Cikkgyűjtés az Atom segítségével

Szeretnénk mindenkinek udvarias értesítést küldeni a weboldalunkon megjelent új tartalomról? Valósítsuk meg a legújabb cikkgyűjtő szabvány támogatását, és újabb eszközt nyerünk látogatóink visszacsalogatására.

A szervezett bűnözés világában a szindikátus együtt dolgozó bűnözők csoportját jelenti. A sajtó világában a szindikátus feladata az információk terjesztése az előfizetők felé, miközben minden kiadó testreszabhatja a kapott információkat. A karikatúrákat, híreket és publikációkat gyakran szindikátusok terjesztik, így a szerzők nagyobb közönséget érhetnek el, az olvasók pedig nagyobb tartalomválasztékból csemegézhetnek.
Az elmúlt néhány évben a webes fejlesztők is átvették a szindikátus kifejezést, és egyből igeként és főnévként is használni kezdték. Szerencsére a weben a szindikátus sokkal inkább a sajtó világában megszokott dolgot takarja, és nem sok köze van a mackósokhoz. Az viszont tény, hogy miként a szervezett bűnözés világában, a kapcsolódó nyilvános viták során is jónéhány ember kapott súlyos sebeket (igaz, nem fegyverek, inkább szavak által), ami megosztottsághoz és sok keserűséghez vezetett a webes szindikálás, cikkgyűjtés terén.
A megosztottság eredménye az ***Atom*** nevű új cikkgyűjtő formátum, amely sokban egyezik az ***RSS***-sel (Az ***RSS a „rich site summary"*** rövidítése. Használatos még az ***RDF site summary*** kifejezés is attól függően, hogy melyik változatról van szó és kit kérdezünk meg.) Én úgy vélem, hogy az Atom számos előnyt kínál az ***RSS*** bármely változatával szemben, és az Atom cikkek létrehozásának egyszerűsége egyértelműen előnyösebb választássá teszi az ***RSS***-hez képest. Figyelembe véve, hogy a legtöbb webnapló termék ***RSS*** cikkek előállítására képes, kijelenthetjük, hogy a két tábor békésen megfér egymás mellett. Érdemes viszont mindkét megoldás működését átlátni, hiszen így megalapozott döntést hozhatunk arról, hogy melyik – esetleg mindkettő – szabályrendszert kövessük.

