Adatbázisok egységesített elérése PHP-val

Írásunkban két olyan módszert tárgyalunk, amelyekkel egyszerűbben férhetünk hozzá adatbázisainkhoz: a PEAR::DB-t és az ADODB-t.

Amennyiben foglalkoztál már PHP-val, a PEAR név biztosan ismerősen cseng. Magyarul körtét jelent, de PHP-s körökben hallatán inkább az e névvel fémjelezett alkalmazáscsomagra gondol az ember. Tapasztaltabb felhasználóknak az ADODB név is ismerősen csenghet. Cikkünkben a PEAR::DB és az ADODB nevű egységesített adatbázis-elérési rétegekkel fogunk megismerkedni. Mindkét programcsomag egy egyszerűnek tűnő, de eddig megoldatlan gondot orvosol. Tegyük fel, hogy az egyik délután írtál egy néhány oldalas PHP-kódot, amellyel a cég MySQL-adatbázisában kell valamilyen műveletet elvégezned. Tételezzük fel azt is, hogy nap mint nap ilyen egyszerűbb kódokkal foglalatoskodsz. Hónapok, esetleg évek során egész szép mennyiségű kód gyűlik össze, benne kisebb-nagyobb programokkal, amelyekben csak az a közös, hogy egy adatbázis-kiszolgálón végeznek különböző műveleteket. A programok száma egyre nő, a cég ügyfélköre is folyamatosan gyarapodik, egy szép napon pedig – amikor már nem tudod tovább állítgatni az adatbázis-kiszolgálót – rádöbbensz, hogy a jelenlegi rendszer teherbírása már nem elégséges. A cég vezetősége úgy dönt, hogy ezt elhárítandó egy Oracle adatbázis lenne a megfelelő. És innentől kezdve csőstől jönnek a nehézségek: az adatbázis megváltoztatásával az összes eddigi alkalmazás alól – mondhatni – kihúzod a szőnyeget. Minden egy csapásra működésképtelenné válik.

Ha nem voltál elég előrelátó, és ilyen helyzetben találod magad, az elkövetkező hónapok lehet, hogy a korábbi programok újraírásával fognak telni. Ha ellenben erre az eshetőségre már a legelején is gondoltál, előfordulhat, hogy mindössze egyetlen fájlban kell egy változó értékét módosítanod. Amennyiben előretekintő vagy, a fenti módosítással a cég – és egyben az adatbázis – fejlődése akadálytalanul folyhat tovább, és neked nem kell mást tenned, mint amit eddig: az adatbázisból értékeket kihalászni, megjeleníteni és olykor módosítani. Ha viszont balszerencsésnek bizonyulsz, nincs más választásod, mint ettől a pillanattól kezdve elszakadni az, eddig használt adatbázisodtól, és egy adatbázis-független kód írásába kezdeni, szabadidődben pedig a korábban felhalmozódott kódtömeg rendbetételével foglalatoskodni.

Mostanra valószínűleg mindenki számára világossá vált, miért is kell a PHP alapértelmezett adatbázis-felületét úgy ahogy van elfelejteni. Lássuk hát a megoldást!

A PEAR

A PEAR egy nyílt forráskódra épülő PHP-programkönyvtár, mely PHP- és C-bővítményeket egyaránt tartalmaz. Hasonló a Perl CPAN-jához. Az teszi mégis különlegessé, hogy a PEAR-ben található bővítményeknek egy sor szabálynak meg kell felelniük. Ezek a szabályok ugyanúgy vonatkoznak a kódolási stílusra, a kód tagoltságára, a bekezdések elhelyezkedésére és a behúzások nagyságára, mint az alkalmazások egységesített programozói felületére, valamint a megjegyzések elhelyezkedésére és szerkezetére. Ezeknek a feltételeknek mind a programkönyvtárhoz kapcsolódó C-programoknak, mind a PHP-programoknak meg kell felelniük. Egy alkalmazás csak akkor lehet a programcsomag része, ha minden elvárásnak eleget tesz.

