Bloglines webszolgáltatások (folytatás)

Néhány webes közösség gonosz módon bezárja a felhasználóit, a Bloglines ezzel szemben a nyitott megközelítés híve és megengedi, hogy saját parancsfájljainkkal irányítsuk a webszolgáltatás API felületét. Vágjunk bele és zárkózzunk fel kedvenc blogjainkkal.

Ezeket a sorokat néhány nappal a 2004-es november 2-án tartott amerikai elnökválasztás után írom. Mint közismert politika függőségben szenvedő, kiélvezhetem a számítógépesített modern érát, és az éppen aktuális bölcsességeket. Többé nem kell kapcsolgatnom a TV csatornák között, vagy újságokat olvasnom a helyi könyvtárban; nyugodtan végigkövethetem a részleteket amint a jelöltektől a sajtóhoz majd a különféle partizán oldalakra kerülnek. Lépést tartani a rengeteg különféle híroldallal igen sok időt vehet igénybe. Mint az elmúlt néhány hónapban láthattuk, mindenki nyert valamit a hírösszesítő programok használatával, amelyek begyűjtik a weblogok, újságok és más, gyakran frissített lapok által készített ***RSS*** és ***Atom*** tartalmakat. Az összesítő, mint a neve is mutatja, ezeket tartalmakat rendezi könnyen áttekinthető listába.

A ***Bloglines.com*** olyan Internetes kezdőpont amely web-alapú hírösszesítőt is kínál nekünk. Mindez önmagában még nem lepne meg senkit, hiszen a hírgyűjtemények, az összesítők és a web együtt szinte tálcán kínálják ezt a lehetőséget. Ráadásul a ***Bloglines*** nem is egyedi. Számtalan egyéb, igaz, talán nem annyira jól ismert Web-alapú hírösszesítő létezik. A ***Bloglines*** azonban egyedi szolgáltatást is kínál használóinak, ugyanis a ***Bloglines*** belső adatbázisát felhasználva saját összesítőt vagy akár saját alkalmazást készíthetnek a ***Bloglines*** által összegyűjtött adatokból. Mindez az információ ellenszolgáltatás nélkül hozzáférhető, egy viszonylag kötetlen szerződés mellett, valamennyi programozó számára, akit érdekel a ***Bloglines*** motor eredményeinek összegyűjtése. Mivel a ***Bloglines*** körülbelül óránként blogok és honlapok százezrein keres frissítéseket, a webszolgáltatás ***API*** használói biztosak lehetnek benne, hogy legfrissebb weblog tartalomhoz jutnak hozzá.

Legutoljára a ***Notifier API***-t vettük szemügyre, amely egy adott felhasználó elérhető, de még olvasatlan gyűjteményéhez enged hozzáférést. Megnéztük a ***Blogroll API***-t is, melynek segítségével a felhasználó, ha úgy kívánja, meghatározhatja vagy programjában felhasználhatja a gyűjteményre hivatkozó emberek listáját. Mit láthattuk, ezek a ***API***-k sokat segítettek, ha az új weblog bejegyzéseket szeretnénk megtalálni, vagy ha saját gyűjteményt listát akarunk összerakni az érdekes weblogokból.

Valami azonban hiányzott a cikkben bemutatott lehetőségek közül. Jó dolog tudni, hogy az új weblog bejegyzések a ***Bloglines*** kedvenceink közé kerül, de még jobb lenne ha azt is tudnánk, melyik blog frissült éppen. Továbbá, nem rossz, hogy lekérdezhetem az aktuális feliratkozási listámat, de még boldogabb lennék, ha azt is tudnám, melyikük frissült, és pontosan mikor frissültek legutoljára, valamint hány bejegyzés van az egyes weblogokban és azok a bejegyzések mit is tartalmaznak. Más szóval, a jelenlegi ***Bloglines*** felületet szeretném a sajátommal lecserélni, és az új weblog bejegyzéseket olyan alakban megjeleníteni, amely nem feltétlenül felel meg a ***Bloglines.com*** weblap által felajánlottnak. Szerencsére, a ***Bloglines*** webszolgáltatás fejlesztői lehetővé tették, hogy pontosan így járjunk el, mégpedig a ***sync API*** segítségével. Ebben a hónapban, folytatjuk a ***Bloglines*** webszolgáltatásainak felfedezését, és elmerülünk a ***sync API*** részleteiben. Megpróbálunk létrehozni egy saját, egyszerű hírösszesítőt, amely rendelkezik néhány, a ***Bloglines*** felületből már ismert képességgel.