Egy kis történelem

Mint a múlt hónapban láttuk, az ***RSS*** valójában két különböző formátum, vagyis inkább két különböző formátumcsalád. Az ***RSS 0.9x*** és az ***RSS 2.0*** ugyanabba a családba tartoznak, és kiválóan szemléltetik a webes cikkgyűjtés fejlődését. Az ***RSS 2.0***-t elsősorban a ***Userland***-től ***Dave Winer***, a ***scripting.com*** és újabban a ***Harvard University*** tartotta, tartja karban. ***Winer*** átadta a szabvány tulajdonjogát a ***Harvardnak***, ám azt is kijelentette, hogy a 2.0 lesz az utolsó változat. Van is benne valami, hiszen az RSS 0.9x és az RSS 2.0 együttesen széles körben elterjedt, üzembiztos, jól ismert – és a kissé zavaros – protokollt jelentenek a webes tartalmak összegyűjtésére.
Létezik egy másik ***RSS***-fajta is, neve a megtévesztő ***RSS 1.0***, amely az erőforrás-fejlesztési keretrendszerre (***resource development framework, RDF***) épül, és amelyet a ***World Wide Web Consortium (W3C)*** gondoz. Az ***RDF***-et arra tervezték, hogy a számítógépek képesek legyenek megérteni a webhelyek tartalmát, ennek alapján képesek legyenek kapcsolatokat teremteni közöttük, pontosan úgy, ahogy az emberek is teszik ezt ösztönösen. Az ***RSS 1.0*** összegzései a többi ***RSS***-változattal nem használhatók, viszont az ***RDF*** alapján szabványos leírást nyújtanak a webhely tartalmáról. Abból, hogy az ***RSS 1.0*** is az ***RSS*** nevet kapta, rengeteg vita és ellenségeskedés származott, sokan a legváltozatosabb módokon szidták ***Dave Winert***, az ***RSS*** előírások bizonytalanságát és az Atom elődjének alkotóit. Végül néhány vezető személyiség – ***Tim Bray***, ***Mark Pilgrim*** és ***Sam Ruby*** vezetésével –olyan cégek támogatásával, mint például a ***Six Degrees*** (ez adja ki a ***Movable Type*** webnapló programot) nekikezdett egy új szabálygyűjtemény kidolgozásának, amelyet eleinte ***PIE***-nak majd ***Echonak*** hívtak, és amely az ***RSS*** gyengeségeinek kiküszöbölését célozta.
Az ***Atom*** fejlesztése eltartott egy ideig, ugyanis első lépésként meg kellett határozni, hogy napjaink webje esetében pontosan mit is jelentsen a cikkgyűjtés. Az ***RSS***-t immár nemcsak eredeti alkalmazói, a híroldalak használják, de sok webnapló és nem szöveges jellegű tartalmat szolgáltató oldal is támogatja. A fejlesztők úgy döntöttek, hogy a többnyelvűségnek kiemelt szerepet szánnak, vagyis azt célozták meg, hogy a cikkeket tetszőleges nyelven lehessen szolgáltatni. Ugyancsak nagy hangsúlyt kapott a bővítmények fejlesztése, vagyis az ***Atomot*** úgy lehet új szolgáltatásokkal bővíteni, hogy a fő Atom szabályokhoz nem kell hozzányúlni.
Cikkem írásakor (2004. augusztusának közepén) az ***Atom*** szabálygyűjtemény 0.3-as változatban létezik, továbbá tartozik hozzá egy szabványos ***API*** is a hálózaton keresztül végzett tartalomszerkesztéshez. Az ***Atom*** elindult az ***IETF (Internet Engineering Task Force***, ez a szervezet állítja össze és teszi közzé az internetes szabványokat) szabvánnyá válás útján, vagyis hamarosan általánosan elfogadott szab-

vány lehet, akár csak a *TCP/IP*, az *SMTP* vagy a *HTTP*. Kétségtelen, hogy az *Atom* növekvő érdeklődésre számíthat azon szervezetek részéről, amelyek már csak az *IETF* jóváhagyó pecsétjére várnak.
Az *Atom* továbbra is kezdeti állapotban van, számos elemnek – például a bővítményeknek – hiányzik a nyilvános leírása. Készítői ugyanakkor máris elmondhatják, hogy készítettek egy szabványt, amelynek összetettsége közel áll az *RSS 0.9x* és *2.0* előírásgyűjteményéhez, jól érhető és áttekinthető, a webes cikkgyűjtő közösség komoly támogatását élvezi, valamint szemléletében messze túlmutat a puszta webes cikkgyűjtésen.

