3.3. RDFS

Az RDFS (RDF Schema), azaz az RDF Szókészlet Leíró Nyelv a W3C ajánlása 2004 februárja óta. Az RDFS az RDF-et bővíti ki szemantikus képességekkel, a következtetésekhez szükséges háttértudás biztosítja. Vegyük például azt a kérdést, hogy ha egy személynek van gyermeke, akkor vajon kijelenthetjük-e róla, hogy ő anya? Nos, számunkra a válasz evidens,

viszont egy alkalmazás számára nem az, meg kell neki „magyarázni”, formálisan leírni, hogy mit jelent az anya fogalma, ezt a célt szolgálnák tulajdonképpen az RDFS szókészletek a szemantikus weben.

Az RDFS lehetővé teszi erőforrások csoportjainak (osztályok), illetve erőforrások közötti viszonyok (tulajdonságok) leírását, ehhez néhány alaposztályt és alaptulajdonságot definiál, amelyek segítségével létrehozhatjuk a saját egyedi szókészletünket. Mivel az RDFS az RDF-et használja a sémák leírásához, ezért az RDF/XML egy elterjed szintaxis a RDF szókészletek rögzítése folyamán is. Az RDFS saját névtérrel rendelkezik, amelyet a `http://www.w3.org/2000/01/rdf-schema#` URI-hivatkozással azonosítunk, és általában az `rdfs` névtérelőtagot társítjuk hozzá.

Az RDFS legfontosabb alapelemei:

- *`rdfs:Class`:* Az RDF osztályok ősosztálya.
- *`rdf:Property`:* Az RDF tulajdonságok ősosztálya.
- *`rdfs:domain`:* Annak a kifejezésére használatos tulajdonság (állítmány), hogy egy adott tulajdonság (alany) csak azokra az erőforrásokra alkalmazható, amelyek a megadott osztály(ok) (tárgy) példányai.
- *`rdfs:range`:* Annak a kifejezésére használatos tulajdonság (állítmány), hogy egy adott tulajdonság (alany) által felvehető értékek, csak a megadott osztály(ok) (tárgy) példányai közül kerülhetnek ki.
- *`rdf:type`:* Annak kijelentésére szolgáló tulajdonság (állítmány), hogy egy adott erőforrás (alany) a megadott osztály (tárgy) példánya. Egy erőforrás több osztálynak is lehet a példánya.
- *`rdfs:subClassOf`:* Annak a kijelentésére szolgáló tulajdonság (állítmány), hogy egy adott erőforrás (alany) a megadott osztály (tárgy) alosztálya, azaz miden példánya egyben példánya a származtató osztálynak is.
- *`rdfs:subPropertyOf:`* Annak a kijelentésére szolgáló tulajdonság, hogy egy adott tulajdonság (alany) altulajdonsága az `rdfs:subPropertyOf` értékeként megadott tulajdonságnak (tárgy), azaz minden esetben, amikor kijelentjük, hogy egy erőforrás az

altulajdonságában megfogalmazott viszonyban van egy másikkal, akkor a tulajdonság által megfogalmazott viszony is fennáll közöttük.

Az előbbiekből kitűnik, hogy RDFS típusrendszere némileg hasonlít az objektumorientált világ típusrendszerére, azonban sok tekintetben különbözik is attól. Az egyik lényeges különbség, hogy az RDFS nem osztályokat definiál és azokhoz tulajdonságokat, hanem éppen fordítva, a tulajdonságokhoz határozza meg tulajdonságkorlátozásokkal (pl. `rdfs:domain`, `rdfs:range`) azon osztályok körét, amelyek egyedei részt vehetnek az általuk megfogalmazott viszonyokban. Ennek a tulajdonság-centrikus megközelítésnek az az előnye, hogy egy adott erőforrás leírását így bárki bővítheti, ami egy lényeges alapgondolata a szemantikus web elképzelésnek.

2.3.4. OWL

Az RDFS csak néhány alapvető eszközt bocsájt rendelkezésünkre az RDF szókészletek létrehozásához, azonban gyakran nagyobb szemantikai kifejezőképességre volna szükségünk, mint amit az RDFS biztosítani tud. Ez előbb említett célra létrejöttek az RDFS-nél gazdagabb sémanyelvek is, az egyik ilyen az OWL (Web Ontology Language) nyelv, amely szintén a W3C műhelyéből származik.

Az OWL a DAML+OIL webontológia nyelv továbbfejlesztett változata, amely webontológiák tervezésére és létrehozására szolgál. Az OWL az RDF-re és az RDFS-re épül, és segítségével:

- formalizálhatjuk egy szakterület fogalmait, kapcsolatait, osztályok és tulajdonságok definiálása útján,
- egyedeket határozhatunk meg és tulajdonságaikról kijelentéseket tehetünk,
- következtetések útján plusz információkat tudunk kinyerni az osztályokról, egyedekről az OWL formális szemantikája réven.

Az OWL nyelvi képességei többek között az alábbi jelentősebb lehetőségekkel bővítik az RDF és az RDFS-nél ismertetetteket:

- kardinalitás korlátozása tulajdonságok esetén, azaz annak a kikötése, hogy egy tulajdonságnak hány különböző értéke lehet, pl. egy póknak csak 8 lába lehet,
- tranzitivitás, inverzió, szimmetria kijelentése tulajdonságoknál,
- osztályok és egyedek egyenlőségének vagy különbözőségének a kijelentése,
- halmazműveletek (metszet, unió) alkalmazhatósága osztályokra.

Az OWL három, egyre növekvő kifejezőerejű alnyelvből áll, amelyek a felhasználók eltérő igényű csoportjait hivatottak kiszolgálni, ezek a következők:

- *OWL Lite:* Leginkább osztályozási hierarchiákat, egyszerű korlátozásokat alkalmazó felhasználók számára. Az RDF egy korlátozott nézete kiterjesztésének tekinthető.
- *OWL DL:* A maximális kifejezőerőt igénylő felhasználók számára úgy, hogy minden következtetés kiszámítható és az véges időn belül be is fejeződik. Az OWL DL nevében a DL a Description Logics (Leíró Logika) kifejezésre utal, amelyen OWL nyelv alapul. Az RDF egy korlátozott nézete kiterjesztésének tekinthető.
- *OWL Full:* A maximális kifejezőerőt és az RDF szintaktikai szabadságát igénylő felhasználók számára, akiknek viszont nem létfontosságú a kiszámíthatóság garanciája. Az RDF kiterjesztésének tekinthető.

3. A szemantikus web gyakorlati alkalmazása

A szemantikus web remek elképzelés a web intelligensebbé tételéhez, de amíg nem terjed el a metaadatok RDF-ben való megadása, a webontológiák kifejlesztése és az olyan eszközök megjelenése, amelyek ezeket ki is tudják használni, addig nem sokat ér. A következőkben néhány gyakorlati módszert szeretnék bemutatni, főleg az XML-alapú rendszerekre koncentrálva ahhoz, hogy hogyan tudjuk a jelenlegi webet „szemantikusabbá” tenni.

3.1. RDF-adatok beágyazása (X)HTML-dokumentumokba

A világháló jelentős részét a HTML, XHTML nyelven megírt weboldalak alkotják, jó lenne tehát, ha fel tudnánk ruházni ezeket a dokumentumokat is RDF-adatokkal, amit aztán például a webes keresőprogramok felhasználhatnának egy hatékonyabb keresés érdekében. A legkézenfekvőbb megoldás az lenne, ha a RDF/XML-forrást beágyaznánk az a (X)HTML-dokumentum kódjába., erre láthatunk példát az alábbi kódban:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="hu"
lang="hu">
<head>
    <title>RDF beágyazása XHTML-be</title>
    <rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:dc="http://purl.org/dc/elements/1.1/">
        <rdf:Description about="http://sevent.dyndns.org">
            <dc:creator>Nádasdi Attila</dc:creator>
        </rdf:Description>
    </rdf:RDF>
</head>
<body>
</body>
</html>
```

Az előbbi megoldással viszont több probléma is akad. Az egyik probléma, hogy az így beágyazott XML-elemek tartalmát a legtöbb böngésző megjeleníti, viszont a metaadatok főként nem embereknek szánt információk, szóval ez egy nem kívánt működés. Hogy a böngésző nem jelenítse meg az RDF/XML-elemek tartalmát, alkalmazhatjuk azt a trükköt, hogy ha a tulajdonságelem értéke karakterlánc típusú literál, akkor ezt a hozzá tartozó csomópontelem attribútumaként is megadhatjuk. Az első példa átalakítva az előzőek szerint:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="hu"
lang="hu">

<head>
    <title>RDF beágyazása XHTML-be</title>
    <rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:dc="http://purl.org/dc/elements/1.1/">
        <rdf:Description rdf:about="http://sevent.dyndns.org"
            dc:creator="Nádasdi Attila"/>
    </rdf:RDF>
</head>
<body>
</body>
</html>
```

A másik probléma, hogy az így módosított dokumentumok már nem lesznek érvényes (X)HTML-dokumentumok, mivel a szabványos (X)HTML DTD-kel szembeni validálás során elbuknak, mert olyan elemeket is tartalmaznak, amelyekre ezek a sémák nincsenek felkészítve.

Van még egy jelentős probléma, az előbb bemutatott beágyazásokkal, mégpedig az, hogy ezekben az esetekben sok redundáns adatot fog tartalmazni a dokumentumunk, mivel a metaadatok és az adatok között sok esetben lesznek átfedések, pl.: megadjuk a weboldalunk készítőjének nevét egy RDF-kijelentésben, amely a feldolgozó alkalmazásoknak fog szólni, és ugyanez megjelenik mondjuk a weboldal alján is, ami a böngészőben fog megjelenni a látogatók számára. Mindez megnehezíti a dokumentumok létrehozását, karbantartását, sőt inkonzisztens állapotot is eredményezhet.

A következőkben olyan módszereket mutatnék be az eRDF és a RDFa formájában, amelyekkel a fentebb említett problémák túlnyomó többsége kiküszöbölhető.

3.1.1. eRDF

Az *eRDF (Embedded RDF)*, segítségével úgy ágyazhatunk be RDF-adatokat (X)HTML-dokumentumokba, hogy azok megtarthassák érvényességüket. Az eRDF nem támogatja a teljes RDF-szókészletet, hanem annak csak egy részhalmazát. Amik hiányoznak az eRDF-ből:

- *Üres csomópontok:* minden csomópontnak kell legyen azonosítója, ami URI vagy erőforrásrész-azonosító;
- *Konténerek:* `rdf:Bag`, `rdf:Alt`, `rdf:Seq`;
- *Implicit típusmegadás:* a típust megadni csak explicit lehet, az `rdf:type` segítségével;
- *XML literálok, tipizált literálok:* minden literál csak egyszerű lehet.

Ahhoz, hogy a dokumentumot fel lehessen ismerni eRDF-et tartalmazó dokumentumként, szükséges a dokumentum fejrészében a metaadat profil elérésének a megadása, ez a következőképpen néz ki:

```
<head profile="http://purl.org/NET/erdf/profile">
...
</head>
```

Egy dokumentum több profilt is tartalmazhat ezeket szóközzel kell elválasztani egymástól. A profil URI-jának címéről kinyerhető egy XSLT-transzfromáció, amelynek segítségével a beágyazott RDF-adatok RDF/XML-be alakíthatók.

A használni kívánt RDF-sémákat deklarálnunk kell, a XML-névterek deklarálásához hasonlóan, ezt a legcélszerűbb a dokumentum `head` részében megtennünk, a következő formában: `<link rel="schema.`*`prefix`*`" href="`*`uri`*`" />`, például:

```
<link rel="schema.dc" href="http://purl.org/dc/elements/1.1/" />
<link rel="schema.foaf" href="http://xmlns.com/foaf/0.1/" />
```

A sémában megfogalmazott tulajdonságot ezek után a sémaperfixszel minősített tulajdonság lokális nevével tudjuk hivatkozni. A prefixszet és a tulajdonságnevet "-" vagy "." jel választja el egymástól, pl.: `foaf-name` vagy `foaf.name`. A két különböző szeparátor a Dublin Core RDF-szókészlettel való kompatibilitás miatt alakult ki. A dokumentum `head` részében a pont, míg a `body` részében a kötőjel szeparátor használata ajánlott.

