# PHP – mnoGoSearch

## Azaz hogyan házasítsuk az izmos keresőmotort a szeretett PHP-val?

Sokszor felmerülő igény egy kiterjedtebb honlappal szemben, hogy a tartalma kereshető legyen. A feladat elég nyögvenyelős lehet, amennyiben a webhely sok, különféle működésű modulból épül fel. Minden egyes modulra saját keresőt építeni, majd egy rendszerbe foglalni őket, igen nagy falat lehet. Érdemes elgondolkodni, nem volna-e jó általánosítani valahogy a feladatot, és az összes egyedi esetet egységes rendszerbe foglalni. Hosszú távon mindenképpen kifizetődő az egyszeri nagyobb munkát bevállalni. Egy jól összerakott keresőrendszert feleltetni meg a későbbi igényeknek már töredékét teszi ki az amúgy szükséges munkaóráknak. Amikor az ember azon gondolkodik, hogy valamiféle általános megoldás lenne az igazi, az sem árt, ha körülszimatol egy kicsit a Világhálón, hátha a több tízezer nyílt forrású megoldás között talál valami, a célnak megfelelőt.

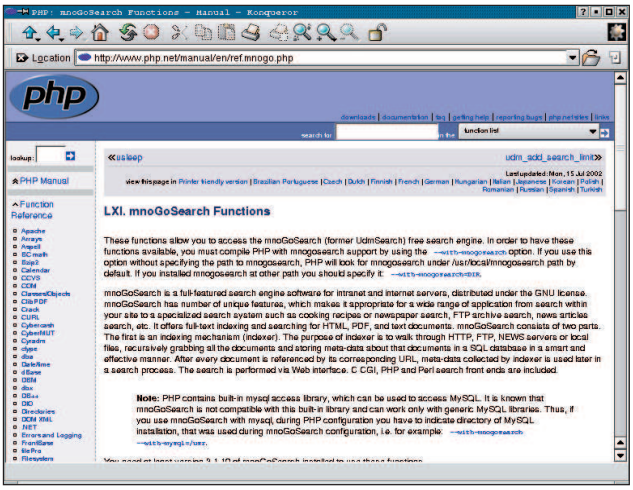
### A mnoGoSearch – ismerkedés

Jómagam a fenti gondolatmenetet végigvezetve jutottam el a mnoGoSearch keresőmegoldásához, még messzire sem kellett érte mennem. A PHP kézikönyvét böngészgetve akadtam rá – mondhatni véletlenül – a mnoGoSearch nevű kiegészítésre. Kis utánajárás után kiderült, hogy a mnoGoSearch maga egy igen kiterjedt, nyílt forrású keresőmotor. Saját, C-ben írt CGI-felülete is jól használható és rugalmas. Emellett Perlhez és PHP-hez is létezik illesztés. Ez a támogatás a PHP hivatalos terjesztésének részét képezi.

Maga a mnoGoSearch két jól elkülöníthető részre bontható. Az egyik végzi a weboldalak, adatbázistáblák indexelését a megadott beállítások alapján. Ezt időszakosan (tipp: cron) futtatva egy szorgalmas keresőrobotot állíthatunk munkába. Magának a keresőfelületnek a megvalósításában lehetőségünk nyílik dönteni, hogy a mnoGoSearch sajátját használjuk-e fel vagy írunk egyet. Ha a kereső saját megoldását alkalmazzuk, a keresőoldalt sablonfájl segítségével faraghatjuk egyedi arculatúra. Ha a PHP-illesztésen keresztül vesszük fel a kapcsolatot, teljesen egyedi felületeket hozhatunk létre. Ennek a programozói feladatnak a bonyolultsága nem haladja meg egy egyszerű adatbázis-lekérdezés szintjét.

A mnoGoSearch nagy erőssége, hogy külső szűrőket is képes alkalmazni, amelyek segítségével akár ***.pdf*** vagy egyéb különlegesebb formátumú állományok szövegének indexelésére is sor kerülhet. Ehhez egy, az adott formátumú anyagot szöveges állománnyá alakító programot kell beszereznünk. Ha a program megvan, a keresőt kell beállítani, hogy az adott típusú állományokat ezen a szűrőn átengedve lásson hozzá a tartalom átvizsgálásához. A mnoGoSearch jó pár formátummal és kapcsolódási lehetőséggel önmagában is elboldogul. Beépített képességei a következők:

- egyszerű text/html, text/plain oldalakat indexel (még szép);
- dinamikusan létrehozott tartalmú oldalakat a legcifrább URL-en keresztül is is elér;
- titkosított (HTTPS) kapcsolatokat is kezel;
- ha kell, jelszavasan védett (HTTP-azonosítás) oldalak is feldolgoztathatók vele;




*1. kép* ➲ http://hu.php.net/mnogo

- FTP-helyek végignyálazása sem okoz neki gondot;
- adatbázis-kiszolgálók tábláinak indexelésével is elboldogul, bár itt vannak megkötések.