1. lista A MySQL-adatbázis létrehozása (db.sql)

```
CREATE DATABASE konyvesbolt;
USE konyvesbolt;

CREATE TABLE konyvek (
  sorszam smallint NOT NULL auto_increment,
  darabszam tinyint,
  konyvcim varchar(20),
  PRIMARY KEY (sorszam)
);

INSERT INTO konyvek (darabszam, konyvcim)
➥VALUES(10, "AkiØrt a harang sz l");
INSERT INTO konyvek (darabszam, konyvcim)
➥VALUES(15, "Egri csillagok");
INSERT INTO konyvek (darabszam, konyvcim)
➥VALUES(20, "Zabhegyezı");
INSERT INTO konyvek (darabszam, konyvcim)
➥VALUES(8, "A PÆl utcai fiœk");
```

A PEAR telepítéséhez először a csomagkarbantartó programot kell letöltenünk és telepítenünk a következő paranccsal:

```
lynx -source http://pear.php.net/go-pear | php4
```

Megeshet, hogy a PHP4 rendszerünkön a `php` paranccsal érhető el, ez esetben a következőt kell beírnunk:

```
lynx -source http://pear.php.net/go-pear | php
```

Mindkettőhöz az szükségeltetik, hogy rendszerünkön a PHP4 CGI-csomagja elérhető legyen, akkor is, ha a PHP4-et nem CGI-ként, hanem Apache-bővítményként futtatjuk. A csomag neve ***php4-cgi***, ami Debian rendszereken az `apt-get install php4-cgi` paranccsal egyszerűen telepíthető, más rendszerek esetén pedig az adott Linux-terjesztéshez adott CD-k valamelyikén találjuk meg. A parancs futtatása után fogadjuk el a felajánlott beállításokat, és kérjük az alapértelmezett csomagok telepítését. Ha rendszerünkön nem a PHP4 CGI-változatát használjuk, a ***/etc/php4/cgi/php.ini*** megváltoztatása nem szükséges, ellenben a ***/etc/php4/apache/php.ini*** fájlban az ***include_path*** nevű változót módosítanunk kell, hogy tartalmazza a PEAR könyvtárát. Ha a kérdéses sor eddig így nézett ki:

```
include_path = "."
```

2. lista A beállításokat tartalmazó fájl (config.inc.php)

```
1. <?php
2.
3. $dbhost = 'localhost';
4. $dbuser = 'fulesmacko';
5. $dbpass = 'mezesbodon';
6. $dbname = 'konyvesbolt';
7.
8. ?>
```

3. lista A MySQL közvetlenül (mysql.php)

```
1. <?php
2.
3. include('config.inc.php');
4. // ...
5. $db = mysql_connect($dbhost, $dbuser,
   $dbpass);
6. // ...
7. mysql_select_db($dbname);
8. $sql = "SELECT * FROM konyvek";
9. $res = mysql_query($sql);
10. while ($row = mysql_fetch_row($res))
11.     printf("%s (%ddb)<br>\n", $row[2],
       $row[1]);
12.
13. printf("K nyvek szÆma: %d<br>\n",
    mysql_num_rows($res));
14. mysql_close($db);
15.
16. ?>
```

akkor erre kell cserélnünk:

```
include_path = "/usr/share/pear:."
```

A telepítést követően rendszerünkön rendelkezésre áll a `/usr/bin/pear` nevű futtatható fájl, amellyel a PEAR bővítményeit kezelhetjük. A rendelkezésre álló bővítmények listája ➲ http://pear.php.net/packages.php címen érhető el, de kipróbálhatjuk az alábbi a parancsot is:

```
pear list-remote-packages
```

A listából kiválasztott elemet a

```
pear install
➥http://pear.php.net/get/csomag_neve
```

paranccsal telepíthetjük fel, vagy ha a szükséges csomagot már letöltöttük, a következő is elég lesz:

```
pear install csomag.tgz
```

A Debian-változat legújabb terjesztéseiben a PEAR, illetve az alapértelmezett csomagok az `apt-get install libphp-pear` paranccsal is telepíthetők.

4. lista A PEAR::DB-példa (pear-db.php)

```
1. <?php
2.
3. include('config.inc.php');
4. require_once('DB.php');
5. $db = DB::connect
   ("mysql://$dbuser:$dbpass@$dbhost/$dbname");
6. // ...
7. // erre itt nincs sz ksØg (select_db)
8. $sql = "SELECT * FROM konyvek";
9. $res = $db->query($sql);
10. while ($row = $res->fetchRow())
11.     printf("%s (%ddb)<br>\n", $row[2],
       $row[1]);
12.
13. printf("K nyvek szÆma: %d<br>\n",
    $res->numRows());
14. $db->disconnect();
15.
16. ?>
```

5. lista Példák adatbázis-azonosítókra

```
dbt pus://felhnØv:jelsz @protokoll+gØpnØv:
➥555//usr/dbnØv.db
dbt pus://felhnØv:jelsz @gØpnØv/dbnØv
dbt pus://felhnØv:jelsz @gØpnØv
dbt pus://felhnØv @gØpnØv
dbt pus://gØpnØv/dbnØv
dbt pus://gØpnØv
dbt pus (vÆltozat)
dbt pus
dbt pus://felhnØv:jelsz @tcp(localhost:555)/
➥dbnØv
dbt pus://felhnØv:jelsz @unix
➥(/var/run/dbsocket)/dbnØv
```