Az aktuális dokumentumról szóló RDF-kijelentéseket a dokumentum *head* részében célszerű elhelyezni. Az RDF-kijelentéseket két (X)HTML-elem segítségével tudjuk ábrázolni. Az egyik a `meta` elem, amely olyan hármasok ábrázolására szolgál, amelyeknél a kijelentés tárgya literál értéket tartalmaz. A tulajdonságot a `meta` elem `name` attribútumában, míg a tulajdonság értékét a `content` attribútumban tudjuk megadni, pl.:

```
<meta name="dc.creator" content="Nádasdi Attila" />
```

Ha a dokumentumunk bázis URI-ja például: `http://example.org/nadasdi_attila`, akkor ez az előbbi kijelentés az alábbi hármast eredményezi:

```
<http://example.org/nadasdi_attila>  dc:creator "Nádasdi Attila" .
```

A másik (X)HTML-elem amit használhatunk a `head` részben a `link`, amely olyan kijelentések leírására szolgál, ahol a tárgy erőforrás. A kijelentés állítmányát a `rel` attribútum értékeként tudjuk megadni, míg a `href` attribútum a tárgyat azonosító URI-t tartalmazza. A `rel` attribútumban több értéket is megadhatunk szóközzel elválasztva, ez egyfajta rövidítési lehetőséget biztosít azokban az esetekben, amikor a kijelentések tárgya ugyanaz az erőforrás. A `rev` attribútumban megadott tulajdonság(ok) esetén az alany és a tárgy szerepköre felcserélődik, vagyis a `href`-ben megadott erőforrás lesz ezen kijelentések alanya, míg a tárgya az aktuális dokumentum. Például:

```
<link href="#magamrol" rev="foaf-homepage foaf-made"
rel="foaf-maker" />
```

A fenti kijelentés az alábbi hármasokat eredményezi:

```
<http://example.org/nadasdi_attila> foaf:maker
  <http://example.org/nadasdi_attila#magamrol> .
<http://example.org/nadasdi_attila#magamrol> foaf:homepage
  <http://example.org/nadasdi_attila> .
<http://example.org/nadasdi_attila#magamrol> foaf:made
  <http://example.org/nadasdi_attila> .
```

Lehetőség van természetesen a *body* elemben is megadni kijelentéseket az aktuális dokumentumról. Bármely elem, amely nem tartalmaz `id` attribútumot, az potenciálisan az aktuális dokumentumról tesz kijelentéseket. Literál értéket tartalmazó hármasok beágyazhatók, bármely elem `class` attribútumának segítségével, ilyenkor a tulajdonság értéke a `class` attribú-

tumban adható meg. A literál értékek kétféleképpen is megadhatók: az elem `title` attribútma értékeként vagy az elem tartalmaként. Például:

```
<div>
Az oldalt <span class="dc-creator">Nádasdi Attila</span>készítette
</div>
<div>
Az oldalt <span class="dc-creator" title="Nádasdi
Attila">én</span> készítettem
</div>
```

Az előbbi mindkét kijelentés ugyanazt a hármast eredményezi:

```
<http://example.org/nadasdi_attila> dc:creator "Nádasdi Attila" .
```

Az `a` elem szolgál a dokumentum törzsében arra, hogy hogy olyan kijelentéseket fogalmazzunk meg, amelyek tárgya erőforrás. Ugyanúgy mint a `link` elemnél itt is alkalmazhatjuk a `rel`, `rev`, `href` attribútumokat, ugyanazokra a funkciókra, mint amit ott megismerhettünk, ráadásul `a` elem tartalma vagy a `title` attribútumában megadott érték implicit RDF-hármas formájában is kigenerálódik, az `rdfs:label` tulajdonság értékeként. Például:

```
<a rel="foaf-maker"
href="http://example.org/nadasdi_attila#magamrol">Magamról</a>
```

Ez a következő hármasokat generálja:

```
<http://example.org/nadasdi_attila> foaf:maker
  <http://example.org/nadasdi_attila#magamrol> .
<http://example.org/nadasdi_attila#magamrol> rdfs:label "Magamról"
.
```

Az `img` elem szintén alkalmazható olyan kijelentésekben, ahol a tárgypozícióban erőforrás található, ilyen esetben a `src` attribútummal tudjuk megadni az erőforrás URI-ját, pl.:

```
<img rel="foaf-img"
src="http://example.org/nadasdi_attila/kepek/kep1.jpg" />
```

Azokat a kijelentéseket, amelyek nem az aktuális dokumentumról szólnak, meg kell jelölnünk az *`id`* attribútum segítségével, amelynek értéke URI, vagy erőforrásrész-azonosító kell, hogy legyen. Például:

```
<p id="magamrol"><span class="foaf-name">Nádasdi Attila</span>
készítette ezt az oldalt.</p>
```

Az előző példa a következő hármast eredményezi:

```
<http://example.org/nadasdi_attila#magamrol> foaf:name
"Nádasdi Attila" .
```

Az *RDF osztályok* jelzésére a `class` attribútumban megadott osztály nevét kötőjellel kell kezdenünk, pl.:

```
<p id="magamrol" class="-foaf-Person">Személy</p>
```

amely a következő hármast jelenti:

```
<http://example.org/nadasdi_attila#magamrol> rdf:type foaf:Person .
```

3.1.2. RDFa

A W3C által kidolgozott RDFa (Resource Description Framework in attributes), olyan attribútumok és feldolgozási szabályok összessége, amelyek segítségével RDF-adatokkal lehet kibővíteni elméletileg bármely XML-alapú jelölőnyelvet (ellentétben az eRDF-fel, amely csak az XHTML, HTML nyelveket támogatja). Az XHTML 1.1 nyelv kibővítéséhez már készült ajánlás, amelyet *XHTML+RDFa*-nak neveznek, és az ilyen dokumentumokat természetesen validálni is lehet a `http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd` sémával segítségével. Az RDFa elgondolását és szintaxisát tekintve sok hasonlóságot mutat az eRDF-fel. A következőkben az XHTML+RDFa nyelvet fogom részletesebben bemutatni.

Az RDFa bevezeti a kompakt URI-k (*CURIE*) fogalmát, mert az URI-k gyakran hosszúak és nehézkes a használatuk. Egy CURIE-t az alábbi formában lehet megadni:

`[ [ prefix ] ':' ] hivatkozas`,

ahol az opcionális prefixet kettőspont választja el a hivatkozástól. Ha a prefix hiányzik akkor az alapértelmezett prefix jut érvényre, amely a `http://www.w3.org/1999/xhtml/vocab#` URI-t takarja. Egy CURIE egy teljes URI-t reprezentál. Az "_" prefix az üres csomópontot hivatott jelezni. Bizonyos helyzetekben egy attribútum értéke lehet URI és CURIE is, hogy meg

lehessen őket különböztetni a CURIE-t kapcsos zárójelekké közzé tesszük, ezt hívjuk *biztonságos CURIE-nek*.

Az RDFa a metaadatok megadásához számos már meglévő XHTML attribútumra és néhány újonnan bevezetésre kerülőre támaszkodik, ezek a következők:

Meglévő XHTML-attribútumok:

- *rel:* CURIE-k szóközzel elválasztott listája lehet az értéke, amelyek két erőforrás közötti kapcsolatot fejeznek ki, az állítmány szerepét töltik be az RDF-terminológiában;
- *rev:* CURIE-k szóközzel elválasztott listája lehet az értéke, amelyek két erőforrás közötti fordított irányú kapcsolatot fejeznek ki, az állítmány szerepét töltik be az RDF-terminológiában;
- *content:* értéke szrting literál lehet, amely egyszerű, literál értéket képviselő tárgy szerepét tölti be az RDF-terminológiában;
- *href:* értéke egy URI lehet (elérhető az interneten keresztül), amely egy kapcsolat résztvevőjét (erőforrás) azonosítja, az RDF-terminológiában erőforrást tartalmazó tárgy;
- *src:* értéke egy URI lehet, amely egy kapcsolat résztvevőjét (erőforrás) azonosítja, amikor az erőforrás be van ágyazva, az RDF-terminológiában erőforrást tartalmazó tárgy.

Új, RDFa specifikus attribútumok:

- *about:* értéke URI vagy biztonságos CURIE lehet, az alany szerepét tölti be az RDF-terminológiában;
- *property:* CURIE-k szóközzel elválasztott listája lehet az értéke, az alany és egy sztring literál közötti kapcsolatot fejezi ki, állítmány az RDF-terminológiában;
- *resource:* értéke URI vagy biztonságos CURIE lehet, erőforrást tartalmazó tárgy az RDF-terminológiában;
- *datatype:* értéke egy CURIE, amely egy adattípust reprezentál, egy tipizált literál kifejezéséhez használjuk;

- *typeof:* CURIE-k szóközzel elválasztott listája lehet az értéke, amely jelzi az alany RDF-típusát.

A szemléltetéshez lássunk egy egyszerű példát, amelyben egy Debrecenről szóló XHTML+RDFa-dokumentumot hozunk létre.

A dokumentum fejlécében megadjuk, hogy ez a dokumentum egy érvényes XHTML+RDFa dokumentum, valamint megadjuk a sémát is, amely segítségével egy alkalmazás is ellenőrizni tudja állításunk:

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
  "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
```

A következő lépésben a gyökérelemben deklaráljuk a használni kívánt névtereket. Az alapértelmezett névtér a `http://www.w3.org/1999/xhtml` névtér:

```
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:dbpp="http://dbpedia.org/property/"
  xmlns:dbpo="http://dbpedia.org/ontology/"
  xmlns:dbpr="http://dbpedia.org/resource/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
```

Következik a `head` elem, amely általában az aktuális dokumentumra vonatkozó kijelentéseinket tartalmazza, amelyet a jelen esetben `base` elem `href` attribútumában megadott URI (bázis URI) azonosít, egyébként pedig a dokumentum aktuális címe. A bázis URI segítségével, relatív URI-kat is alkalmazhatunk a dokumentumon belül. A `meta` és a `link` elemeket itt is hasonlóan kell használni, mit az eRDF-nél láthattuk, egyedül a szeparátorban van eltérés, ami itt kettőspont:

```
<head>
  <title>Debrecen</title>
  <base href="http://example.org/debrecen/" />
  <meta property="dc:creator" content="Nádasdi Attila" />
  <link rel="foaf:primaryTopic"
    href="http://hu.wikipedia.org/wiki/Debrecen" />
</head>
```

A `body` elem tartalmazza a településre vonatkozó kijelentéseket, ahol a települést a `[dbpr:Debrecen]` biztonságos CURIE azonosítja, ami egyenértékű a `http://dbpedia.org/resource/Debrecen` URI-val. Az eRDF-fel ellentétben az RDFa-ban alkalmazhatunk tipizált literálokat, a `datatype` attribútum segítségével, sőt ha eltérő formátumban kell ábrázolni a metaadatot és az adatot, mint pl. itt a terület megadásánál, akkor azt is meg lehet oldani a `content` attribútum használatával:

```
<body about="[dbpr:Debrecen]" typeof="dbpo:Place">
  <p>
    <span property="rdfs:label" xml:lang="hu">Debrecen</span>
    Magyarország második legnagyobb és legnépesebb városa,
    <a rel="dbpp:subdivisionName" rev="dbpp:seat"
      href="http://dbpedia.org/resource/Hajd%C3%BA-Bihar_County">
      Hajdú-Bihar megye
    </a> és a Debreceni kistérség székhelye.
  </p>
  <ul>
    <li>Területe:
      <span property="dbpp:areaTotalKm" content="461.25"
        datatype="xsd:double">461,25</span> km<sup>2</sup>
    </li>
    <li>Népessége:
      <span property="dbpp:populationTotal"
        datatype="xsd:integer">206225</span> fő
    </li>
  </ul>
</body>
```

Kinyerve a beágyazott RDF-adatokat, az alábbi RDF/XML-dokumentumot kapjuk:

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:dbpp="http://dbpedia.org/property/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#">

  <rdf:Description rdf:about="http://example.org/debrecen/">
    <dc:title>Debrecen</dc:title>
    <dc:creator xml:lang="hu">Nádasdi Attila</dc:creator>
    <foaf:primaryTopic
      rdf:resource="http://hu.wikipedia.org/wiki/Debrecen"/>
  </rdf:Description>

  <rdf:Description
    rdf:about="http://dbpedia.org/resource/Debrecen">
    <rdf:type rdf:resource="http://dbpedia.org/ontology/Place"/>
    <rdfs:label xml:lang="hu">Debrecen</rdfs:label>
    <dbpp:subdivisionName rdf:resource=
      "http://dbpedia.org/resource/Hajd%C3%BA-Bihar_County"/>
    <dbpp:areaTotalKm rdf:datatype="xsd:double">
      461.25
    </dbpp:areaTotalKm>
    <dbpp:populationTotal rdf:datatype="xsd:integer">
      206225
    </dbpp:populationTotal>
  </rdf:Description>

  <rdf:Description rdf:about=
    "http://dbpedia.org/resource/Hajd%C3%BA-Bihar_County">
    <dbpp:seat
      rdf:resource="http://dbpedia.org/resource/Debrecen"/>
  </rdf:Description>

</rdf:RDF>
```

3.2. RDF-források csatolása (X)HTML-dokumentumokhoz

Létezik egy, az eddigiektől eltérő megközelítése annak, hogy hogyan tegyünk közzé metaadatokat egy weboldalról, ez pedig azon alapszik, hogy ne a weboldal forrásába ágyazva kódoljuk bele az RDF-adatokat, hanem tegyük azt egy külön állományba és a weboldalból csak hivatkozzunk rá. Nos, ennek a megközelítésnek szintén vannak előnye és hátrányai is.