A keresőmotor maga is SQL-alapú, adatbázisban tárolja az indexelés során összegyűjtött adatokat. Rengeteg adatbáziskiszolgáló-típussal képes együttműködni. Ha indexelésük céljából közvetlen adattáblákban is turkálni szeretnénk, egy erős megszorítással találkozunk: csak ugyanabban az adatbázisban képes feldolgozni a mnoGoSearch-adattáblákat, amelyikben a saját indexelésre fenntartott táblákat is tárolja. Ez alól az egyedüli kivétel az, ha háttéradatbázisként MySQL-t használunk.

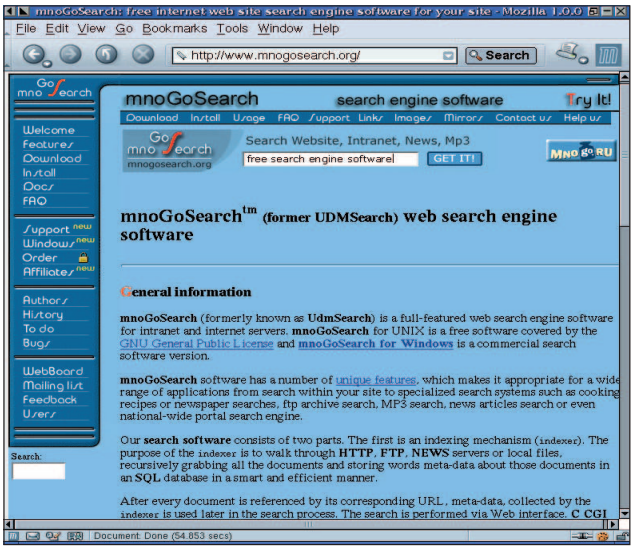
### A mnoGoSearch telepítése

Először is szükség lesz egy friss forrásra. Ehhez a mnoGoSearch honlapját kell felkeresnünk, azon belül is a következő oldalt: ➲ http://www.mnogosearch.org/download.html. A cikk írásakor a legutolsó üzembiztos kiadás mellett döntöttem, ami a 3.1.20 változatszámmal bír. Ennek telepítését Debian Woody környezetben próbáltam ki, de egyes apró eltéréseket kivéve ez a folyamat valószínűleg más GNU/Linux-rendszereken is követhető. A bemutatott telepítés MySQL jelenlétét feltételezi, ugyanis ezt választottam háttéradatbázisnak.

Ha megvan a ***mnogosearch-3.1.20.tar.gz*** állomány (vagy az éppen legfrissebb), a szokásos módon lehet nekiesni. Kicsomagolás, belépés a könyvtárba, ismerkedés a `./configure --help` kiadása által kapott lehetőségekkel. Eztán a `configure` parancsfájl megfelelő beállításokkal való futtatása következik, végül `make`, `make install`.

Ez a hagyományos telepítési módszer, ám érdemes észrevenni, hogy a fejlesztők egy kellemes Perlben írt kis kérdezgetőprogramot mellékeltek, ami végül a válaszoknak megfelelően hívja meg a `configure` héj-parancsfájlt. Lássuk, miféle beállításokat hajthatunk végre!

Ha a fent említett `install.pl` futtatása mellett döntünk, az első kérdése arra fog vonatkozni, hogy mi legyen a telepítés célkönyvtára – ez alapesetben a */usr/local/mnogosearch* könyvtár. Ha nincs jobb ötlet, így is maradhat, a PHP is itt fog majd kutakodni utána.
Ha a `configure` közvetlen meghívását választjuk a fordítás előtti beállítások végrehajtására, e fenti adat a `--prefix=<k nyvtÆr>` kapcsolóval adható meg. Ha meg