Előkészületek

Mielőtt belevágnánk programjaink és elveink megreformálásába, nézzük csak meg, hogyan dolgoznánk, ha adatbázisainkat a PHP felületén keresztül közvetlenül kezelnénk. Ebben az esetben az első dolgunk az, hogy meghatározzuk, milyen adatbázist szeretnénk használni, mert ettől függ a kód többi része. Mi most a MySQL-t vesszük alapul, tekintve, hogy ez az egyik legnépszerűbb adatbázisrendszer (PostgreSQL-rajongóknak: csak a PostgreSQL után!). Hozzunk tehát létre a MySQL-ben egy új adatbázist, és abban egy példatáblát (lásd *1. lista*).
Ha ezzel megvagyunk, készítsünk egy egyszerű eljárást, amellyel adatbázisunk tartalmát kiíratjuk (*2.* és *3. lista*). Ha ez is kész, továbbléphetünk, és megtudhatjuk, hogyan zajlik mindez a függetlenített adatbázis-elérési módozatokkal.

Adatbázis-kezelés PEAR::DB módra

A *4. listán* a MySQL-es példa látható kissé átalakítva. Az óriási különbség a két programkód között az, hogy míg az első esetben a kód mondjuk Oracle-re történő átültetése legalább 5–10 percünkbe kerülne, addig ebben az esetben nem tart tovább néhány másodpercnél. Nem is beszélve arról, hogy az

6. lista A PEAR::DB által támogatott adatbázistípusok

```
mysql   -> MySQL
pgsql   -> PostgreSQL
ibase   -> InterBase
msql    -> Mini SQL
mssql   -> Microsoft SQL Server
oci8    -> Oracle 7/8/8i
odbc    -> ODBC (Open Database Connectivity)
sybase  -> SyBase
ifx     -> Informix
fbsql   -> FrontBase
```

Oracle PHP-s felületének leírását is át kellene rágnunk, mielőtt belekezdünk az új kód megírásába. Itt mutatkozik meg igazán, micsoda kényelmet jelent egy egységesített felület használata.

Vegyük szépen sorra, mit jelentenek az egyes sorok! Először is a 3. sorban beillesztjük az adatbázissal kapcsolatos beállításokat tartalmazó fájlt – ez egyértelmű. A 4. sorban esetleg a `require_once()` tűnhet furcsának. A `_once` módosító arra hivatott, hogy ha a programunk egy más részén már beszúrtuk a kérdéses fájlt, programunkat esetlegesen megzavarva ne szúródjék be újra. Az `include()` és a `require()` között egyébként csak annyi a különbség, hogy az `include()` csak figyelmeztet, ha nincs meg a keresett fájl, viszont a `require()`, ha nem találja a fájlt, a program futását le is állítja. Az első lényeges különbségek az 5–7. sorban mutatkoznak meg. Míg a PHP alapfelületével szükséges volt külön kijelölni az adatbázist, PEAR::DB esetében egy különleges DSN-t, vagyis adatbázis-azonosítót kell megadnunk. A DSN-nek csak a neve ijesztő, használata jóval egyszerűbb, mint ahogyan azt az *5. lista* egyértelműen megmutatja. A *6. lista* a PEAR::DB által támogatott adatbázisokhoz tartozó kulcsszavakat tartalmazza, amelyeket a DSN-ek megadásánál vehetünk igénybe.

A 8. és 9. sorban sem találhatunk lényeges újdonságokat: megadjuk a lekérdezést, majd a `$db` objektumon keresztül futtatjuk. A 10. sor már tartalmaz néhány újdonságot, itt tűnik fel a `fetchRow()` tagfüggvény is. Ennek a feladata, hogy az adatbázistól sorban lekérdezze az SQL-utasításra válaszul érkezett mezőket. A 13. sorban található `numRows()` tagfüggvény a lekérdezés által visszaadott sorok számát jeleníti meg, ezt követően pedig a `disconnect()` tagfüggvénnyel leválunk az adatbázisról.

Az adatbázis módosítása az `INSERT`, `UPDATE` és `DELETE` utasításokkal ugyanúgy működik, ahogy korábban megszoktuk, csak ebbe az új környezetbe van beleültetve.