Atom cikk létrehozása

Míg az *RSS*-t elsősorban híroldalak és webnaplók bejegyzéseinek összegyűjtésére tervezték, az *Atom* inkább általános célú megoldásnak készült. Lehet például olyan rendszert építeni, hogy egy üzem gépei állapotjelentésüket *Atom* formátumban készítik el, majd egy cikkgyűjtő ezek alapján jelzi a meghibásodásokat. Megoldható, hogy a könyvtárak Atom cikkeket készítsenek legújabb beszerzéseikről, a különféle témákkal kapcsolatos könyvek elérhetőségét pedig intelligens cikkgyűjtők figyeljék. További felhasználási terület a faxgépek faxmodemekre cserélése, majd a faxképek Atom alapú terjesztése a megfelelő személyek vagy csoportok felé.
Az *Atom* akár szerkesztőségi rendszer összeállítására is alkalmas lehet, ahol a tudósítók írásaikat nem elektronikus levélben, hanem *Atom* cikkek formájában küldik el. Mindegyik szerkesztő összegyűjtené az irányítása alá tartozó tudósítók *Atom* cikkeit, majd a szerkesztési munka befejezése után kimenő *Atom* cikkekbe másolná őket. A cikkgyűjtés utolsó fázisa a tördelő részleg lenne, ahol előkészítenék a szövegek nyomtatását. Az újság tartalma tehát *Atom* cikkek áramlása révén állna össze, a végső cikk maga az újság lenne.
Atom cikket előállítani rendkívül egyszerű, ha *Perl*-t vagy más magas szintű nyelvet használunk, amelyhez létezik *Atom* könyvtár. A *Perl* esetében például az *XML::Atom* modullal tudunk dolgozni, amely a *CPAN*-ról tölthető le. Nekem Fedora Core 2 alatt, a *Perl 5.8.3*-as változatával volt némi gondom az *XML::Atom* telepítésével, ezt úgy tudtam megoldani, hogy az elhagyható *DateTime* modult kivettem a telepítésből. Éles környezetben ezt a megoldást nem javaslom.
Bár a teljes csomag neve `XML::Atom`, az *Atom* cikkeket előállító programok valójában az `XML::Atom::Feed` és az `XML::Atom::Entry` modult használják. Példaként lássunk egy rövid *Perl* programot, amely egy egyszerű cikket állít elő, és amely részben az `XML::Atom::Feed` internetes perldoc leírásában szereplő példaprogramra alapul:

```
#!/usr/bin/perl

use strict;
use diagnostics;
use warnings;

use XML::Atom::Feed;
use XML::Atom::Entry;

# Új Atom cikk létrehozása
my $feed = XML::Atom::Feed->new;
$feed->title(´Webnaplóm´);

my $entry;
# A cikk első bejegyzésének létrehozása
$entry = XML::Atom::Entry->new;
$entry->title(´Első írás´);
$entry->content(´Első írás törzse´);
$feed->add_entry($entry);

# A cikk második bejegyzésének létrehozása
$entry = XML::Atom::Entry->new;
$entry->title(´Második írás´);
$entry->content(´Második írás törzse´);
$feed->add_entry($entry);

# Az XML kimenet előállítása

my $atom_feed_xml = $feed->as_xml;

# Az XML kimenet megjelenítése
print $atom_feed_xml, "\ n";
```

A fenti program az alábbi cikket állítja elő, amelyet a könnyebb olvashatóság kedvéért kicsit átformáztam:

```
<?xml version="1.0"?>
<feed xmlns="http://purl.org/atom/ns#">
<title>
    Webnaplóm
</title>
<entry
xmlns:default="http://www.w3.org/1999/xhtml">
    <title>
    Első írás
    </title>
    <content mode="xml">
    <default:div
        ➥xmlns="http://www.w3.org/1999/xhtml">
        Első írás törzse
    </default:div>
    </content>
</entry>
<entry
xmlns:default="http://www.w3.org/1999/xhtml">
    <title>
    Második írás
    </title>
    <content mode="xml">
    <default:div
        ➥xmlns="http://www.w3.org/1999/xhtml">
        Második írás törzse
    </default:div>
    </content>
</entry>
</feed>
```

Mint láthatjuk, létre kell hoznunk egy `XML::Atom::Feed` objektumot, amibe az `XML::Atom::Entry` egy vagy több példánya kerül. Az Atom cikken belül minden bejegyzés