*2. kép* A mnoGoSearch hivatalos honlapja

vagyunk elégedve az alapértelmezett könyvtárral (lásd az előző bekezdést), az adat megadása el is hagyható.
Ha másképp nem rendelkezünk, a most megadott könyvtár alá kerül minden mnoGoSearchcsel kapcsolatos adat, a beállításoktól egészen a naplóállományokig. Nálam ez így maradt.
Kézi `configure` híváskor ez esetben semmit nem kell megadni, az `install.pl` futtatásánál is elég ENTER-t ütni az alapértelmezett ***no*** válaszra. Ha ez a felállás mégsem tetszene, egy ***yes*** után az összes alkönyvtár elhelyezkedése sorra megadható. Parancssoros beállítás esetén a szükséges kapcsolók a `configure` súgójából ismerhetők meg.
Ha a könyvtárak megadásán túltettük magunkat, következik a háttéradatbázis megadása. A beépített adatbázis-támogatásra nem lesz szükség, hiszen a gépen van telepített MySQL.
Az `install.pl` ezt magától képes megtalálni, tegye is csak. Válaszoljunk ***yes***-t arra a kérdésre, hogy maga keresse-e meg a lehetséges adatbázis-csatlakozásokat. Ha nem találná a megfelelő könyvtárat, magunknak kell megadnunk. Először az adatbáziskiszolgáló típusát, majd azt a könyvtárat kell megadni, ami alatt a MySQL fejlécfájlok és függvénykönyvtárak találhatók. A MySQL-támogatással való felruházáshoz kézi vezérlés esetén a `--with-mysql=<k nyvtÆr>` megadására lesz szükség.
Az ezt követő kérdésekre mindenütt hagyhatjuk az alapértelmezett választ, ezek a beállítások céljainkhoz javarészt jók lesznek. Ha a kézi `configure` meghívást választottuk, és semmilyen egyedi beállítást nem akarunk alkalmazni, nem kell további kapcsolókkal bajlódnunk.
Vagy így, vagy úgy, de eljutottunk oda, hogy a `configure` parancsfájl sikeresen lefutott. Ekkor már a hagyományokat követve jöhet a szokásos `make`, majd a `make install`. Ezáltal a mnoGoSearch alaptelepítése a helyére került.

## További tennivalók és beállítások

Bizony, a keresőrendszer telepítése még csak az első lépés volt. A feltett csupasz rendszert életre kell kelteni, be kell állítani. Először is a mnoGoSearch számára létre kell hozni a MySQL háttéradatbázist:

```
mysqladmin -uroot -p<jelsz > create
➥mnogosearch
```

Ezáltal létrejött az üres adatbázis, amelyben majd a mnoGoSearch a szótárát, az oldaladatbázisát fogja tárolni. Ehhez azonban még nincsenek meg a szükséges adattáblák. Ezek létrehozásához a mnoGoSearch forráskönyvtárában, a ***create/<adatbázisszerver_neve>*** könyvtárban találunk segédanyagot, jelen esetben a ***create/mysql*** alatt. Ezt a szöveges állományt egy az egyben ráengedhetjük a MySQL-re, amely a kívánt adattáblákat létrehozza:

```
mysql -u<felhasznÆl > -p<jelszava> mnogosearch
➥< create.txt
```

Az adatbázis megvan, a táblák rendben, de akad még tennivaló a ház körül. Ha másért nem is, de próbaképpen érdemes beüzemelni a mnoGoSearch saját CGI-felületét. Ehhez a mnoGoSearch telepítési könyvtárának ***bin*** alkönyvtárában található ***search.cgi*** programocskát kell átmásolni az Apache ***cgi-bin*** könyvtárába. Emellett ***search.htm*** néven másoljuk le a mnoGoSearch */etc* könyvtárában levő ***search.htm-dist*** nevű sablonállományt is. Ez azért szükséges, mert ezen arculatmeghatározó állomány nélkül a keresőfelület meg sem mozdul, ráadásul innen veszi a beállításait. Ekkor a keresőfelületnek a ***www.kiszolgálódneve.hu/cgi-bin/search.cgi*** címen működnie kell, de természetesen nem lehet keresni, hiszen még nincs semmilyen adathalmaz. Ráadásul egy kis beállítgatásra is szükség lesz, hiszen a keresőprogram még arról sem tud, merre keresse a háttéradatbázisát. A keresőfelület beállításához nem, de az indexelő robot beállításához szükséges beállításokért felelős állományra szükség lesz. Ez a fájl a sablonfájllal egy könyvtárban található, ***indexer.conf-dist*** néven. Ebből ***indexer.conf*** elnevezéssel szükség lesz egy másolatra, amelyben a mnoGoSearch indexelő motorjának beállításait tudjuk megadni. Furcsának tűnhet, hogy külön beállítóállomány szükségeltetik az indexelést végző programrésznek, külön a keresőfelületnek. Jobban belegondolva viszont belátható, hogy ez a két modul akár két teljesen más gépen is elhelyezkedhet, ráadásul ugyanahhoz az adatbázishoz csak az indexelő részre kell olyan adatbázis-kapcsolat, amelynek a mnoGoSearch adatbázisába írni kell tudnia. A ***search.cgi*** teljesen mellőzhető, hiszen e cikk végcélja, hogy bemutassa, hogyan lehet PHP-ban mnoGoSearch-háttérrel dolgozó keresőfelületeket kialakítani. Mivel amúgy sem árt a PHP és a mnoGoSearch házasítása előtt kipróbálni, hogy a keresőt hajlandó-e működni, és mert a beállítások oroszlánrészét úgyis az ***indexer.conf***-ban kell megadni, érdemes ezt a kis pluszmunkát elvégezni. Lássuk hát, mit és hogyan kell „belőnünk”!
Be kell állítani többek között, hogy melyik kiszolgálón található az adatbáziskiszolgáló, és milyen felhasználónév–jelszó párossal lehet hozzáférni. Kezdjük az ***indexer.conf***-fal! A `DBAddr` címke után az adatbázis-hozzáférés adatait címformátumban kell megadnunk. Szokatlan elgondolásnak tűnhet, hogy egy ilyen erőforrást is ebben a formában adjunk meg, ám mivel ez egy az `ftp://` vagy `http://` címekkel képviselt adatforrással teljesen egyenértékű háttérerőforrást jelöl meg, e megadási

mód abszolút jogos és célravezető:

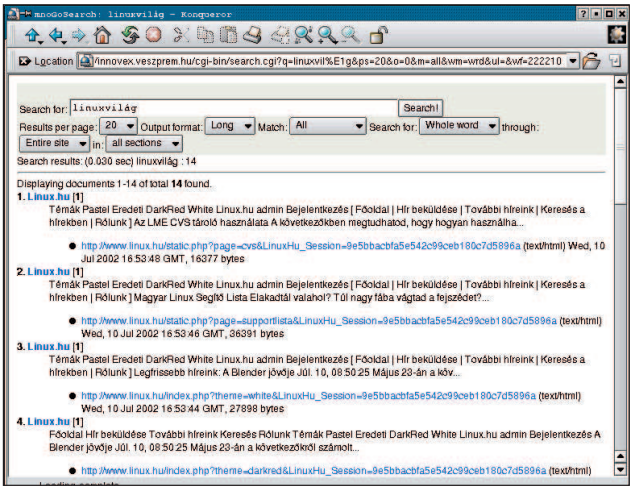
```
DBAddr mysql://felhasznalo:jelszo@localhost/
➥mnogosearch/
```

A cím elején a `mysql://` jelzi, hogy mysql „protokollt” szükséges használni. Ezután a `mnogosearch` adatbázishoz férő MySQL-felhasználó neve és jelszava következik, kettősponttal elválasztva. A @ jelet követően lehet megadni az adatbázis-kiszolgáló gazdagép teljes nevét. Mivel ez nálam ugyanaz a gép, elegendő volt a `localhost`. Itt ha a `mysql` nem a szabványos kapun figyel, egy kettősponttal elválasztva a kapu száma is megadható. Az ezt követő perjel után adható meg az adatbázis neve, ami pár bekezdéssel ezelőtt `mnogosearch` néven lett létrehozva. Ugyancsak ezt az adatsort kell a ***search.htm*** sablonállományban megadni, hogy a keresőfelület is tudja, hol kutakodjon. A megadás módja teljesen megegyező, itt is a `DBAddr` címke a felelős az adatbázis-kapcsolat meghatározásáért. Ezzel mind a ***search.cgi***-t, mind az indexelő programot működőképessé varázsoltuk. A keresőfelület már bármely szóra hibajelzés nélkül képes kijelenteni, hogy nincs rá találat az adatbázisában.

A használhatóság céljából érdemes egy kis figyelmet szentelni a `LocalCharset` címkének, ugyanis itt adható meg, hogy milyen kódolás szerint értelmezzen, valamint hogy milyen karakterek számítanak értékelhető betűnek. E beállítás hiányában az indexelő motor a számára nem kezelhető karakterek mentén a szavakat szétvágja, és a két szórészletet tárolja indexelt szóként. Ez magyar szövegnél annyit jelentene, hogy a hosszú ékezetes karaktert tartalmazó szavaink biztosan több szóként tárolódnának. Példának okáért a „következőek” szó helyett a mnoGoSearch a „következ” és az „ek” szavakra lelne. Állítsuk tehát a `LocalCharset` címke mögötti értéket az iso-8859-2 értékre. Ezt mind az ***indexer.conf***, mind a ***search.htm*** állományban meg kell tennünk.

A ***search.html***-ben megadandó beállításoknak ezzel a végére is értünk. Az egyelőre szóba sem került, miként fogja az indexelő program feldolgozni a kívánt oldalakat, ráadásul azt sem tudja, melyek az indexelhető (azaz a kereshetőnek szánt) webcímek és tartományok. Ahhoz, hogy a honlapokat feldolgozó program tényleges munkát végezhessen, meg kell neki adnunk azokat a belépési pontokat, ahonnan a tartalom indexelését el kell kezdenie. Szintén megadandó, hogy a lapban talált hivatkozások közül melyeket kövesse és melyeket ne. Erre való az ***indexer.conf*** `Server` címkéje. Ezáltal adhatók meg az olyan belépési pontok címei, melyek mentén a weblap feldolgozása megindulhat. A lap címének megadása mellett azt is közölni kell, hogy az esetleges hálóhivatkozások milyen szinten legyenek lekövetve. Ennek négy különféle módja létezik:

- page: semmiféle hivatkozást nem hajlandó követni, csakis ezt az oldalt indexeli. Ritkán van rá szükség.
- path: ez esetben szorgalmas kis pókunk a megadott címen található oldalban csak az olyan hivatkozásokon fog végigszaladni, amelyek a megadott címmel egy könyvtárba vagy az alá mutatnak. Mondhatni, az eme elérési útvonal alatti területet fogja végigzongorázni. Alapértelmezésben, ha külön nem adjuk meg, az indexelő program ezt a fajta hivatkozáskövetési módot fogja alkalmazni.
- site: annyiban különbözik az előzőtől, hogy a teljes webhelyet hajlandó végigkövetni, még akkor is, ha a megadott belépési pont annak csak egy alkönyvtára. Csak akkor megy ki innen, ha a kijutáshoz szükséges hivatkozást talál.