A `fetchRow()` függvényről még tudnunk kell valamit. Sokkal barátságosabbá tehetjük programunkat, hogyha számok helyett az adatbázis egyes mezőire a nevükkel hivatkozunk. Ennek beállítására szolgál a `fetchRow()` tagfüggvény első és egyben el is hagyható kapcsolója, amely alapértelmezésben a `DB_FETCHMODE_ORDERED` értéket veszi fel. Ha azonban az eddigiekkel ellentétben a `DB_FETCHMODE_ASSOC` vagy `DB_FETCHMODE_OBJECT` értékekkel hívjuk meg, a mezőkre a nevükkel is hivatkozhatunk: az első esetben egy nevesített tömb segítségével, az utóbbinál pedig objektumhivatkozáson keresztül. Például:

```
$row = $res->fetchRow(DB_FETCHMODE_ASSOC);
print($row[ konyvcim ]);
```

vagy:

```
$row = $res->fetchRow(DB_FETCHMODE_OBJECT);
print($row->konyvcim);
```

Ha a `fetchRow()`-nak nem akarjuk minduntalan megadni az általunk előnyben részesített hivatkozási mód nevét, a kö-

7. lista A prepare(), az execute() és a quote() (pear-db-2.php)

```
1. $ujkonyvek = array("AAAA", "BBBB", "CCCC");
2.
3. $sql = "INSERT INTO konyvek (darabszam,
   konyvcim) VALUES(10, ?)";
4. $query = $db->prepare($sql);
5.
6. foreach ($ujkonyvek as $ujkonyv)
7.     $res = $db->execute($query,
       $db->quote($ujkonyv));
8.
9. $ujkonyvek2 = array();
10. $ujkonyvek2[] = array(12, "SzØp
    remØnyek");
11. $ujkonyvek2[] = array(21, "Bœcsœ a
    fegyverektıl");
12.
13. $sql = "INSERT INTO konyvek (darabszam,
    konyvcim) VALUES(?, ?)";
14. $query = $db->prepare($sql);
15.
16. foreach ($ujkonyvek2 as $ujkonyv2)
17.     $res = $db->execute($query,
       DB->quote($ujkonyv2));
```

8. lista Az ADODB-példa (adodb.php)

```
1. <?php
2.
3. include('config.inc.php');
4. require_once('adodb.inc.php');
5. $db = NewADOConnection("mysql");
6. $db->Connect($dbhost, $dbuser, $dbpass,
   $dbname);
7. // erre itt nincs sz ksØg (select_db)
8. $sql = "SELECT * FROM konyvek";
9. $res = $db->Execute($sql);
10. for (; !$res->EOF; $res->MoveNext())
11.     printf("%s (%ddb)<br>\n",
       $res->fields[2], $res->fields[1]);
12.
13. printf("K nyvek szÆma: %d<br>\n",
    $res->RecordCount());
14. $db->Close();
15.
16. ?>
```

vetkező paranccsal alapértelmezetté tehetjük:

```
$db->setFetchMode(DB_FETCHMODE_ASSOC);
```

Hibakezelés

A `PEAR::DB isError()` statikus – vagyis nem objektumhoz kötött – tagfüggvényével ellenőrizhető, hogy az elvégzett művelet során történt-e valamilyen hiba:

```
$db = DB::connect($DSN);

if (DB::isError($dB))
      die($db->getMessage());
```

A hiba leírásával a `getMessage()` tagfüggvény tér vissza. Az adatbázis-lekérdezésnél előforduló hibákat az alábbi módon ellenőrizhetjük:

```
$res = $db->query($sql);

if ($db->isError($res))
      die(DB::errorMessage($res));
```

A `PEAR::DB`-nek egy önműködő szerkezete is létezik a hibák kezelésére. Erről bővebben a ➲ http://pear.php.net/manual/en/ címen olvashatsz, a `DB::setErrorHandling()` tagfüggvénynél.

További lehetőségek

Amennyiben több hasonló módosítást szeretnénk végrehajtani egy adatbázison, jó szolgálatot tehet nekünk a `prepare()` és az `execute()` tagfüggvény. Mindkettő ismerős lehet, hiszen a Perl DBI programozása során már találkozhattunk velük. Vess egy pillantást a *7. listára*!
A `prepare()` tagfüggvénnyel egy lekérdezést többszöri használatra lehet felkészíteni, a lekérdezésben változó értékeket pedig az `execute()` tagfüggvény egymás után következő meghívásai során adjuk meg. Az `execute()` első értéke a `prepare()` által visszaadott `$query` változó, melyet az értéket tartalmazó változó vagy – több érték esetén – az értékeket tartalmazó tömb követ.
A `quote()` függvény szerepe nagyon fontos: a segítségével biztosíthatjuk ugyanis, hogy az adatbázisba kerülő értékek mindig az adott adatbázisnak megfelelően kódolódjanak. Az „ez egy 'A' betű" karakterláncban az aposztrófot a MySQL esetén egy visszafelé tört vonallal védenénk le („ez egy \'A\' betű"), míg MS Accessben vagy Sybase-ben egy másik aposztróffal („ez egy ''A'' betű"). Egy jól megtervezett adatbázis-egységesítési rendszerben az ilyesmikre is gondolni kell!