Feliratkozások és elemek

Végső soron a ***Bloglines*** és a hozzá hasonló hírösszesítők nem állnak másból, mint egy ***URL*** listából. A két hónappal korábban összerakott, ***Python*** nyelvű és ***Universal Feed Parser*** alapú hírösszesítőnk pontosan ilyen program volt – ***URL*** listát vizsgált egy fájlban és az adott ***URL***-ekhez tartozó legfrissebb tartalmat gyűjtötte össze. Minden egyes weblog küldemény a lista egy-egy ***URL***-jének felel meg. Ha az ***URL***-t eltávolítjuk a feliratkozási listánkból, a hozzá tartozó küldemények többé már nyilvánvalóan nem érdekesek a felhasználó számára, ezért nem is látja őket.

Az a tény, hogy a ***Bloglines*** több felhasználóval dolgozik, azt jelenti, hogy nem elég egyetlen ***URL*** listát tárolnia, hanem azt is meg kell jegyeznie, hogy melyik ***URL*** melyik felhasználóhoz tartozik. Bár ez nyilván kicsit megbonyolítja a dolgokat, a modern magas szintű nyelvek segítségével nagyon könnyű megérteni a két adatszerkezet közötti különbségeket. Az ***URL*** lista egyszerű tárolása helyett egy hash táblát kell létrehoznunk, ahol a felhasználói azonosítót használjuk kulcsként értéke pedig az adott felhasználóhoz

1. lista A felhasználó feliratkozásainak megjelenítése

```
#!/usr/bin/perl
use strict;
use diagnostics;
use warnings;
use WebService::Bloglines;
my $username = 'reuven@lerner.co.il';
my $password = 'MYPASS';
my $bloglines =
  WebService::Bloglines->new(username =>
$username,
                             password =>
$password);
# Olvasottként akarjuk megjelölni?
my $mark_unread = 0;
# Milyen dátumtól kezdve szeretnénk letölteni?
# (ennek Unix időnek kell lennie)
my $subscriptions = $bloglines->listsubs();
if ($subscriptions)
{
    # a tartalmak listája
    my @feeds = $subscriptions->feeds();
    # Beszerezzük a tartalmak címét és URL-ét
    foreach my $feed (@feeds) {
        my $title  = $feed->{title};
        my $url    = $feed->{htmlUrl};
        my $subId  = $feed->{BloglinesSubId};
        print "Subscribed to '$title', "
            . "subId '$subId' at '$url'\n";
    }
}
else
{
    print "No subscriptions.\n"
}
```

tartozó lista lesz. Ha a felhasználó egyedi azonosítóját ismerjük, könnyedén nyomon tudjuk követni az adott felhasználó feliratkozásait.

Természetesen, a ***Bloglines*** nem néhány ezer ember listáit tartja karban, hanem sok tíz vagy százezerét. Így nyugodtan állíthatjuk, hogy nem ilyen naiv megvalósítást használnak, amely legfeljebb egy kísérlet, vagy néhány ember számára tervezett összesítő számára felelhet meg. A dolgok kicsit trükkösebbé válnak amikor megközelítjük a ***Bloglines*** felhasználói terhelését. A felhasználók feliratkozási listái nem lehetnek egyszerű ***URL***-ek; sokkal valószínűbb, hogy az ***URL***-ekhez rendelt azonosító számokat (vagy ahogy az adatbázis körökben nevezik „elsődleges kulcsot") használnak. Egy ilyen rendszernek megvan az az előnye, hogy több jelentkezőnek is lehetővé teszi a feliratkozást a lap hírösszesítőjére, a ***Bloglines*** a már meglévő feliratkozások alapján javaslatot tud tenni nekik , hogy mely további weblogok érdemesek még a figyelmükre.

Ezek után nem okoz nagy meglepetést ha megtudjuk, hogy az új weblog küldemények lekérése a ***Bloglinestól*** két lépésből áll. Az első lépésben lekérjük a feliratkozások listáját, pontosabban a ***Bloglinestól*** lekérdezzük a felhasználóhoz rendelt feliratkozási azonosítókat. Ezt követően utasítjuk a ***Bloglinest***, hogy küldje el az adott felhasználóhoz és a kapott azonosítóhoz tartozó összes új elemet.