*3. kép* A CGI-felület munkára fogva

Azaz ha a ***http://www.valami.hu/pillanatnyi*** címről indulva hivatkozást talál a ***www.valami.hu***-ra vagy annak bármely alkönyvtárára, vígan követni fogja. Más webhelyekre mutató címeket viszont továbbra sem követ.

- world: ezt a típust választva indexelő programunkat arra utasítjuk, hogy minden fellelt hivatkozást gátlástalanul kövessen. Figyelem, ezzel a típussal érdemes nagyon óvatosan bánni, mert hamar elérhetjük vele gépünk tárhelyének határait, hiszen egy jó kezdőoldalról pók barátunk a fél Internetet végig próbálja majd nyálazni!

Az elméleti alapok után jöjjön egy élő példa! Jómagam pókomat a ***www.linux.hu*** címre engedtem rá, a következő sorral:

```
Server site http://www.linux.hu/
```

Ezzel a sorral pókunk számára megjelöltünk egy belépési pontot, ahol megkezdheti a munkáját. Derék munkásunk ügyesen végig is nyálazza a megtalált hivatkozásokat, csakhogy kizárólag a ***www.linux.hu*** címen belül maradva. Próbaszerencse, együk meg a pudingot. Az indexelés elindításához a mnoGoSearch ***sbin*** könyvtárában az ***indexer*** nevű futtatható állományt kell elindítani. A pók elstartol, és meg sem áll az utolsó oldalig. Amit talál, nem túl sok, mert a ***linux.hu*** főbb alegységei különböző altartományokban helyezkednek el: ***lists.linux.hu***, ***devel.linux.hu*** stb. Ezeket az indexelő – teljesen jogosan – nem követi. De mi módon tehetnénk meg, hogy ezeket is kövesse? Újabb `Server` parancsokkal? Azok nem erre valók, hiszen tartománymegjelölésen túl fő feladatuk a belépési pontok megjelölése. Olyan címtartományokat kell megadni, ahová a keresőpók még beteheti a lábát, s ez a `Realm` címkével lehetséges. Ahhoz, hogy programunk a ***linux.hu***-t,és összes altartományát indexelje, a következő sorra is szükség lesz:

```
Realm http://*.linux.hu/
```

Ez utasítja az indexelőt, hogy bátran kövessen minden olyan hivatkozást, ami megfelel a fenti szűrésnek. A `Realm` alapesetben nem szabályos kifejezésekkel szűr, ehelyett kevésbé erőforrás-igényes megoldással él: a * (csillag) jelent akárhány akármilyen karaktert, a ? (kérdőjel) pedig egy tetszőleges karaktert. A legtöbb címszűrést ennyivel el is lehet intézni, néha azonban a szabályos kifejezésekre is szükség lehet. A fenti sort behelyettesíthetjük a következővel:

```
Realm Regex ^http://.*\.linux.hu/
```