objektum egy-egy <entry> címkével párosul, ezek pedig egy-egy üzenetet jelenítenek meg webnaplónkból vagy éppen az üzem felügyeleti rendszeréből.
Az Atom szabálygyűjtemény szerint a cikk számos jellemzőt és alelemet is magába foglalhat, ilyen például a nyelv, a webnapló vagy webhely leírása, a szerzői jogi megjegyzések és a származási hellyel kapcsolatos egyéb tájékoztatások. Az egyes bejegyzések ugyanakkor saját elemkészlettel rendelkeznek, mint például cím, létrehozás időpontja és összegzés. Minden Atom elemnek van egy *MIME* típusa, amely a tartalom jellegéről tájékoztat, hasonlóan a *HTTP*-válaszokhoz és az elektronikus levelek mellékleteihez.
Természetesen cikk létrehozására a fenti módon csak akkor van szükség, ha új *Atom*-képes alkalmazást írunk, vagy szeretnénk ilyen irányba bővíteni meglévő webnapló alkalmazásunkat. A legtöbb webnapló termék már most is képes Atom cikkek előállítására, akár a normál csomag részeként, akár valamilyen beépülő modul vagy egyéb bővítmény révén. A *Blosxom Weblog* rendszerhez például beépülő modul készült, segítségével könnyedén megoldható az Atom cikkek előállítása. A beépülő modult egyszerűen csak be kell másolni a *plugins* könyvtárba, és ettől kezdve bárki kaphat Atom cikkeket a kérdéses webnaplóról.
Remélem, nem okozott meglepetést, hogy mindez ennyire egyszerű, ugyanis a *Blosxom Perlben* íródott, a *Perl* pedig kiváló eszközöket kínál az *XML* kezelésére, a beépülő modulnak pedig csak annyi a dolga, hogy összesítse és újraírja a webnapló legújabb bejegyzéseinek tartalmát. Abból, hogy a *Blosxom* ennyire könnyűvé teszi a beépülő modulok számára a főoldal módosítását (ahogy az *Atom* cikk megjelenésének jelzését is) és a tartalom lekérdezését (a beépülő modul *API*-n keresztül), egyenesen következik, hogy alóla rendkívül könnyű az Atommal dolgozni. Mivel a legtöbb webnapló program magas szintű nyelven, tehát *Perlben*, *Pythonban* vagy *PHP*-ban készül, az *Atom* kezelésének megvalósítása még nulláról kezdve sem okozhat nehézséget.

Atom cikk feldolgozása

Az *Atom* cikkek feldolgozására rengeteg módszer közül választhatunk, akár cikkgyűjtőt szeretnénk írni, akár *Atom* alapú alkalmazást szeretnénk készíteni. A legegyszerűbb megoldás a cikkek felfedezésére és lekérdezésére továbbra is az `XML::Atom::Feed` használata. Például:

```
#!/usr/bin/perl

use strict;
use diagnostics;
use warnings;

use XML::Atom::Feed;

# A www.diveintomark.org Atom cikkeinek
# lekérdezése
my @uris =
    XML::Atom::Feed->find_feeds(
        “http://www.diveintomark.org/”);

    # Az egyes Atom cikkekhez tartozó URI-k
    # kiírása
    foreach my $uri (@uris)
    {
    print “uri = ´$uri´\ n”;
    }
```

A fenti példával egyszerűen meg tudjuk jeleníteni az *URI*-kat. Most, hogy már tudjuk, hol keressük az egyes cikkeket, elő tudjuk állítani a bennük szereplő hivatkozások listáját, majd a hivatkozásokat *XML* formára tudjuk hozni:

```
#!/usr/bin/perl

use strict;
use diagnostics;
use warnings;

use XML::Atom::Feed;

# Atom cikk lekérése
my @uris = XML::Atom::Feed->
➥find_feeds(“http://www.diveintomark.org/”);

foreach my $uri (@uris)
{
my $feed = XML::Atom::Feed->new(URI->new($uri));

my @links = $feed->link();

foreach my $link (@links)
{
    my $link_xml = $link->as_xml();
    print “link = ´$link_xml\ n”;
}
}
```

Természetesen *XML*-t nem állítunk elő és nem jelenítünk meg, a lényeg a hivatkozások kinyerése, amelyeket elektronikus levélben el tudunk küldeni az előfizetőknek, hozzá tudunk adni a megfelelő adatbázishoz, de akár el is hagyhatjuk közülük az adott feltételeknek meg nem felelőket.
Mivel az *Atom* cikkek szabályos felépítésűek, illetve internetes szabványokra alapulnak, mint az *XML*, az *Unicode* és a *MIME*, biztosak lehetünk abban, hogy a cikkből kiemelt tartalmat egyszerű módszerekkel tudjuk kezelni. A különféle tartalomtípusokat különféle kezelőprogramoknak továbbíthatjuk, különféle módszerekkel dolgozhatjuk fel (akárcsak a korábban említett szerkesztőségi rendszerben) vagy új cikkekbe másolhatjuk őket, így akár egy „szupercikkgyűjtőt” is létre tudunk hozni.
Aki szeretne cikkgyűjtőt készíteni, vagy szeretne pontosabb képet alkotni arról, hogy a milliófféle *RSS*- és *Atom*-változatokkal hogyan tud dolgozni, annak érdemes megismernie *Mark Pilgrim* cikkgyűjtőjét. A program *Python* alapú, szerzője folyamatosan frissíti, és talán ez a legjobb leírással ellátott nyílt forrású motor a cikkgyűjtő rendszerek anyagainak kezelésére.