Előnyként lehet felsorolni, hogy az adat és a metaadat különválasztásával, könnyebbé válik a karbantartás, a weboldalunk érvényességet még annyira sem befolyásolja ez a megoldás,

mint amit a metaadatok beágyazásánál tapasztalhattunk, ráadásul a RDF-adatokat többféle formában (pl. Notation3, Turtle, N-Triples) is csatolhatjuk, nem csak RDF/XML-ben. A feldolgozó alkalmazások számára is könnyebbség, ha nem a weboldalból kell „kibányászniuk” a metaadatokat.

Hátrányként említhető meg a redundancia, ami az adatok duplikált tárolásából adódóan fog megjelenni, mivel sok átfedés lesz a megjelenítésre kerülő adatok és a metaadatok között, ami például az RDFa alkalmazása esetén nagyrészt elkerülhető.

Ha mégis a metaadatok csatolása mellett döntünk, azt legegyszerűbben a (X)HTML-dokumentumunk `head` részében elhelyezett `link` elem segítségével tehetjük meg. A `link` elem `href` attribútumában adhatjuk meg a RDF-adatokat tartalmazó állomány URI-ját, míg a `type` attribútum szolgál a média típus megadására, amely RDF/XML esetében az `"application/rdf+xml"` értéket kapja. Megadható még a `rel` attribútum is, amely az (X)HTML-dokumentum és a csatolt metaadatok közötti viszony kifejezésére használható, ennek az `"alternative"` vagy `"meta"` érték a legmegfelelőbb. A `link` elem helyett használhatjuk az `a` elemet is a dokumentum tetszőleges helyén, ugyanazokkal az opciókkal, mint amit a `link`-nél megismerhettünk. Lássunk a metaadatok csatolására egy példát is:

```
<head>
  <link rel="meta" type="application/rdf+xml"
    href="metaadat.rdf"/>
</head>
```

3.4. Mikroformátumok

A web szemantikusabbá tételében nem csak a W3C által preferált RDF és a rá épülő rétegek (RDFS, OWL) alkalmazása az egyetlen üdvözítő módszer, említést kell még tennünk a mikroformátumokról, amelyek egyszerű, nyílt adatformátumok, amelyek már létező és széles körben alkalmazott szabványokon alapulnak. Tulajdonképpen (X)HTML-minták, amelyek már meglévő (X)HTML-komponensekből épülnek fel, és ebben a tekintetben leginkább a már megismert eRDF-re hasonlítanak.

A mikroformátumok alapelvei:

- Egy adott adott probléma megoldására születek, pl.: események, vélemények, emberek leírására, tehát nem olyan általánosak mint az RDF.
- Olyan egyszerűek amennyire csak lehet.
- Elsősorban emberek, másodsorban gépek számára tervezték őket.
- Széles körben alkalmazott szabványokra épülnek (pl. HTML, XHTML, Atom, RSS).
- Modulárisak és beágyazhatóak.
- Lehetővé teszik és ösztönzik a decentralizált fejlesztést, tartalmakat és szolgáltatásokat, ami tulajdonképpen a Tim Berners-Lee által megfogalmazott szemantikus web alapgondolata is.

A jelenleg elérhető, már stabil specifikációval rendelkező mikroformátumok a következők:

- *hCalendar:* Naptárbejegyzés- és eseményformátum.
- *hCard:* Névjegykártya-formátum.
- *rel-licence:* Tartalmak licencének jelzésére szolgáló formátum.
- *rel-nofollow:* Hiperhivatkozásoknál alkalmazható formátum, amely jelzi az alkalmazások számára (pl. keresőrobot), hogy a hivatkozást, amely tartalmazza, nem érdemes követni, mert nem tartalmaz lényegi információkat.
- *rel-tag:* Hiperhivatkozásoknál alkalmazható formátum, amely jelzi, hogy a hivatkozás olyan oldalra visz, amely meg van jelölve az adott címkével.
- *VoteLinks:* Hiperhivatkozásoknál alkalmazható formátum, annak jelzésére, hogy a hivatkozásra való kattintással valaminek a támogatása mellett tesszük le a voksunkat vagy éppen ellenkezőleg, esetleg tartózkodunk a véleményezéstől.
- *XFN:* Az emberi kapcsolatok reprezentálására szolgáló formátum.
- *XMDP (XHTML Meta Data Profile):* HTML-metaadat profilok definiálására szolgáló formátum.
- *XOXO (eXtensible Open XHTML Outlines):* XHTML vázlatformátum.

Ezeken kívül számos olyan mikroformátum létezik, amelynek a specifikációja még nem teljesen kiforrott, de már használható állapotban van. Ízelítőnek megadnék egy zenei esemény leírást a hCalendar mikroformátum segítségével:

```
<div id="hcalendar-Modern-Art-Orchestra" class="vevent">

  <!-- Az esemény URL-je -->
  <a href="http://est.hu/esemeny/3075458" class="url">
    <!-- Esemény kezdete -->
    <abbr title="2010-11-28T19:00" class="dtstart">
      November 28, 2010 7
    </abbr>
      <!-- Esemény vége -->
    – <abbr title="2010-11-28T21:00"
        class="dtend">9pm</abbr>
      <!-- Az esemény rövid összefoglalója -->
    : <span class="summary">Modern Art Orchestra koncert</span> at
    <!-- Az esemény helyszíne -->
    <span class="location">Bárka, Budapest, Üllői út 82.</span>
  </a>

  <!-- Az esemény leírása -->
  <div class="description">
    A legkiválóbb fiatal muzsikusokból verbuválódott formáció a
kortárs klasszikusok és a jazz határainak áthágására, új
hangzások, stílusok létrehozására esküdt fel.
  </div>

  <!-- Az esemény címkéi -->
  <div class="tags">
    Tags: <a href="http://eventful.com/events/tags/jazz"
            rel="tag">jazz</a>
  </div>

</div>
```

3.5. GRDDL

A *GRDDL (Gleaning Resource Descriptions from Dialects of Languages, Dialektusok Forrásleírásainak Böngészése)* egy W3C ajánlás, amely egy olyan mechanizmust takar, amelynek segítségével RDF-adatokat lehet kinyerni XML-dokumentumokból. A RDF-adatok kinyerését a dokumentumhoz rendelt transzformáció (amely általában egy vagy több XSLT-stíluslapban van megadva) felhasználásával, a GRDDL-képes ügynökprogramok végzik. Leg-

gyakoribb felhasználási módja az eRDF-be, RDFa-ba vagy mikroformátumokba kódolt adatok kinyerése és RDF/XML-be való átalakítása.

Az általánosan használható megoldás, hogy egy *jólformázott XML-dokumentumokhoz* GRDDL-transzformációkat társítsunk, az a gyökérelemben elhelyezett a `grddl` névtérdeklaráció és a `grddl:transformation` attribútum megadásából áll. A `grddl:transformation` attribútum értékeként egy vagy több IRI-t[10] (Internationalized Resource Identifiers) lehet megadni szóközzel elválasztva, ezek egy futtatható scriptre vagy programra hivatkoznak, amelyek a forrásdokumentum RDF-be transzformálását végzik. Bázis URI alkalmazásával a `grddl:transformation` IRI-jei lehetnek akár relatívak is, erre láthatunk példát többek között a következő kódban:

```
<html xmlns="http://www.w3.org/1999/xhtml"
  xml:base="http://www.w3.org/2001/sw/grddl-wg/td/"
  xmlns:grddl="http://www.w3.org/2003/g/data-view#"
    grddl:transformation="glean_title.xsl">
<head>
  <title>Debrecen</title>
  <link rel="alternate"
    href="http://dbpedia.org/resource/Debrecen"/>
</head>
</html>
```

Az előző XML-dokumentum transzformációja után létrejövő RDF/XML-dokumentum:

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description
    rdf:about="http://dbpedia.org/resource/Debrecen">
    <ns0:title
      xmlns:ns0="http://purl.org/dc/elements/1.1/">
      Debrecen
    </ns0:title>
  </rdf:Description>
</rdf:RDF>
```

Az előbbi megoldással az a probléma, hogy érvényes XHTML-dokumentumokra nem tudjuk alkalmazni anélkül, hogy az érvényességet el ne veszítsük. Szerencsére HTML-specifikus komponensek segítségével át tudjuk hidalni ezt a nehézséget. A `head` elem `profil` attribútumával adhatjuk meg dokumentumunk által használt metaadat profilt/profilokat. A metaadat

10 Az IRI-k olyan nemzetöziesített erőforrás-azonosítók, amelyekben megengedettek az Unicode karakterek alkalmazása. Minden URI egyben IRI is.

profilok, olyan dokumentumok, amelyek a profilhoz tartozó tulajdonságok és az általuk felvehető értékek halmazát írják le. Ha a forrásdokumentum olyan dialektust alkalmaz, amely rendelkezik GRDDL-kompatibilis profillal (azaz a szükséges GRDDL-transzformációkat is tartalmazza), akkor elég csak a profil URI-ját megadnunk a `profile` attribútum értékeként, egy GRDDL-képes ügynökprogram követve a profil URI-ját már ki tudja nyerni a szükséges transzformációt (alkalmazza a profildokumentumban deklarált transzformációt, melynek eredményeként előáll egy RDF-kijelentés, melyben a `http://www.w3.org/2003/g/data-view#profileTransformation` tulajdonság értékeként beazonosítható az adott profillal rendelkező összes dokumentumon végrehajtandó transzformáció), amelyet alkalmazva előállítja a forrásdokumentum RDF-reprezentációját. Ilyen GRDDL-kompatibilis profillal rendelkezik például a már korábban ismertetett eRDF, melynek profildokumentuma a `http://purl.org/NET/erdf/profile` URI-hivatkozással érhető el. Részletek az eRDF profildokumentumából:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
  lang="en">
  <head profile="http://www.w3.org/2003/g/data-view">
    <title>
       Embedded RDF HTML Profile
    </title>
    ...
    <!-- A profiltranszformáció kinyerése szolgáló transzformáció
    -->
    <link rel="transformation"
      href="http://www.w3.org/2003/g/glean-profile" />
    ...
  </head>
  <body>
    ...
    <p>This following link provides the statement :
      <a
        <!-- Profiltranszformáció -->
        rel="profileTransformation"
        href="http://purl.org/NET/erdf/extract-rdf.xsl">
          extract-rdf.xsl</a>,
      <a rel="profile"
        href="http://purl.org/NET/erdf/profile">profile</a>
    </p>
    ...
  </body>
</html>
```

Az eRDF profildokumentumából kinyert RDF-kijelentés, amely tartalmazza a profiltranszformációt:

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="http://purl.org/NET/erdf/profile">
    <ns0:profileTransformation
      xmlns:ns0="http://www.w3.org/2003/g/data-view#"
      rdf:resource="http://purl.org/NET/erdf/extract-rdf.xsl"/>
  </rdf:Description>
</rdf:RDF>
```

Ha a forrásdokumentum dialektusa nem rendelkezik metaadat profillal, vagy ha rendelkezik, de az nem felel meg az igényeinknek, akkor a GRDDL profilját kell alkalmaznunk, amelyet a `http://www.w3.org/2003/g/data-view` URI-val azonosítunk, ekkor azonban meg kell adnunk egy vagy több `link` elemet is, amelyeknek `rel` attribútumában a `"transformation"` értéket, `href` attribútumában pedig a transzformációkat azonosító IRI-ket definiáljuk. Tulajdonképpen a GRDDL-profil segítségével jelentjük ki, hogy a forrásdokumentum GRDDL-metaadatokat tartalmaz és a `rel` attribútumban megadott `transformation` tuljadonság a profildokumentumban van specifikálva. Lássunk erre egy példát:

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head profile="http://www.w3.org/2003/g/data-view">
    <title>Debrecen</title>
    <link rel="transformation"
       href=
         "http://www.w3.org/2000/06/dc-extract/dc-extract.xsl"/>
  </head>
  ...
</html>
```

A transzformációkat nem csak az egyes dokumentumokhoz vagy profilokhoz társíthatjuk, hanem egész dialektusokhoz is, amelyek egy XML-névtéren osztoznak. Bármely erőforrás amely kinyerhető egy adott névtér URI-ból, *névtérdokumentumnak* nevezzük. Egy névtérdokumentum lehet egy XML-séma vagy egy RDF-séma is akár. A névtérdokumentumok társítása a GRDDL-transzformációkkal nagyon hasonló a profildokumentumoknál ismertetettekkel, azzal a különbséggel, hogy a névtérdokumentumban megadott transzformáció az összes a névtérhez tartozó dokumentumra vonatkozni fog, és ezt nem kell külön a forrásdokumentumban jelölni. A névtérdokumentum GRDDL-eredményeként előálló RDF-kijelentés tartalmazni fogja a `http://www.w3.org/2003/g/data-view#namespaceTransformation` tulajdonsá-

got, melynek értéke fogja azonosítja az adott névtérhez tartozó dokumentumokon végrehajtandó transzformációt.