Ez pontosan ugyanazt a szűrést fogja elvégezni, mint az eggyel feljebbi. Szabályos kifejezést ennél sokkal összetettebb feladatokra érdemes alkalmazni. Ha ezen beállításokkal élve indítanánk el az „indexer”-t, kis hivatkozáskövető robotunk már jelentős mennyiségű oldalon menne keresztül, de ne pazaroljuk a ***linux.hu*** erőforrásait. Ha egy mód van rá, az indexelő robot figyelmét inkább egy saját weblapra irányítsuk!
A feldolgozás végeztével, de akár közben is, böngészőnkkel ismét meglátogathatjuk a ***search.cgi***-t, és lőn csoda: ha jó szót írunk be, előbukkan a találati lista. Hurrá, pezsgőbontás, tűzijáték! A keresőmotor telepítése és élesztése sikeresen lezajlott. Korántsem lehet elégedetten hátradőlni a karosszékben, hiszen egészen idáig egy mukkot sem ejtettem arról, miként fog a mnoGoSearch és a PHP együttműködni.
Ahhoz, hogy a hőn áhított együttműködés létrejöjjön, a PHP-t természetesen a mnoGoSearch-támogatással kiegészítve újra kell fordítani. Ezért akár az egész PHP is újrafordítható, de vélhetőleg a kiszolgálón futó PHP-programocskáink jelentéktelen hányada fogja a PHP-nak ezt a modulját dolgoztatni. És biza, ha befordítjuk, minden PHP-fájl futtatásakor ott lesz a memóriában, amiből pedig sosem elég. Fontos még, hogy gépünkön a ***libz.so*** függvénykönyvtár telepítve legyen, és a PHP is `-with-zlib` beállítással legyen lefordítva. A `zlib` Debianon az `apt-get install libz` parancssorral ott is volt a gépen, ezek után a PHP újrafordítása is könnyedén lezajlott. Ha már újrafordítás, a `--with-mnogosearch` kapcsolót is be lehetne illeszteni, de a fenti takarékossági megfontolásból nem tettem meg. Kinek hogy tetszik: a PHP újrafordítható, ha az eredeti beállítások mellé a mnogosearch beillesztését is felvesszük. Az újrafordítás megkezdése előtt ne feledjünk el egy `make clean`-t is kiadni, és a ***config.cache*** állomány törlése is hasznos. A mnogosearch-kiegészítés külön lefordítása sem ördöngösség, ehhez mindössze a PHP forráskönyvtárából a `./configure -with-mnogosearch=shared` parancsra, majd egy `make`-re lesz szükség. Ezzel a műveletsorral a forráskönyvtár ***modules*** alkönyvtárában egy ***mnogosearch.so*** nevű állományt kapunk vissza, ezt kell beraknunk a PHP kiterjesztő moduljait tartalmazó könyvtárba. E könyvtár elhelyezkedése egy `phpinfo()`-ból `extension_dir` néven megtudható, de a ***php.ini***-ben természetesen meg is adható. A lényeg a végeredmény, a ***mnogosearch.so*** az `extension_dir`-nek megfelelő könyvtárba kell kerüljön. Innentől amelyik PHP-parancsfájlban szükségünk lesz a mnoGoSearch-kezelő függvényekre, be kell szúrni egy

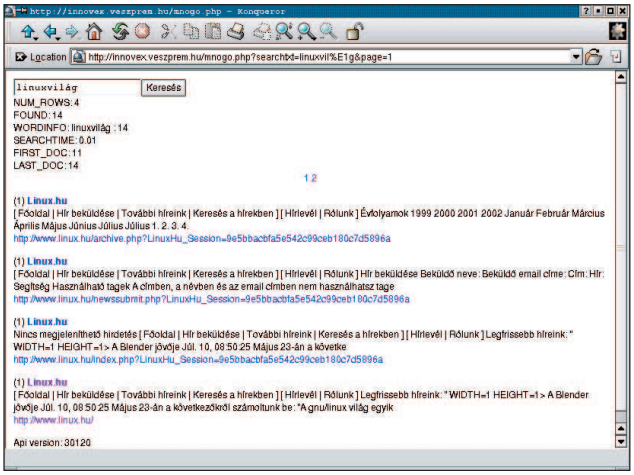
```
dl( mnogosearch.so );
```

sort az elejére, és rendszerünkbe máris behúztuk a megfelelő kiterjesztést.

## Együtt a PHP-val

Most, hogy minden együtt van, minden működik, lássuk, mi módon tudjuk kisajtolni a kereséseket a mnoGoSearch adatbázisából a PHP erre szakosodott, immár telepített utasításain keresztül. Ahhoz, hogy a mnoGoSearch keresőjében lekérdezéseket hajthassunk végre, szükség lesz egy kapcsolódásra. Ez a modul is a szokásos utat követi, ahogyan azt a fájl- vagy az adatbázis-kezelésben megszokhattuk. Először le kell foglalnunk egy adott típusú erőforrást, hogy aztán ennek az azonosítójával hivatkozva végezhessünk műveleteket. Az első lépés mindig az `udm_alloc_agent()` függvény meghívása lesz, az utolsó szó joga pedig minden esetben az `udm_free_agent()`-et illeti.

E második függvénnyel tudjuk elvégezni munkánk végén a takarítómunkát – némi hasznos memória visszaadásával a rendszernek. Ennek egyetlen értéke az azonosító, amit a kapcsolat létrehozásakor az `udm_alloc_agent()`-től kapunk. A függvény egyetlen kötelező értékében a kapcsolódási adatokat kell megadnunk, melyek megadási formája teljes mértékben megegyezik a mnoGoSearch beállítási fájljaiban használatos `DBAddr`-féle címformával. Érdemes megemlíteni, hogy az itt megadott adatbázisfelhasználó–jelszó páros nem kerül kiértékelésre, csak ha az `udm_find()` függvény meghívásával érdemi keresést kezdeményezünk. Ezért ne csodálkozzunk, ha az esetleges csatlakozási hiba nem itt, az erőforrás foglalásakor keletkezik, hanem pár utasítással később. Nagyon fontos, ezért külön kiemelendő, hogy az adatbázis-hozzáférést megadó címet egy / (perjellel) zárjuk, enélkül ugyanis a kapcsolat nem fog létrejönni.