A ***Bloglines*** webkiszolgáló ***API*** megvalósítása több különféle programnyelven is elérhető. Minthogy az új alkalmazásokat többnyire ***Perlben*** készítem, a `WebService::Bloglines` modult fogom használni amelyet feltöltöttek a ***CPAN***-ra, azaz a ***Comprehensive Perl Archive Network*** nevezetű világméretű web és ftp kiszolgáló hálózatra, amelyről a ***Perl*** és annak moduljai tölthetőek le. Az *1. listában* látható egyszerű program (***bloglines-listsubs.pl***) megjeleníti a címet, a feliratkozás azonosítóját és a hozzá tartozó ***URL*** a felhasználó összes feliratkozására. Minden feliratkozáshoz további információ is elérhető; ezeket a `WebService::Bloglines`, illetve a ***Bloglines API*** dokumentáció részletesen ismerteti.

Amennyiben meg szeretnénk őrizni a felhasználók ***Boglines.com*** felületén látható sorrendjét , érdemes a `folders` függvényt használnunk az 1. listában olvasható `feed` függvény helyett. A `feed` ugyanis egyszerűen csak visszaadja a feliratkozásokat, a `folders` ellenben ugyanolyan szerkezetben adja őket vissza, ahogy azt a Bloglines lapján láthattuk.

Feliratkozások letöltése

Most már tudjuk, hogyan szerezhetjük meg egy adott ***Bloglines*** felhasználóhoz tartozó feliratkozás azonosítókat, így le tudjuk kérdezni a feliratkozás azonosítókhoz tartozó egyes elemeket is. Ezt mutatja be a *2. lista* rövid programja amely letölti a felhasználó összes feliratkozását, majd valamennyihez megmutatja a legutóbb frissült elemeket. A kimenete nem ***HTML***, hanem egyszerű szöveges formátum, következésképpen a megjelenített hivatkozásokra nem tudunk rákattintani. Mindazonáltal nem különösebben nehéz egy ilyen programot ***cron*** feladatként lefuttatni és a kimenetét egy ***HTML*** fájlba irányítani, így percre friss, személyre-szabott tartalomlistát kapunk. Természetesen a ***Bloglines*** ingyenesen biztosítja nekünk ezt a szolgáltatást valahányszor csak belépünk a weblapjukra. Így aztán a program érdekes tesztje a ***Bloglines*** webszolgáltatásainak, de azokhoz képest igazából nem nyújt semmi újdonságot.

Az egyik ügyes húzás amit a ***Bloglines*** a webszolgáltatás definíciójában bevezetett , hogy ***HTTP*** visszatérési kódokat használ a hibák és szokatlan események visszajelzésére. Például, a `200 (OK)` válaszkód azt jelzi, hogy van új beolvasandó elem illetve, hogy a `getitems($subId)` egy vagy több ilyen elemet tartalmaz. A 304 (változatlan) válaszkód amely egyébként azt jelzi, hogy a ***HTML*** az utolsó lekérdezés óta nem változott, itt egy kicsit más szerepet kap; azt jelenti, hogy a az adott feliratkozó már az összes elemet látta ebben a feliratkozásában. A többi válaszkód (401, 403 és 410) azonosítási hibákat jelez, ami valószínűleg annak következménye, hogy a kérelmező felhasználó elgépelte a ***Bloglines*** felhasználónevét, jelszavát, esetleg mind a kettőt.

Sajnos ***Perl*** alatt e hibakódok kezelése messze van az optimálistól. Lekérdezésükhöz az `eval` blokkon belül meg kell hívnunk a `$bloglines->getitems()` függvényt, majd közvetlenül az `eval` meghívását követően meg kell vizsgálnunk a `$@` értékét. Ha a `$@` üres, feltételezhetjük, hogy 200-as (OK) HTTP válaszkódot kaptunk és vannak beolvasandó új elemek. Amennyiben azonban értéket tartalmaz, újra kell írnunk a kimeneti üzenetet, ahogy azt a 2. listában meg is