4. SEvent: szemantikus programkereső

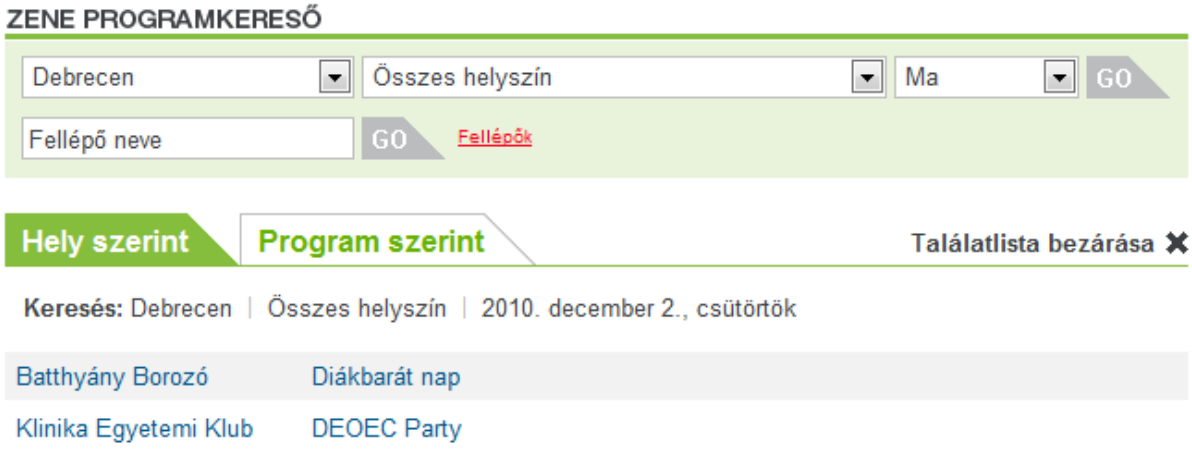
Ebben a fejezetben a dolgozathoz készített szemantikus webalkalmazást, a SEvent-et szeretném bemutatni.

A SEvent egy olyan programkereső alkalmazás, amely segítségével, az Est.hu-ról begyűjtött zenei- és sportesemények keresésére és azok sokoldalú megjelenítésére nyílik lehetőség. A kereső a http://sevent.dyndns.org címen érhető el jelenleg. Az előnyei az Est.hu keresőjével szemben:

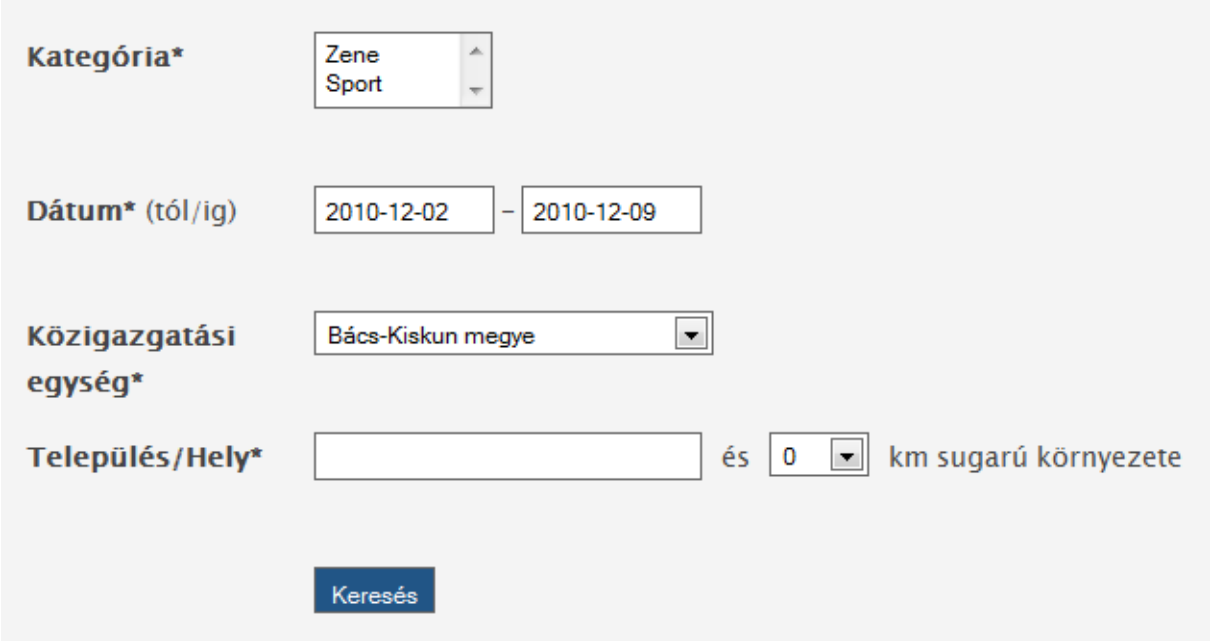
- Egy adott időintervallumot is megadhatunk a kereséskor.
- Lehetőség a távolságalapú keresésre, azaz egy adott pont (amely tetszőleges magyarországi település vagy hely lehet) és a megadott sugarú környezetében fellelhető események megkeresése.
- A találatok különböző nézetek (pl. táblázat, mozaik, térkép) szerinti megjelenítése.
- A találatok szűrési lehetőségei (idő, helyszín, kategória, település, távolság szerint), amelyeket egy kattintással vissza is vonhatunk.
- A nekünk tetsző esemény(ek) felvétele a Google Calendar-ba.

Továbbfejlesztési lehetőségek:

- A jelenlegi kategóriák bővítése (pl. mozi, színház, kiállítás).
- Az alkalmazás hosztolásának megbízhatóbb megoldása, mivel egyelőre egy asztali PC-ről üzemel.
- Összetett események kezelése.
- Az események résztvevőinek önálló erőforrásként való kezelése.



4.1. ábra: Az Est.hu zenei programkeresője



4.2. ábra: A SEvent keresője

A SEvent egy háromrétegű, kliens-szerver architektúrára épülő alkalmazás, amely a CodeIgniter[11] PHP keretrendszert, a Simile Exhibit[12] Javascript keretrendszert, valamint az Openlink Virtuoso Open-Source Edition[13] adatbázisszervert használja működéséhez. A Virtuo-

11 http://codeigniter.com/

12 http://www.simile-widgets.org/exhibit/

13 http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main

so-ról és az Exhibit-ről később részletesebben is szót ejtek, mivel fontos részét képzik az alkalmazásnak. Ezeken felül még a Geonames[14], a Google Maps[15] és a Google Calendar[16] szolgáltatásaira támaszkodik.

A Geonames egy nyílt, az egész világot lefedő földrajzi adatbázis, amely több mint 8 millió földrajzi névről tud adatokat szolgáltatni. A Geonames adatok kinyerhetők különböző webszolgáltatások segítségével, de letölthető akár az egész adatbázis is, ráadásul ami számunkra a legfontosabb, ezek adatok RDF-ben is rendelkezésre állnak. A Geonames adatokat a SEvent-ben a települések azonosítására (Geonames URI-k), a bevitt adatok validálására (pl. a megadott megyében szerepel-e az adott település), kényelmi funkciók (pl. a település nevének automatikus kiegészítése gépeléskor) megvalósítására, valamit a koordináták kinyerésére alkalmazom, ez utóbbi elengedhetetlen például a távolságalapú kereséshez.

A Google Maps szolgálatatásai a földrajzi koordináták címekből való kinyeréséhez, valamint ezek térképes megjelenítéséhez nélkülözhetetlenek. A SEvent-ben lehetőség van tetszőleges események felvételére a saját Google Naptárunkba, ehhez szintén egy Google által biztosított módszert alkalmazok.

4.1. Események adatainak begyűjtése

Ahhoz, hogy eseményekre tudjunk keresni a SEvent-ben, előtte szükség van az ehhez szükséges adatok összegyűjtésére az Est.hu-ról. A rendszer nem a keresés folyamán gyűjti össze ezeket az adatokat, mivel ez sok esetben igen időigényes művelet, hanem a rendszer üzembeállítása előtt egy RDF-adatbázis lett feltöltve ezekkel az adatokkal, és a keresések kiszolgálása aztán már innen történik. Az adatbázist természetesen rendszeresen kell frissíteni, hogy az aktuális állapotot tükrözze, erről egy ütemezett script gondoskodik.

Az adatok összegyűjtéséhez az ún. *web scraping*-nek nevezett technikát alkalmaztam, amely működése hasonló a webes keresők indexelő robotjainak működéséhez. Jelen esetben a web scraper egy olyan scriptet takar, amely a számára URL-el megadott weboldalt letölti és a forráskódjából kinyeri egy adott esemény, számunkra fontos adatait. Ez a feladat elég körül-

14 http://www.geonames.org/

15 http://code.google.com/intl/hu/apis/maps/

16 http://code.google.com/intl/hu/apis/calendar/

ményes tud lenni az oldal felépítésétől függően, ráadásul, amint módosítanak a weboldal szerkezetén, a scraper-ünk máris nem fogja a szükséges adatokat szolgáltatni, szóval csak akkor érdemes ezt a technikát alkalmazni, ha nincs más lehetőségünk az adatok megszerzésére. A fejlesztés folyamán sajnos jómagam is szembesültem ezekkel a problémákkal. Mielőtt alkalmazzuk a web scraping-et érdemes tájékozódni a kiszemelt weboldal felhasználási feltételeiről, hogy engedélyezett-e az onnan származó adatok felhasználása, mivel a szerzői jog védi ezeket a tartalmakat is.

A web scraping-hez a Simple HTML DOM nevű PHP függvénykönyvtárat alkalmaztam, amellyel viszonylag egyszerűvé válik a HTML DOM fák bejárása és a szükséges adatok kinyerése, ráadásul az érvénytelen HTML-dokumentumokkal is meg tud birkózni. Lássunk erre egy példát az `Est` adatgyűjtő osztályból:

```
// az Est keresési találata lapjainak bejárása
for ($akt_oldal = 0; $akt_oldal < $ossz_oldal; $akt_oldal++) {

    // lapozás, ha több oldalból áll a találati lista
    if ($akt_oldal != 0) {
        $params = array('dt' => $datum, 'varos' => $varos,
            'page' =>  ($akt_oldal + 1));
        $kov_oldal = $this->curl->simple_post(
            'http://est.hu/ajax.php?i=' . $kereso, $params
        );
        // új simple_html_dom példány létrehozása
        $talalatDom = new simple_html_dom();
        // a paraméterben átadott találati weboldal forrásának
        // betöltése
        $talalatDom->load($kov_oldal);
    }

    // az egyes eseményeket tartalmazó táblázat sorainak a
    // bejárása, a find első paramétere egy CSS szelektor,
    // amellyel kiválasztható a megfelelő DOM csomópont(ok)
    foreach ($talalatDom->find('tr[class="paros_sor"],
        tr[class="paratlan_sor"]') as $t_index => $tr) {
        ...
    }
  ...
}
```