*4. kép* Keresés, immár PHP-val

Az előbb említett `udm_find()` függvény által tudunk kereséseket végrehajtani. Két értéket vár, az elsőben a keresés kezdeményezése előtt lefoglalt erőforrás azonosítóját kell megadni, másodikként magát a keresőkérést. Egy keresés megadásakor a keresendő szavak szóközzel elválasztott listáján felül egyéb trükkökkel is élhetünk: a különféle szavakat zárójelekkel csoportosíthatjuk, köztük logikai műveleteket is végrehajthatunk. Az & (és jel) a logikai és-t, míg a | (csőjel) a logikai vagyot hivatott jelképezni. Megjegyzem, a sima szavak felsorolása is olyan, mintha logikai és jellel választottam volna el őket egymástól. Azaz a `linux php` keresőkulcs ugyanazt az eredmény fogja adni, mint a `linux&php`. Ugyanakkor a `linux|php` olyan találatokat hoz, amelyekben vagy a linux vagy a php szó megtalálható. A kizárt szavak megjelölésére a keresőkben általában megszokott kizáró - (mínuszjel) helyett itt a ~-t (tildejelet) állították hadrendbe. Az ilyen jellel előjelzett szavakat tartalmazó dokumentumok e lehetőség kihasználásával nem kerülnek a találati listába. Megjegyzem, önmagában egy `~php` keresési kulcs megadása nem értelmezhető, nem fogja az összes olyan találatot kiadni, amelyben nem szerepel a php szó. Ez csak ara való, hogy a meglévő találati listából kizárja a nem megfelelőket. Az alábbi tehát működik: `linux ~php`, ami által olyan dokumentumok kerülnek a találati listára, melyek Linux témába vágóak, ámde amelyekben szó sem esik a php-ról. Ha végül sikerült keresést kezdeményeznünk, eredményéhez egy azonosítón keresztül férhetünk hozzá. Ezt az azonosítót az `udm_find()` adja meg számunkra, visszatérési értékként.

A keresőoldal

```
<?

define('_PERPAGE',10);

if (!extension_loaded('mnogosearch.so')) {
    dl('mnogosearch.so');
}

echo '
<form action="'.$PHP_SELF.'">
<input type=text name="searchtxt"
➥value="'.$searchtxt.'">
<input type="submit" name="start_search"
➥value="KeresØs">
</form>';


$agentid = udm_alloc_agent
➥('mysql://nØv:jelsz @localhost/mnogosearch/');
udm_set_agent_param
➥($agentid,UDM_PARAM_CHARSET,'iso-8859-2');
udm_set_agent_param
➥($agentid,UDM_PARAM_PAGE_SIZE,_PERPAGE);
udm_set_agent_param
➥($agentid,UDM_PARAM_PAGE_NUM,$page);

$sres = udm_find($agentid,$searchtxt);

$num_rows = udm_get_res_param
➥($sres, UDM_PARAM_NUM_ROWS);
$num_found = udm_get_res_param
➥($sres, UDM_PARAM_FOUND);

echo 'NUM_ROWS: '.$num_rows.'<br>';
echo 'FOUND: '.$num_found.'<br>';
echo 'WORDINFO: '.udm_get_res_param
➥($sres, UDM_PARAM_WORDINFO).'<br>';
echo 'SEARCHTIME: '.udm_get_res_param
➥($sres, UDM_PARAM_SEARCHTIME).'<br>';
echo 'FIRST_DOC: '.udm_get_res_param
➥($sres, UDM_PARAM_FIRST_DOC).'<br>';
echo 'LAST_DOC: '.udm_get_res_param
➥($sres, UDM_PARAM_LAST_DOC).'<br>';


$maxpage = ceil($num_found / _PERPAGE);
$page = floor($page);
$page = ($page > $maxpage) ? $maxpage:$page;

echo '<center>';
for ($i=0; $i<$maxpage; $i++) {
    echo '
    <a href="'.$PHP_SELF.'?searchtxt=
    ➥'.urlencode($searchtxt).'&page='.($i).'">
    ➥'.($i+1).'</a>';
}
echo '</center>';

for ($i=0; $i<$num_rows; $i++) {

    $hit_url = udm_get_res_field
    ➥($sres,$i, UDM_FIELD_URL);
    $hit_title = udm_get_res_field
    ➥($sres,$i, UDM_FIELD_TITLE);
    $hit_text = udm_get_res_field
    ➥($sres,$i, UDM_FIELD_TEXT);
    $hit_rating = udm_get_res_field
    ➥($sres,$i, UDM_FIELD_RATING);

    echo '
    <p>
    ('.$hit_rating.')
    <a href="'.$hit_url.'"><b>'.$hit_title.'
    ➥</b></a><br>'.$hit_text.'<br>
    <a href="'.$hit_url.'">'.$hit_url.'
    ➥</a><br>';

}

echo '<p>Api version: '.udm_api_version().'<br>';

?>
```