RSS vagy Atom?

A fő kérdés: weboldalunk – vagy éppen webnaplónk – az *RSS* vagy az *Atom* formátumú cikkgyűjtést támogassa, esetleg mindkettőt? Számomra egyértelmű, hogy az *Atom* a leg-

jobb a napjainkig kidolgozott kettő (pontosabban három) cikkgyűjtő formátum közül. ***Dave Winer RSS*** formátumai megjelenésükkor komoly újdonságnak és előrelépésnek számítottak, ám túl sok hibájuk volt ahhoz, hogy egy teljes értékű, akár vállalati környezetben is használható szabvány alapját adják. A ***HTML*** és a ***JavaScript*** korai változatainak példáján láthattuk, hogy a félkész szabványok csak kínlódást hoznak, és figyelembe véve, hogy a cikkgyűjtés a jövőben jó eséllyel rendkívül fontos kommunikációs módszerré válik, a teljesség és az egyértelműség alapkövetelménynek számít.

Hasonlóan fontos a különféle nyelvek támogatása, valamint azt sem szabad elfeledni, hogy az emberek nem csak szövegeket szeretnének közzétenni. Az ***Atom*** egyértelműsége a különleges karakterek kezelése tekintetében szintén hangsúlyos tényező, ennek köszönhetően biztosak lehetünk abban, hogy ha elhelyezünk a webnaplónk szövegében egy < and > elemet, akkor ezzel nem fogjuk megzavarni a cikkgyűjtés működését. A legfontosabb talán mégis is, hogy a tervezett bővítési lehetőségek révén az Atom bármely különleges csoport vagy alkalmazás igényeinek képes lesz megfelelni úgy, hogy magát az alapot nem kell majd módosítani.

Az ***Atom*** rendkívül sokoldalú, használata mégis egyszerű, ennek fejlesztése során is nagy figyelmet szenteltek. Egy új ***API***-t létrehozni nem egyszerű, különösen, ha a lehető leginkább általános jellegűnek kell lennie.

Végül megjegyeznék annyit, hogy az ***RSS*** változatszámok kavalkádja, amely kicsinyes és szinte már a politikára emlékeztető vitákat szült – miközben maga a kavalkád is ezeknek a vitáknak köszönhetően alakult ki – gyakorlatilag senkinek nem vált hasznára. Az ***Atom*** különböző nevet kapott, és bár ennek a névnek homályos a jelentése, legalább nem súlyosbítja a fejlesztők és a felhasználók oldalán az ***RSS*** miatt kialakult félreértéseket.

Összefoglalás

Az ***Atom*** révén esélyt kaptunk az ***RSS*** kapcsán felmerült gondok megoldására, illetve arra, hogy a cikkgyűjtés újfajta, internetes alkalmazások között folyó, magas szintű adatcsere megoldások alapjává válhasson. Az ***Atom*** némileg bonyolultabb ***Dave Winer RSS***-változatainál, ám – legalábbis első változatát tekintve – egyszerűbb, mint a webhelyek leírását és tartalmuk összesítését az ***RDF*** alapján végző ***RSS 1.0***. Az ***Atom*** cikkek kezelését leegyszerűsítő, könnyen használható segédeszközök, a bővíthetőség és a szerzőknek az internetes szabványokra fordított figyelme egyértelművé teszik, hogy az ***Atom*** fontos szerepet fog játszani a webes kommunikáció jövőjében.

Linux Journal 2004. november, 127. szám

Reuven M. Lerner (➲ http://www.lerner.co.il/atf) Nyílt forrású programokra, valamint web- és adatbázis-alkalmazásokra szakosodott tanácsadó. Könyve, a Core Perl, 2002 januárjában jelent meg a Prentice Hall gondozásában. Reuven feleségével és lányaival nemrég költözött Chicagóba.