Egy esemény legfontosabb összetevőit (helyszín, település, időpont) külön PHP osztályok reprezentálnak, tehát van `Esemeny`, `Helyszin`, `Idopont` és `Telepules` osztály. Mindegyik osztály rendelkezik egy `toRdf` nevű metódussal, amellyel elő lehet állítani egy speciális adat-

szerkezetet, amellyel az osztály példányait lehet RDF-kijelentések formájában reprezentálni. Ezt az *RDF/PHP*-nak nevezett formátumot, több PHP-s szemantikus web keretrendszer is alkalmazza, mint pl. az ARC[17], a Talis Platform[18] vagy az Erfurt[19], amelynek a Virtuoso szervert kezelő osztályát (`Virtuoso` osztály a SEvent-ben) én is alkalmazom a SEvent-ben. Az RDF/PHP-ba alakított adatokat a `Sevent::insertEsemeny` nevű metódusa fogja beszúrni az adatbázisba, a `Virtuoso::addMutipleStatements` metódus segítségével, amely átalakítja ezt az adatszerkezetet a megfelelő SPARQL `INSERT` utasításra:

```
// Sevent osztály
/**
 * Az eseményeket tartalamzó RDF/PHP tömb beszúrása az
 * adatbázisba.
 * @param array $esemeny az események RDF/PHP tömbje
 */
public function insertEsemeny($esemeny)
{
    // ha a modell nem létezik, akkor előbb létrehozzuk
    if (!$this->_store
        ->isModelAvailable('http://sevent.dyndns.org')) {
        $this->_store->createModel('http://sevent.dyndns.org');
    }

    // ha az $esemeny tömb, akkor beszúrjuk a
    // http://sevent.dyndns.org URI-val azonosított gráfba
    if (is_array($esemeny)) {
        $this->_store->addMultipleStatements(
            'http://sevent.dyndns.org', $esemeny
        );
    }
}
```

17 http://arc.semsol.org/

18 http://www.talis.com/platform/

19 http://aksw.org/Projects/Erfurt

```
// Virtuoso osztály
/**
 * A megadott RDF-hármasokat, amelyet az RDF/PHP tömb tartalmaz,
 * beszúrja a megadott URI-jú gráfba.
 * @param string $graphUri a gráf URI-ja
 * @param array  $statementsArray a RDF-hármasokat tartalmazó
 * RDF/PHP tömb
 */
public function addMultipleStatements($graphUri,
    array $statementsArray)
{
    $insertSparql = sprintf(
        'INSERT DATA INTO <%s> {%s}',
        $graphUri,
        //ez a metódus alakítja át az RDF/PHP tömböt N-Triples
        // formátumra
        $this->buildTripleString($statementsArray)
    );

    return $this->_execSparql($insertSparql);
}
```

Az RDF/PHP-ben megadott RDF-leírás tulajdonképpen egy többdimenziós asszociatív tömb, melyben az alanyok és az állítmányok a tömb kulcsaiként, míg a tárgyak beágyazott tömbökként reprezentálhatók. Az üres csomópontok ábrázolásához a Turtle szintaxisnál bevett jelölést használhatjuk, pl.: `_:ures_csomopont1`. A tárgyat reprezentáló tömb a következő kulcsokat tartalmazhatja:

- `type`: a tárgy típusa, értéke „uri”, „literal”, vagy „bnode” lehet (URI, literál vagy üres csomópont), kötelező megadni;
- `value`: a tárgy értéke, URI esetén meg kell adni a teljes URI-t, kötelező megadni;
- `lang`: a literál érték nyelve, opcionális;
- `datatype`: a literál érték adattípusa, opcionális.

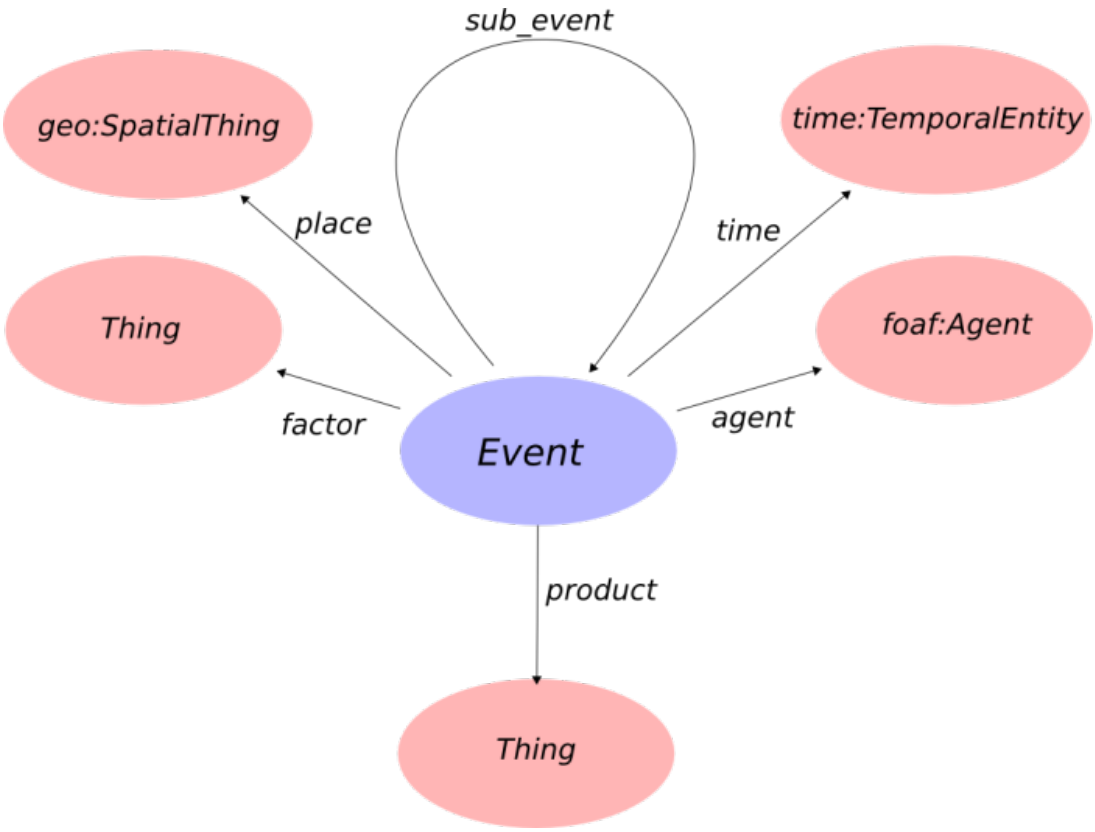
Nézzünk egy példát, amely egy eseményhez tartozó települést kódol RDF/PHP-ba és ugyanez RDF/XML-ben:

```
array(
  'http://sws.geonames.org/721472/' => array(
    'http://www.w3.org/1999/02/22-rdf-syntax-ns#type' =>
      array(array('value' => 'http://sevent.dyndns.org/Telepules',
        'type' => 'uri')),
      'http://www.geonames.org/ontology#name' =>
        array(array('value' => 'Debrecen', 'type' => 'literal'))
  )
)

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:sevent="http://sevent.dyndns.org/"
  xmlns:gn="http://www.geonames.org/ontology#">

  <rdf:Description rdf:about="http://sws.geonames.org/721472/">
    <rdf:type rdf:resource="sevent:Telepules" />
    <gn:name>Debrecen</gn:name>
  </rdf:Description>

</rdf:RDF>
```

4.2. Események modellezése RDF-ben

Az események RDF-modellje egy már létező ontológiára, az *Event ontológiára*[20] épül. A SEvent-ben az eseményeket jelen pillanatban két fő kategóriába lehet sorolni, az egyik a zenei esemény, amelynek a `ZeneEvent` nevű RDF-osztály, a másik a sportesemény, amelyeknek pedig a `SportEvent` nevű RDF-osztály feleltethető meg. Mindkét RDF-osztály az Event ontológia `Event` osztályának alosztálya. Az Evet ontológiában egy eseményt a következő gráffal ábrázolhatjuk:

20 http://motools.sourceforge.net/event/event.html



4.2. ábra: Esemény gráfja az Event ontológiában

Az Event ontológia osztályai:

- *Event:* Az eseményeket reprezentálja.
- *Factor:* Passzív tényezők (pl. eszköz, hangszer), amely kapcsolódnak az eseményekhez.
- *Product:* Bármi, amit egy esemény létrehozhat (pl. egy hang, egy gól).

Az Event ontológia fontosabb tulajdonságai:

- *agent:* Egy eseményt kapcsol össze egy aktív ügynökkel (pl. egy személlyel, egy számítógéppel).
- *factor:* Egy eseményt és egy passzív tényezőt (pl. egy eszköz, egy hangszer) kapcsol össze.
- *place:* Egy eseményt és egy földrajzi objektumot kapcsol össze.

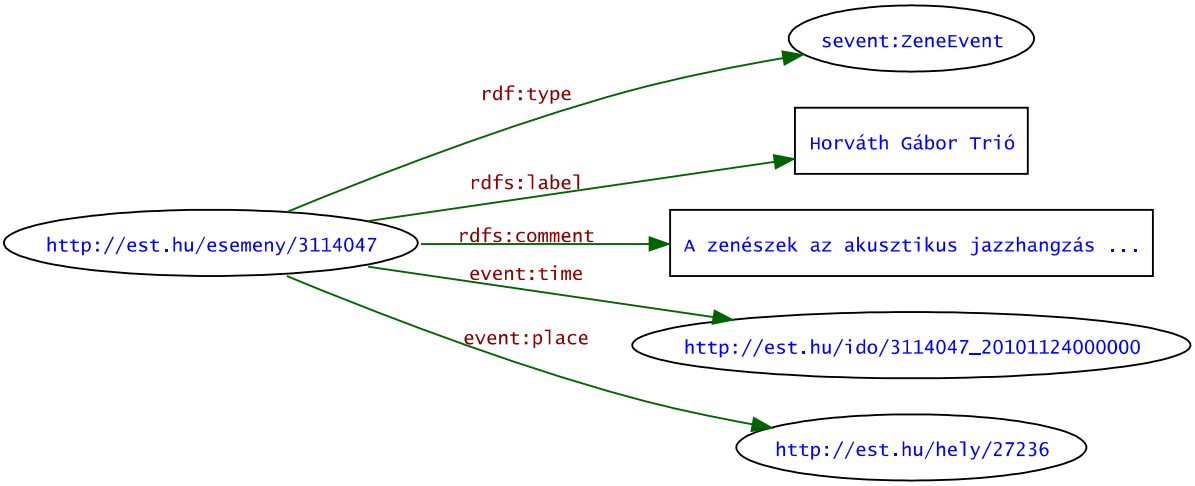
- *product:* Egy eseményt és egy az esemény folyamán létrejövő dolgot (pl. egy hang, egy gól) kapcsol össze.
- *time:* Egy eseményt és egy idő objektumot kapcsol össze.
- *sub_event:* Lehetőséget biztosít egy komplex esemény (pl. egy fesztivál, ahol több fellépő is szerepel) egyszerűbb összetevőkre bontására.

Az előbbi tulajdonságokból a SEvent-ben egyelőre csak az `event:time` és az `event:place` van használatban. A SEvent RDF-modelljét két gráf alkotja, az egyiket a `http://sevent.dyndns.org` URI azonosítja, ez tartalmazza az események adatait, míg a `http://sevent.dyndns.org/telepulesek` a Geonames-ről származó, megközelítőleg 16000 magyarországi település és hely adatait foglalja magába. Így, hogy két gráfunk van az események karbantartása, keresése is egyszerűbbé válik. Nézzünk meg egy konkrét esemény RDF/XML-leírását, valamint gráfmodelljét:

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

<!-- Névterek deklarálása -->

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:event="http://purl.org/NET/c4dm/event.owl#"
  xmlns:sevent="http://sevent.dyndns.org/"
  xmlns:gn="http://www.geonames.org/ontology#"
  xmlns:address="http://schemas.talis.com/2005/address/schema#"
  xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
  xmlns:time="http://purl.org/NET/c4dm/timeline.owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#">

  ...
```

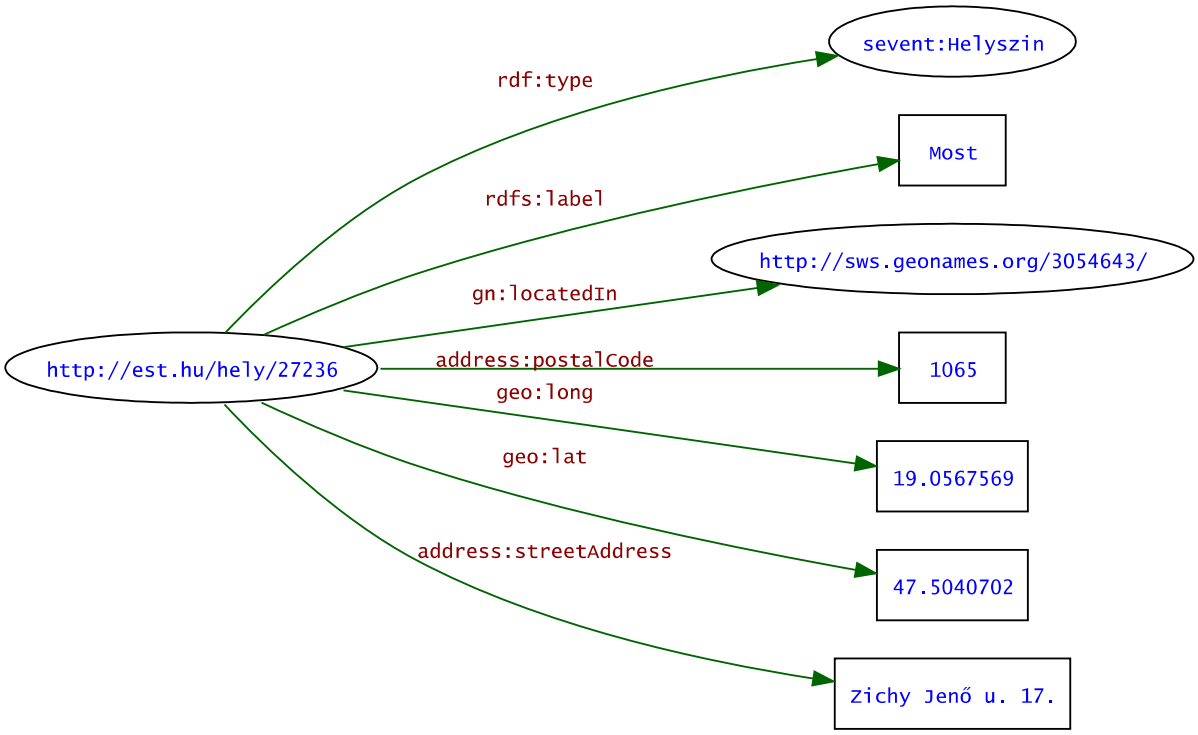
```
  ...