Miféle adatok nyerhetők ki ebből az azonosítóból? Az `udm_get_res_param()` a keresés eredményére általánosan jellemző adatokat adja vissza, míg az `udm_get_res_field()` segítségével a találatok adatainak kinyerése lehetséges. Kezdjük az általános adatokkal! Ezek megszerzéséhez első értékként a keresési eredményt megjelölő azonosítót kell megadni, tehát azt, amit az `udm_find()` kiadásakor kaptunk, másodiknak pedig azt kell megadnunk, hogy mely adatra van szükségünk. Ezek az adatok a következők:

- UDM_PARAM_NUM_ROWS: a listázásra kerülő találatok száma.
- UDM_PARAM_FOUND: az összes találatok száma. Mivel a találatok megadható méretű oldalakra bontódnak szét, az eggyel ezelőtti adat nem feltétlenül egyezik meg ezzel, ami valóban megadja, hány találat sikeredett.
- UDM_PARAM_WORDINFO: szöveges formában adja vissza, hogy az egyes szavakra külön-külön hány találat érkezett.
- UDM_PARAM_SEARCHTIME: századmásodperc pontossággal a keresési idő.
- UDM_PARAM_FIRSTDOC: az éppen listázandó oldal első találatának sorszáma.
- UDM_PARAM_LASTDOC: az éppen listázandó oldal utolsó találatának sorszáma.

Lapok? Bizony-bizony, a `mnoGoSearch` modul a találatokat lapokra osztja. Ha másképp nem rendelkezünk, az első oldalt fogjuk visszakapni, és egy oldalon húsz találat megjelenítésére kapunk módot. Ezen természetesen lehet változtatni, csupán az `udm_set_agent_param()` függvényt kell szorgalmasan hívogatni. Pár példa a teljesség igénye nélkül:

```
// 10 talÆlat legyen 1 oldalon:
udm_set_agent_param($aid,UDM_PARAM_PAGE_SIZE,10);
```

```
// UgrÆs a harmadik oldalra
// (az elsı oldal szÆma a 0)
udm_set_agent_param($aid,UDM_PARAM_PAGE_NUM,2);
// Ennek megadÆsa nagyon fontos, mert enØlk l
// a mnoGoSearch hibÆsan keresi a megadott
// Økezetes szavakat!
// Meg kell egyeznie az indexer.conf-ban
// megadott LocalCharset ØrtØkkel.
udm_set_agent_param($aid,UDM_PARAM_CHARSET,
➥'iso-8859-2');
```

A fentieken túl ezen a függvényen keresztül rengeteg beállítást lehet megadni, de helyszűke miatt erre most nincs lehetőség, viszont a ➲ http://hu.php.net/mnogo címen a teljes lista elérhető. Térjünk inkább rá, hogyan is lehet a találatokat kipréselni, hogy azután gusztusos formában tálalhassuk a nyájas látogató elé. Erre a korábban már említett `udm_get_res_field()` hivatott. Első értékként a keresési eredménylistára hivatkozó azonosítót kell megadni, másodikként pedig a találat oldalon belüli sorszámát. Figyelem, a számozás nulláról indul! A harmadikkal megmondjuk, melyik adatra van szükségünk az adott találati sorral kapcsolatban, lehetséges értékei közül a fontosabbak:

- UDM_FIELD_URL: a talált oldal webcíme.
- UDM_FIELD_TITLE: a meglelt weboldal `<title>` `</title>` címkéi közt megadott oldal elnevezés.
- UDM_FIELD_DESC: a meglelt oldal leírása. Ezt a mnoGoSearch az oldal `Description`-típusú metaeleméből nyeri ki.
- UDM_FIELD_TEXT: az oldal szövegének első pár sora. A fenti `Description` hiányában érdemes kijelezni, hogy a keresést kezdeményező némi rálátással bírhasson, ez-e a neki megfelelő oldal.

## Végszó

Nagyjából ennyi azoknak a PHP-utasításoknak a köre, amelyek alkalmazása elengedhetetlen, ha ezen a módon akarunk kereséseket végrehajtani. Összefoglalóként lássuk mindezt egyben is, egy igen egyszerű keresőoldal példáján keresztül, melyet a 66. oldalon tanulmányozhatnak részletesebben.

*Heilig (Cece) Szabolcs* (cece@php.net) Veszprémben él. Hobbija, kedvenc időtöltése és munkája is a programozás. Szabadidejében hajlamos kerékpárra pattanni, vagy baráti társaságban szerepjátékokkal foglalatoskodni.

**Kapcsolódó címek**

A keresőmotor hivatalos lapja
➲ http://www.mnogosearch.org
A mnoGoSearch PHP-modul leírása
➲ http://hu.php.net/mnogo