2. lista bloglines-getitems.pl

```
#!/usr/bin/perl
use strict;
use diagnostics;
use warnings;
use WebService::Bloglines;
my $username = 'reuven@lerner.co.il';
my $password = 'MYPASS';
my $bloglines =
  WebService::Bloglines->new(username => $username,
                             password =>
$password);
# Olvasottként akarjuk megjelölni?
my $mark_unread = 0;
# Milyen dátumtól kezdve szeretnénk letölteni?
# (ennek Unix időnek kell lennie)
my $subscriptions = $bloglines->listsubs();
if ($subscriptions)
{
    # a tartalmak listája
    my @feeds = $subscriptions->feeds();
    foreach my $feed (@feeds) {
        my $title  = $feed->{title};
        my $url    = $feed->{htmlUrl};
        my $subId  = $feed->{BloglinesSubId};
        print "Subscribed to '$title', "
            . "subId '$subId' at '$url'\n";
    my $update;
    # Elkapjuk a hibákat !
    eval {$update = $bloglines->getitems($subId);};
    # Figyeljük a hibákat, kiírjuk ha nincs változás.
    if ($@) {
        if ($@ =~ /^304 No Change/) {
            print "\t No change\n";
        }
        else {
            print "\t Error code '$@' "
                . "retrieving updates.\n";
        }
    }
    # Nincs hiba?  Néhány alap dolgot kiírunk az
elemekről.
    else
    {
        foreach my $item ($update->items)
        {
        my $title       = $item->{title};
        my $creator     = $item->{dc}->{creator};
        my $link        = $item->{link};
        my $pubDate     = $item->{pubDate};
        print "\t$title by $creator "
            . "on $pubDate ($link)\n";
        }
    }
    }
}
else
{
    print "No subscriptions.\n"
}
```

tettük. Amennyiben ezt a metódushívást nem tesszük `eval` blokkba, akkor sajnos a programunk végzetes hibával leáll, amint a 200-as válaszkódon kívül bármi mást kapunk.

Végül a ***Bloglines*** funkcióit két opcionális paraméter teszi teljessé. Az első n-ként ismert paraméter egyszerű igaz-vagy-hamis érték (1 vagy 0) amely értesíti a ***Bloglines***-t , hogy frissítenie kell-e a „már láttam" bitet az éppen letöltött cikkeknél. Egyébként amikor a felhasználó a ***Bloglines.com*** webes felületén keresztül nézi a weblog küldeményeit ezt mindig 1-re állítjuk, ami azt jelenti, hogy a korábban elolvasott cikkek újra már nem kerülnek elő. Talán, mivel tudták, hogy a webszolgáltatások ***API*** más hírösszesítő alkalmazásokat is kiszolgál, a Bloglines bölcsen 0-ra állította ezen érték alapértelmezését az ***API***-ban.

A második, d nevű opcionális paraméter azt az első dátumot mutatja meg a ***Bloglinesnak*** amelytől kezdve az adott honlap cikkeit le szeretnénk tölteni. Az érték ***UNIX*** időformátumban értendő, azaz 1970 január elseje óta eltelt másodpercek számát kell megadnunk. Ezt a számot a nagyobb nyelvek mindegyikében készen kapjuk, és segítségével nagy pontossággal meg tudjuk adni, hogy pontosan milyen mélyen szeretnénk az adott honlap ***Bloglines*** tárolta történelmébe nyúlni.

Összefoglalás

Az igazat megvallva lelkes ***Bloglines*** felhasználónak tartom magam, annak ellenére, hogy pontosan tudnám hol is van a lap és a cég székhelye. Nehezen tudom elképzelni, hogy továbbra is ingyenes és reklámmentes marad, hacsak fejlesztői nem jótékonyságból cselekszenek vagy végzetesen naivak. Kedvelem a kiváló felületét, a tényt, hogy könnyedén el tudom érni azokat a weblogokat amelyektől politikai éleslátásom függ (vagy éppen mulatságom, attól függően ki hogyan ítéli meg az ilyen bölcsességeket) és gyors, hibatűrő képességeit.

Azonban ahogy az ***Amazon***, ***eBay*** és ***Google*** megmutatta az elmúlt néhány évben, nyers adatainkhoz webszolgáltatás felületet adva olyan új kreatív alkalmazásoknak nyitunk kaput, amelyekre a belső fejlesztők álmukban se gondoltak volna. A ***Bloglines*** mostanában kezdte el funkcióit webszolgáltatások formájában közzétenni, és igaz ugyan, hogy mindez egyelőre csak kezdeti és kísérleti lépés, a magam részéről nagyon ígéretesnek látom. Alig várom, hogy olyan alkalmazásokat lássak, amelyek erre az ***API***-ra valamint a ***Bloglines*** és versenytársai által kiadott további API felületekre épülnek, hogy a ***Bloglines*** lapját tegyék a weblogok, olvasók és fejlesztők Mekkájává.

Linux Journal 2005. március, 131. szám

Reuven M. Lerner veterán web/adatbázis tanácsadó és fejlesztő, jelenleg végzős hallgató a Northwestern University Learning Sciences programjában. Weblogja a altneuland.lerner.co.il címen olvasható őt magát pedig reuven@lerner.co.il címen érhetjük el.