A PEAR::DB hátulütői

A PEAR::DB egyik legnagyobb hibája minden bizonnyal az, hogy még igencsak fejlesztés alatt áll, következésképpen néhány dolog nem működik benne, vagy nem úgy működik, ahogy az ember elvárná. A PEAR::DB-ből egyelőre az adatbázisrendszerek kezeléséhez elengedhetetlen dátumkezelés egységesítését megvalósító függvények is hiányoznak. Erre feltétlenül szükség van ahhoz, hogy valódi adatbázis-független kódot írjunk, így remélhetőleg a közeljövőben ennek a megalkotására is sor kerül. Addig pedig – ha a PEAR::DB mellett döntünk – saját magunk alkotta függvényekkel vagyunk kénytelenek elvégeztetni ezt a feladatot.

Bemutatkozik az ADODB

Az ADODB ugyanazt a célt szolgálja, mint a PEAR DB nevű csomagja. Működési elve és programozói felülete is hasonló, a különbség csupán annyi, hogy az ADODB jóval kiforrottabb kódra épül, és számtalanul többen használják, mint a PEAR::DB-t. Az is az ADODB mellett szól, hogy jelenleg sokkal több adatbázisrendszert támogat, mint a PEAR::DB.
A PHP MySQL-ről átalakított példaprogramunk ADODB-s változata a *8. listán* látható. Az elgondolás itt is ugyanaz, mint a PEAR::DB esetében, és a két felülethez tartozó kód – a kezelést illetően – lényegében ugyanaz, eltekintve a megvalósítás részleteiben rejlő különbségektől. Az első különbség az, hogy az ADODB esetében a csatlakozás két sorba tagolódik. A második szembetűnőbb: a `for`-t használjuk az eddig megszokott `while` helyett. Ez azért szükséges, mert az ADODB az egyes sorok lekérdezése után a sormutatót nem önműködően lépteti tovább, azt magunknak kell a `MoveNext()` tagfüggvény meghívásával megtenni.

Összegzés

Az ADODB és a PEAR::DB két jól átgondolt, adatbázis-kezelést egységesítő rendszer, melyek használata minden komoly fejlesztő és cég számára mindenképpen indokolt. Jelen pillanatban az ADODB használata fejlettségének köszönhetően elterjedtebb, ugyanakkor a PEAR::DB a PHP PEAR projektjének a része – idővel valószínűleg ez fog fölénybe kerülni. Természetesen csak akkor, ha az ADODB-t mind az adatbázisok támogatottságában, mind a leírás használhatóságában utolérik. Az ADODB-hez nagyon jó leírás tartozik, míg a PEAR-hez a szép és ügyes PHPDoc-ra épülő – ez azonban egyelőre sajnos még nagyon hézagos: teljesen hiányoznak belőle a példaprogramok, a tagfüggvények leírása sokszor egy-egy mondatra korlátozódik, nem is beszélve arról, hogy a dolgok a valóságban gyakorta nem úgy működnek, mint ahogyan elképzelték. Bármelyik megvalósítás mellett döntünk is, végeredményben csak jól járhatunk.

Gludovátz Gábor (ggabor@sopron.hu) egy soproni cég Linux-rendszerekkel foglalkozó rendszergazdája. Kedvenc időtöltése a programozás, és a Linux lelkivilágában való kutakodás. Ha ideje engedi, szívesen hódol szenvedélyének és bringáján a környező erdőket járja. Honlapja a ➲ http://www.sopron.hu/~ggabor/ címen érhető el.

Kapcsolódó címek

PEAR ➲ http://pear.php.net
ADODB ➲ http://.php.weblogs.com/ADODB/
PEAR-leírás ➲ http://pear.php.net/manual/en/
Különféle PEAR::DB-oktatók címei
➲ http://vulcanonet.com/soft/?pack=pear_tut
➲ http://www.phpbuilder.com/columns/allan20010115.php3?page=1
➲ http://evolt.org/article/Abstract_PHP_s_database_code_with_PEAR_DB/17/21927/index.html
➲ http://www.devshed.com/Server_Side/PHP/DBAbstraction/page1.html