<!-- Esemény csomópontelem-->

<rdf:Description rdf:about="http://est.hu/esemeny/3114047">
  <rdf:type rdf:resource="sevent:ZeneEvent"/>
  <rdfs:label>Horváth Gábor Trió</rdfs:label>
  <rdfs:comment>
    A zenészek az akusztikus jazzhangzás vonalán haladva
    finoman csempészik be a különböző stílusokat (r&#39;n&#39;b,
    pop, filmzene).
  </rdfs:comment>
  <event:time
    rdf:resource="http://est.hu/ido/3114047_20101124000000" />
  <event:place rdf:resource="http://est.hu/hely/27236" />
</rdf:Description>

  ...
```

Az esemény csomópontelem gráfja:



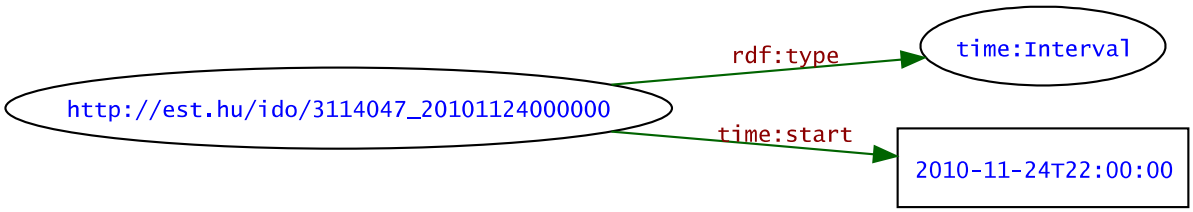
```
  ...

<!-- Helyszín csomópontelem -->

<rdf:Description rdf:about="http://est.hu/hely/27236">
  <rdf:type rdf:resource="sevent:Helyszin" />
  <rdfs:label>Most</rdfs:label>
  <!-- a település URI-ja, ahol a helyszín található -->
  <gn:locatedIn rdf:resource="http://sws.geonames.org/3054643/" />
  <address:postalCode>1065</address:postalCode>
  <geo:long rdf:datatype="xsd:double">19.0567569</geo:long>
  <geo:lat rdf:datatype="xsd:double">47.5040702</geo:lat>
  <address:streetAddress>Zichy Jenő u. 17.</address:streetAddress>
</rdf:Description>

  ...
```

A helyszín csomópontelem gráfja:



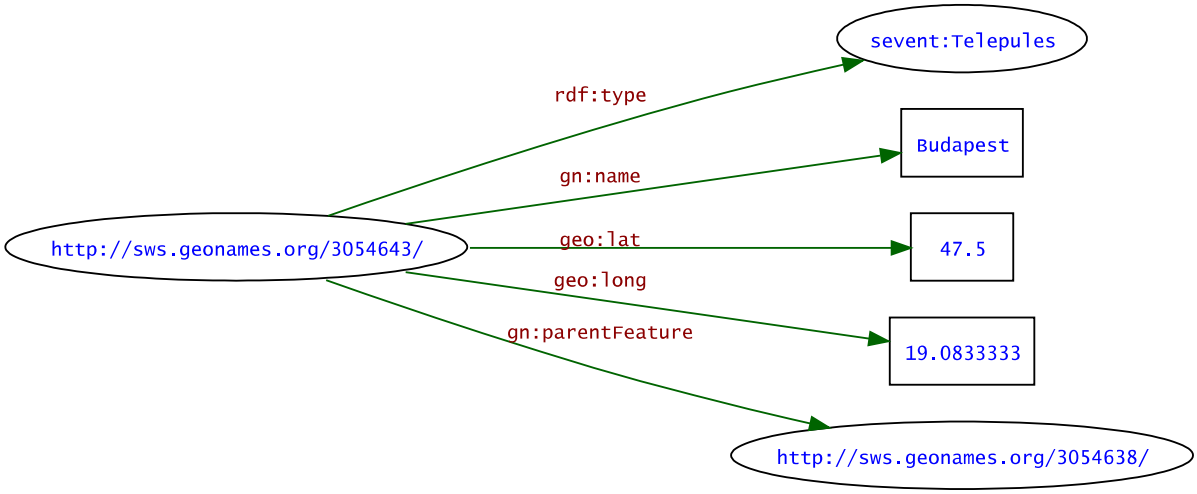
```
  ...

<!-- Időpont csomópontelem -->

<rdf:Description
rdf:about="http://est.hu/ido/3114047_20101124000000">
  <rdf:type rdf:resource="time:Interval" />
  <time:start rdf:datatype="xsd:dateTime">
    2010-11-24T22:00:00
  </time:start>
</rdf:Description>

  ...
```

Az időpont csomópontelem gráfja:




```
  ...

<!-- Település a http://sevent.dyndns.org gráfban -->

<rdf:Description rdf:about="http://sws.geonames.org/3054643/">
  <rdf:type rdf:resource="sevent:Telepules"/>
</rdf:Description>

<!-- Település a http://sevent.dyndns.org/telepulesek gráfban -->

<rdf:Description rdf:about="http://sws.geonames.org/3054643/">
  <gn:name>Budapest</gn:name>
  <geo:lat>47.5</geo:lat>
  <geo:long>19.0833333</geo:long>
  <!-- Közigazgatási egység -->
  <gn:parentFeature
    rdf:resource="http://sws.geonames.org/3054638/" />
</rdf:Description>

</rdf:RDF>
```

A település csomópontelemek gráfja:



4.3. RDF-adatok tárolása és lekérdezése

Az események adatainak perzisztens tárolására az Openlink Virtuoso szerverének nyílt forráskódú változatát alkalmazom. Virtuoso magját egy nagyteljesítményű objektumrelációs adatbázis-kezelő rendszer alkotja. A Virtuoso támogatja a tranzakciókezelést, az SQL-t, a tárolt eljárások alkalmazását, illetve számos adatelérési interfészt biztosít, pl. ODBC, JDBC, ADO.NET, OLE/DB. A PHP webalkalmazásoknál igen gyakran alkalmazott MySQL adatbázis-kezelő rendszerrel szemben, a Virtuoso számos előnnyel rendelkezik egy szemantikus webalkalmazás esetén. Az egyik legnagyobb előnye, hogy natívan támogatja az RDF-et, valamint a SQL-be ágyazott SPARQL lekérdezéseket, így az RDF-adatok tárolása és lekérdezése is jóval gyorsabb, ráadásul beépített webszerverrel is rendelkezik, így képes a dinamikus weboldalak kiszolgálására, amelyek íródhatnak VSP (Virtuoso Server Pages), PHP, ASP vagy egyéb nyelveken. Ezen kívül a Virtuoso-nál SPARQL végpontot is be lehet állítani, aminek segítségével HTTP kapcsolaton keresztül is lekérdezhetünk adatokat az RDF-tárolóból, a legkülönbözőbb szerializált formákban (pl. RDF/XML, N-Triples, JSON, XML).

A SEvent-ben ODBC-n (Open DataBase Connectivity) keresztül kapcsolódunk és kezeljük a Virtuoso szervert, ennek a támogatását egy külön osztály biztosítja, amelyet az Erfurt szemantikus keretrendszerből vettem át, némi módosítással. Lássunk ebből az osztályból némi ízelítőt:

```
class Virtuoso extends Controller
{
    // az adatbáziskapcsolat tárolása
    protected $_connection;
    protected $_graphs;
    // az osztálypéldány tárolása
    static $instance = false;

    private function Virtuoso()
    {
        parent::Controller();
        $this->_connect('VOS', 'dba', 'dba');
    }

    // A példányosítás csak ezzel a metódussal lehetséges, mivel
    // a konstruktor kívülről nem elérhető
    // a Virtuoso osztálynak csak egy példánya létezhet
    // (singleton minta)
    public function getInstance()
    {
        if (!Virtuoso::$instance) {
            Virtuoso::$instance = new Virtuoso();
        }
        return Virtuoso::$instance;
    }

    // klónozni sem lehet kívülről a példányt
    private function __clone() {}

    // a példány megszünésekor az adatbáziskapcsolat is lezárodik
    public function __destruct()
    {
        // kapcsolat bezárása
        odbc_close($this->_connection);
    }

    // Kapcsolódás a Virtuoso szerverhez ODBC-n keresztül
    protected function _connect($dsn, $user, $pass)
    {
        $this->_connection = odbc_connect($dsn, $user, $pass);

        if ($this->_connection === false) {
            throw new Exception(
                'Nem lehet kapcsolódni a Virtuoso szerverhez!'
            );
        }
    }

    ...
```

```
/**
 * Belső segédmetódus, amely ODBC-n keresztül, közvetlenül lefuttatja a
 * lekérdezéseket.
 * @param string $query a SPARQL lekérdezés
 * @param string $graphUri melyik gráfon fusson le a lekérdezés,
 *        NULL esetén a lekérdezésből nyeri ki (ha meg van adva)
 * @return az ODBC eredmény-azonosító
 * @throws Exception ha eredménytelen a lekérdezés
 */
protected function _execSparql($query, $graphUri = 'NULL')
{
    if ($graphUri !== 'NULL') {
        $graphUri = '\'' . $graphUri . '\'';
    }

    // aposztrófok, idézőjelek levédése
    $query = addslashes($query);

    // DB.DBA.SPARQL_EVAL a Virtuoso beépített függvénye a SPARQL
    // lefuttatásához, paraméterei: 1. SPARQL lekérdezés,
    // 2. a gráf URI-ja, amelyiken le kell futtatni,
    // 3. max. hány sort adhat vissza eredményként, ha 0, akkor nincs
    // korlátozva
    $resultId = odbc_exec($this->_connection,
        'CALL DB.DBA.SPARQL_EVAL(\'' . $query . '\', '
            . $graphUri . ', 0)');

    if ($resultId === false) {
        throw new Exception($this->getLastError());
    }

    return $resultId;
}

/**
 * Kívülről is elérhető metódus a SPARQL lekérdezések futtatásához.
 * @param string $query a SPARQL lekérdezés
 * @param string $graphUri melyik gráfon fusson le a lekérdezés,
 *        felülírja a lekérdezésben megadottat
 * @param string $resultFormat a lekérdezés eredményének formátuma,
 *        értéke "array" vagy "json"
 * @return array|JSON a lekérdezés eredménye, formátuma a $resultFormat
 *         értékétől függ
 */
public function sparqlQuery($query,
    $graphUri = 'http://sevent.dyndns.org', $resultFormat = 'array')
{
    $rid = $this->_execSparql($query, $graphUri);

    if ($resultFormat == 'array') {
        return $this->_odbcResultToArray($rid);
    } elseif ($resultFormat == 'json') {
        return $this->_odbcResultToJson($rid);
    }
}
```

4.4. Felhasználói felület (Simile Exhibit)

A keresés találatainak a megjelenítése a Simile Exhibit Javascript keretrendszer segítségével történik. A Simile projekt a MIT (Massachusetts Institute of Technology) nyílt forráskódú kezdeményezése, amely főképp az adatmegjelenítésre alkalmas webes widgetek[21] fejlesztését tűzte ki célul. Az Exhibit-tel olyan weboldalakat hozhatunk létre, amelyek az adatok rendezését, szűrését, különböző nézetek szerinti megjelenítését támogatják, ráadásul mindezt akár adatbázis használata nélkül.

Ahhoz, hogy elkezdhessük weboldalunkon az Exhibit-et használni, csatolnunk kell az a Exhibit API-t, valamint a megjelenítésre kerülő adatokat, amelyeket egy speciális szerkezetű JSON (JavaScript Object Notation) állomány tartalmaz. Ez a következőképpen néz ki:

```
<!-- Exhibit adatok csatolása -->
<link href="exhibit-data.js" type="application/json"
rel="exhibit/data" />

<!-- Exhibit API csatolása -->
<script src="http://api.simile-widgets.org/exhibit/2.2.0/exhibit-
api.js" type="text/javascript">
</script>

<!-- Exhibit Map bővítmény csatolása -->
<script src="http://api.simile-
widgets.org/exhibit/2.2.0/extensions/map/map-extension.js?
gmapkey=abc" type="text/javascript">
</script>
```

4.4.1. Az Exhibit adatmodellje

Az Exhibit az adatok leírásához egy saját adatmodellt definiál, amely a következő összetevőkből állhat:

- *Items (tételek):* Egy relációs adatbázis rekordjaihoz hasonlóak, egy tétel reprezentálhat bármit, pl. egy személyt, egy objektumot, egy fogalmat. Minden tételnek egyéni azo-

21 Olyan kisebb alkalmazások, amelyek weboldalakba ágyazhatók és onnan futtathatók, jellemzően Javascript vagy Adobe Flash nyelven íródtak.

nosítója (ID-ja) van, ami egy sztring. Ezen kívül a tételek rendelkeznek címkével (label) is, amely a tétel weboldalon való megjelenítésekor fog kiíródni. Az ID-nál és a címkénél is fennáll, ha nem adjuk meg explicit őket egy tételnél, az Exhibit akkor is rendel hozzájuk.

- *Types (típusok):* Minden tételnek van típusa is, ha nem adunk típust egy tételnek, akkor annak a típusa alapértelmezés szerint „Item” lesz. A típusok létrehozására egyébként nincs korlátozás. A típusoknak szintén van címkéjük.
- *Properties (tulajdonságok):* Minden tétel rendelkezhet tulajdonságokkal. Pl. egy „esemény” tétel rendelkezhet egy „helyszín” tulajdonsággal. A tulajdonságok a következő típusú értékeket vehetik fel: text (pl. „alma”), number (pl. 67.5), date (pl. 2010-11-15), boolean (true|false), url (pl. http://example.org), item (az ID-jával tudjuk hivatkozni).

Lássunk egy példát az Exhibit JSON adatmodellre:

```
{
    "items":[{
        "id":"http://est.hu/esemeny/3144789",
        "label":"Adventi jótékonysági koncert",
        "type":"Zene",
        "helyszin":"http://est.hu/hely/32985",
        "ido":"http://est.hu/ido/3144789_20101211060000",
    }],
    "properties":{
        "helyszin":{
            "uri":"http://purl.org/NET/c4dm/event.owl#place",
            "label":"helyszíne",
            "valueType":"item"
        },
        "ido":{
            "uri":"http://purl.org/NET/c4dm/event.owl#time",
            "label":"időpontja",
            "valueType":"item"
        }
    },
    "types":{
        "Zene":{
            "pluralLabel":"Zenei események"
        },
        "Sport":{
            "pluralLabel":"Sportesemények"
        }
    }
}
```

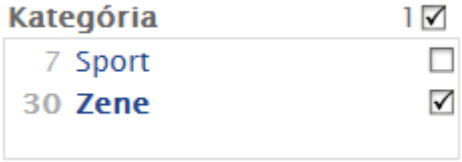
Lehetőségünk van egyébként nem Exhibit JSON formátumú adatállományok futási időben történő átkonvertálására is bizonyos típusú állományokból (pl. Excel, RDF/XML) a Babel[22] nevezetű webszolgáltatás segítségével, bár ez a megoldás sokszor jóval lassabb, mintha saját kóddal oldanánk meg a konvertálást, ráadásul nagyon gyakran nem is elérhető ez a szolgáltatás. A Babel használatakor mindössze a `link` elem `type` attribútumát kell az adatállománynak megfelelő MIME-típusra állítani, ha az eltér a „exhibit/data” értéktől, akkor az Exhibit automatikusan megpróbálja a Babel-el átkonvertálni.

4.4.2. Tételek szűrése (Facets)

Az tételek szűrését ún. *facet*-nek nevezett elemek segítségével tudjuk megtenni. A facet a felületen egy szűrődoboz formájában jelenik meg, és a beállított tulajdonság értékeit tartalmazza. Kijelölve egy vagy több elemet a facet-ből, csak azok az tételek fognak megjelenítésre kerülni, amelyek rendelkeznek ezen tulajdonsághoz tartozó értékekkel. Egy facet kódja így nézhet ki:

```
<div ex:role="collection" id="esemeny"
    ex:itemTypes="Zene, Sport"></div>

<div ex:role="facet" ex:expression=".type"
    ex:facetLabel="Kategória" ex:scroll="true"
    ex:collectionID="esemeny">
</div>
```

Az Exhibit által használt attribútumokat az `ex` prefixszel kell minősíteni. A fenti kódban deklaráltunk egy kollekciót is, amely segítségével egy közös néven („esemeny”) tudunk hivatkozni a megadott típusú („Zene”, „Sport”) tételekre. Egy facet-et az `ex:role="facet"` attribútum beállításával tudunk megadni. Az `ex:expression` értékeként beállíthatjuk, hogy milyen tulajdonság értékeire szűrjön a facet (a példában ez az „esemeny” kollekció `type` tulajdonsága, melynek értéke „Zene” vagy „Sport” lehet). Facet példa:



22 http://simile.mit.edu/babel/

Az előbb bemutatott facet az alapértelmezett ún. lista facet, ettől eltérő facet-eknél meg kell adnunk az `ex:facetClass` értékeként a facet típusát, pl. `TextSearch`, `NumericRange`.

4.4.3. Nézetek (Views)

A nézetek az item-ek kollekciójának megjelenítési módjai. A SEvent-ben háromféle nézetet alkalmazok, ezek a táblázat, a mozaik és a térkép nézet. A *táblázat nézet (tabular view)*, ahogy neve is mutatja, az item-eket egy táblázatos formában jeleníti meg, amelyet az `ex:viewClass="Tabular"` attribútum beállításával tudunk deklarálni. Beállíthatjuk az oszlopokban milyen tulajdonságok értékei jelenjenek meg, illetve az oszlopok fejlécére kattintva rendezhetjük is a sorokat. A SEvent-ben a táblázat nézet egy rövid, tömör, csak a legfontosabb információkat (esemény neve, helyszíne, időpontja) tartalmazó áttekintést nyújt az eseményekről.

TÁBLÁZAT NÉZET · MOZAIK NÉZET · TÉRKÉP NÉZET

3 Zenei esemény

Esemény	Helyszín	Időpont
Diákbarát nap	Batthyány Borozó	2010-12-06 00:00:00
School Mondays	Cool Music and Dance Club	2010-12-06 00:00:00
Szentpéteri Csilla, Zséda	Református Nagytemplom	2010-12-06 19:00:00

```
<div ex:role="view" ex:viewClass="Tabular"
  <!-- az oszlopokban megjelenő tulajdonságok -->
  ex:collectionID="esemeny" ex:columns=".label, .helyszin, .ido"
  <!-- a sorok egyedi stílusának beállítása -->
  ex:rowStyler="zebraStyler"
  ex:border="0"
  <!-- dátumformátum beállítása -->
  ex:formats="date {template: 'yyyy.MM.dd. HH:mm'}
    <!-- szeparátorok beállítása -->
    list {separator: ''; pair-separator: ''; last-separator: ''}"
  ex:cellSpacing="0"
  <!-- az oszlopfejlécek beállítása -->
  ex:columnLabels="Esemény, Helyszín, Időpont">
</div>
```

4.3. ábra: Táblázat nézet és Exhibit-kódja

A következő a *mozaik nézet (tile view)*, amely az alapértelmezett nézet is egyben az Exhibit-ben, tehát ennél nem kötelező megadni a `ex:viewClass` attribútumot. Ez már egy részletesebb információkkal szolgáló nézet, ahol szintén lehetőségünk van az események rendezésére, sőt csoportosítására is.


```
<div ex:role="view"
  <!-- csoportosítás tiltása -->
  ex:grouped="false"
  <!-- lapozás engedélyezése -->
  ex:paginate="true"
  ex:collectionID="esemeny"
  <!-- a tulajdonságok, amelyek alapján rendezhetünk -->
  ex:possibleOrders=".label, .type, .helyszin"
  <!-- szeparátorok beállítása -->
  ex:formats="list {separator: ''; pair-separator: '';
    last-separator: ''}">
</div>
```

4.4. ábra: Mozaik nézet és Exhibit-kódja

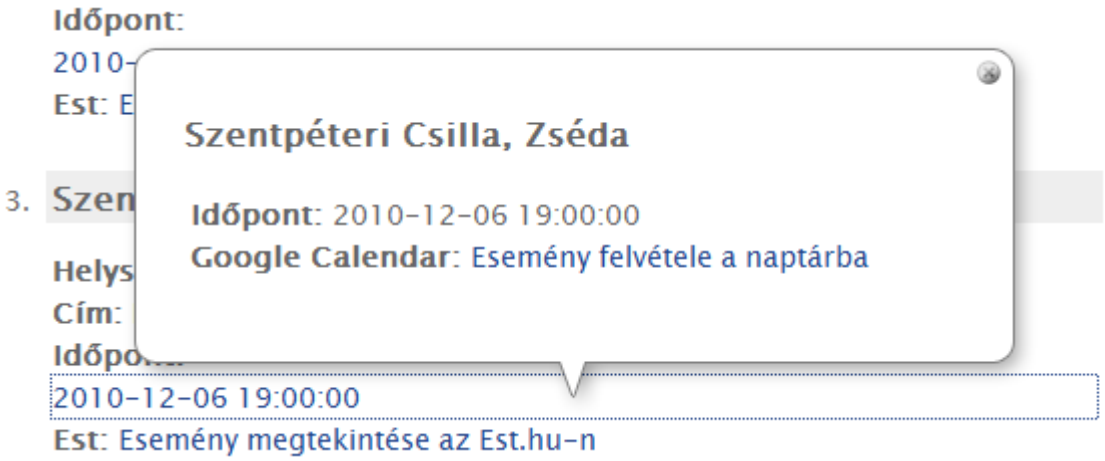
Az utolsó nézet, amit alkalmazok a SEvent-ben az a *térkép nézet (map view)*. A térkép nézet segítségével földrajzilag is beazonosíthatóvá válnak az események helyszínei. A különböző kategóriába tartozó eseményeket a rendszer, különböző színű markerekkel jelöli a térképen. Ha egy helyszínen a kiválasztott időintervallumban több esemény is megrendezésre kerül, akkor a markerben az események száma is megjelenik, amelyre kattintva bővebb információkat is megtudhatunk a programokról.


```
<div ex:role="view" ex:viewClass="Map"
  <!-- a koordinátákat szolgáltató tulajdonságok -->
  ex:latlng="concat(.helyszin.lat, ',', .helyszin.lng)"
  <!-- a markerek színét befolyásoló tulajdonság -->
  ex:colorKey=".type"
  <!-- a térkép alapértelmezett középpontja, nagyítása és
    magassága -->
  ex:center="47.16,19.55" ex:zoom="7" ex:mapHeight="660"
  ex:collectionID="esemeny">
</div>
```

4.5. ábra: Térkép nézet és Exhibit-kódja

4.4.4. Tételek egyedi megjelenítése (Lenses)

Gyakran lehet szükségünk a tételek egyedi módon való megjelenítésére, mert például az alapértelmezett nem felel meg az igényeinknek, erre a feladatra az ún. *lens*-ek kínálnak megoldást. Egy lens-t az `ex:role="lens"` attribútum megadásával tudjuk deklarálni.




```
<div ex:role="lens" class="lens"
  <!-- az item típusa amely a lens-ben meg fog jelenni -->
  ex:itemType="Ido">
  <!-- az ido kapcsolat ellenkező irányában résztvevő item
    (esemény) label tulajdonsága lesz az elem tartalma -->
  <h3 ex:content="!ido.label"></h3>
  <ul>
    <li><b>Időpont: </b><span ex:content=".label"></span></li>
    <li><b>Google Calendar: </b>
      <!-- a ex:href-content segítségével dinamikusan tudjuk a
        hivatkozás URL-jét beállítani -->
      <a target="_blank" ex:href-content=".gcal" >
        Esemény felvétele a naptárba</a>
    </li>
  </ul>
</div>
```

4.6. ábra: Lens és Exhibit-kódja

5. Összefoglalás

Dolgozatomban igyekeztem átfogó képet adni a szemantikus web elképzelésről, valamint annak gyakorlati alkalmazási lehetőségeiről. Az első részben bemutattam a szemantikus web legfontosabb rétegeit, az XML, RDF, RDFS és OWL nyelveket, különös hangsúlyt fektetve a XML-re és annak szerepére a rá épülő rétegekben. A 2.1 ábrán bemutatott szemantikus web architektúrája azonban koránt sem befejezett, sok felsőbb réteg (pl. egyesítő logika, bizonyítás, bizalom) még csak kutatás fázisában létezik, de úgy gondolom ezen rétegek megvalósításánál is jelentős szerepet fog játszani az XML. Az XML egy univerzális nyelv már ma is az alkalmazások közötti adatcsere területén, a szemantikus webnél ez különösen hangsúlyos, mivel ezzel éppen a weben lévő adatok gépi feldolgozhatóságát próbáljuk elősegíteni, ezért remek elképzelés volt a szemantikus web architektúráját az XML-re építeni.

Az XML egy csatlakozási pont is a jelenlegi és a szemantikus web között. Hogy a már meglevő, jól működő webalkalmazásainkat ne kelljen kidobnunk a szemantikus web kedvéért, olyan technológiákra van szükség, amelyek megkönnyítik a két világ közötti átjárást. Ilyen eszközt én is bemutattam a dolgozatom második részében, ez volt a GRDDL, amely egy viszonylag egyszerűen alkalmazható mechanizmus a RDF-adatok kinyerésére XML-dokumentumokból. De említhetném még a XSPARQL[23] nevű lekérdezőnyelvet, amely egyelőre még csak kezdeti stádiumában létezik, de olyan eszközt ad kezünkbe, amellyel XML-adatokat tudunk kinyerni RDF-forrásokból és fordítva, egy viszonylag egyszerű lekérdezés formájában. Az ilyen és ezekhez hasonló technológiáknak fontos szerepük lesz a szemantikus web jobb elterjedtségének a növelésében, mert megkönnyítik a már meglevő adataink, alkalmazásaink újrahasznosítását.

Végül az utolsó részben bemutattam a SEvent-et, amely annak az illusztrálására volt példa, hogy hogyan lehet egy már létező, nem szemantikus webalkalmazásból (Est.hu) kinyerni

23 http://xsparql.deri.org/

adatokat és átültetni azokat szemantikus web alapokra, valamint kombinálni azokat más forrásokból származó RDF-adatokkal. A fejlesztés során sok buktatóval szembesültem, de úgy érzem sokat tanultam a szemantikus web gyakorlati alkalmazásáról ennek folyamán. Bár még számos lehetőség rejlik a SEvent továbbfejlesztésében, úgy érzem a kezdeti célt sikerült elérnem.

Irodalomjegyzék

[1] Szeredi Péter, Lukácsy Gergely, Benkő Tamás: *A szemantikus világháló elmélete és gyakorlata.* TypoTeX, Budapest, 2005.

[2] Gottdank Tibor: *Szemantikus web: bevezetés a tudásalapú internet világába*. ComputerBooks, Budapest, 2005.

[3] Michael J. Young: *XML lépésről lépésre.* Szak Kiadó, Bicske, 2002.

[4] Jeff T. Pollock: *Semantic Web For Dummies.* Wiley Publishing, Inc., Indianapolis, Indiana, 2009.

[5] Grigorius Antoniou, Frank van Harmelen: *A Semantic Web Primer.* The MIT Press, Cambridge, Massachusetts, 2004.

[6] H. Peter Alesso, Craig F. Smith: *Thinking on the Web: Berners-Lee, Gödel, and Turing.* Wiley-Interscience, New Jersey, 2006.

[7] Herman Iván: *Áttekintés a Szemantikus Webről.* 2004. 02. 26. http://www.w3.org/2004/Talks/0226-Budapest-IH/. Letöltés dátuma: 2010. 10. 10.

[8] Herman Iván: *Szemantikus Web: egy rövid bevezetés.* 2006. 03. 18. http://www.w3.org/2006/Talks/0318-Budapest-IH/Overview.html. Letöltés dátuma: 2010. 10. 10.

[9] Herendy Csilla: *A kereső, a dokumentumok és a user. A szemantikus web egy lehetséges nézőpontja.* 2010 tavasz. http://www.mediakutato.hu/cikk/2010_01_tavasz/03_szemantikus_web. Letöltés dátuma: 2010. 10. 13.

[10] Jeszenszky Péter: *XML alkalmazások.* 2010. 08. 25. http://www.inf.unideb.hu/~jeszy/download/xml/xml-apps-2x2.pdf. Letöltés dátuma: 2010. 10. 24.

[11] Jeszenszky Péter: *Az XML 1.0 ajánlás.* 2009. 11. 10. http://www.inf.unideb.hu/~jeszy/download/xml/xml-spec-2x2.pdf. Letöltés dátuma: 2010. 10. 24.

[12] Jeszenszky Péter: *XML névterek.* 2010. 10. 04. http://www.inf.unideb.hu/~jeszy/download/xml/xml-namespaces-2x2.pdf. Letöltés dátuma: 2010. 10. 24.

[13] Jeszenszky Péter: *XPath 1.0.* 2009. 11. 10. http://www.inf.unideb.hu/~jeszy/download/xml/xpath-2x2.pdf. Letöltés dátuma: 2010. 10. 24.

[14] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, John Cowan (szerk.): *Extensible Markup Language (XML) 1.1 (Second Edition).* W3C Recommendation. 2006. 09. 29. http://www.w3.org/TR/xml11/. Letöltés dátuma: 2010. 10. 28.

[15] James Clark, Steve DeRose (szerk.): *XML Path Language (XPath) Version 1.0*. W3C Recommendation. 1999. 11. 16. http://www.w3.org/TR/xpath/. Letöltés dátuma: 2010. 11. 13.

[16] Anders Berglund, Scott Boag, Don Chamberlin, Mary F. Fernández, Michael Kay, Jonathan Robie, Jérôme Siméon (szerk.): *XML Path Language (XPath) 2.0*. W3C Recommendation. 2007. 01. 23. http://www.w3.org/TR/xpath20/. Letöltés dátuma: 2010. 11. 13.

[17] James Clark (szerk.): *XSL Transformations (XSLT) Version 1.0.* W3C Recommendation. 1999. 11. 16. http://www.w3.org/TR/xslt. Letöltés dátuma: 2010. 11. 13.

[18] Micael Key (szerk.): *XSL Transformations (XSLT) Version 2.0.* W3C Recommendation. 2007. 01. 23. http://www.w3.org/TR/xslt20/. Letöltés dátuma: 2010. 11. 13.

[19] Wikipédia: *XML.* 2010. 04. 12. http://hu.wikipedia.org/wiki/XML. Letöltés dátuma: 2010. 11. 05.

[20] Jeff Heflin (szerk.): *Az OWL Web Ontológia Nyelv. Alkalmazási esetek és követelmények.* W3C ajánlás. 2005. 04. 25. http://www.w3c.hu/forditasok/OWL/REC-webont-req-20040210.html. Letöltés dátuma: 2010. 11. 04.

[21] Wikipedia: *Semantic Web Stack.* 2010. 01. 20. http://en.wikipedia.org/wiki/Semantic_Web_Stack. Letöltés dátuma: 2010. 11. 05.

[22] Berners-Lee, Tim; Hendler, James; Lassila, Ora: *The Semantic Web.* Scientific American. 2001. 05. 17. http://www-sop.inria.fr/acacia/personnel/Fabien.Gandon/lecture/licence_travaux_etude2002/TheSemanticWeb/. Letöltés dátuma: 2010. 10. 11.

[23] Wikipedia: *Ontology (information science).* 2010. 10. 12. http://en.wikipedia.org/wiki/Ontology_%28information_science%29. Letöltés dátuma: 2010. 11. 13.

[24] Wikipedia: *Ontology language.* 2010. 02. 19. http://en.wikipedia.org/wiki/Ontology_language. Letöltés dátuma: 2010. 11. 13.

[25] W3CSchools: *XSLT Tutorial.* http://www.w3schools.com/xsl/. Letöltés dátuma: 2010. 11. 04.

[26] Frank Manola, Eric Miller (szerk.): *Az RDF bevezető tankönyve.* W3C Ajánlás. 2005. 04. 25. http://www.w3c.hu/forditasok/RDF/REC-rdf-primer-20040210.html. Letöltés dátuma: 2010. 11. 17.

[27] Jeszenszky Péter: *Resource Description Framework (RDF).* 2010. 09. 16. http://www.inf.unideb.hu/~jeszy/download/semweb/RDF-2x2.pdf. Letöltés dátuma: 2010. 10. 10.

[28] Dave Beckett (szerk.): *Az RDF/XML szintaxis specifikációja (átdolgozott kiadás).* W3C ajánlás. 2005. 04. 25. http://www.w3c.hu/forditasok/RDF/REC-rdf-syntax-grammar-20040210.html. Letöltés dátuma: 2010. 11. 12.

[29] Dan Brickely, R.V. Guha (szerk.): *Az RDF Szókészlet Leíró Nyelv 1.0: RDF Séma.* W3C ajánlás. 2005. 04. 25. http://www.w3c.hu/forditasok/RDF/REC-rdf-schema-20040210.html. Letöltés dátuma: 2010. 11. 21.

[30] Jeszenszky Péter: *RDF Séma: az RDF szókészlet leíró nyelve*. 2010. 09. 16. http://www.inf.unideb.hu/~jeszy/download/semweb/RDF-Schema-2x2.pdf. Letöltés dátuma: 2010. 10. 10.

[31] Deborah L. McGuinness, Frank van Harmelen (szerk.): *Az OWL Web Ontológia Nyelv – Áttekintés.* 2005. 04. 25. http://www.w3c.hu/forditasok/OWL/REC-owl-features-20040210.html. Letöltés dátuma: 2010. 11. 13.

[32] Michael K. Smith, Chris Welty, Deborah L. McGuinness: *Az OWL Web Ontológia Nyelv – Útmutató.* 2005. 04. 25. http://www.w3c.hu/forditasok/OWL/REC-owl-guide-20040210.html. Letöltés dátuma: 2010. 11. 13.

[33] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, Lynn Andrea Stein: *Az OWL Web Ontológia Nyelv – Referencia.* 2005. 04. 25. http://www.w3c.hu/forditasok/OWL/REC-owl-ref-20040210.html. Letöltés dátuma: 2010. 11. 13.

[34] Talis: *Embedded RDF Wiki.* 2006. 10. 23. http://research.talis.com/2005/erdf/wiki/Main/RdfInHtml. Letöltés dátuma: 2010. 11. 25.

[36] Ben Adida, Mark Birbeck (szerk.): *RDFa Primer.* W3C Working Group Note. 2008. 10. 14. http://www.w3.org/TR/xhtml-rdfa-primer/. Letöltés ideje: 2010. 11. 26.

[37] Ben Adida, Mark Birbeck, Shane McCarron, Steven Pemberton (szerk.): *RDFa in XHTML: Syntax and Processing.* W3C Recommendation. 2008. 10. 14. http://www.w3.org/TR/rdfa-syntax/. Letöltés dátuma: 2010. 11. 26.

[38] Microformats: *About Microformats.* http://microformats.org/about. Letöltés dátuma: 2010. 11. 28.

[40] Microformats: *Microformats Wiki.* 2010. 10. 20. http://microformats.org/wiki/Main_Page. Letöltés dátuma: 2010. 10. 28.

[41] Dan Conolly (szerk.): *Gleaning Resource Descriptions from Dialects of Languages (GRDDL).* W3C Recommendation. 2007. 09. 11. http://www.w3.org/TR/grddl/. Letöltés dátuma: 2010. 11. 29.

[42] Harry Halpin, Ian Davis (szerk.): *GRDDL Primer.* W3C Working Group Note. 2007. 06. 28. http://www.w3.org/TR/grddl-primer/. Letöltés dátuma: 2010. 11. 29.

[43] Openlink Virtuoso: *Virtuoso Open-Source Edition Wiki.* http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/. Letöltés dátuma: 2010. 08.07.

[44] Simile Widgets Documentation Wiki: *Exhibit 2.3.* 2010. 08. 23. http://simile-widgets.org/wiki/Exhibit. Letöltés dátuma: 2010. 08. 27.