A Zope és az adatbázisok

Reuven bemutatja, hogy egyszerű Zope-helyünket milyen könnyedén át lehet alakítani oly módon, hogy az adatokat relációs adatbázisból olvassa, és oda is írja vissza.

Szinte mindenki, aki komoly weblapot szeretne készíteni, lapját előbb vagy utóbb relációs adatbázishoz szeretné majd kapcsolni. Meglehet, a relációs adatbázis módszer immár harmincéves, de igen rugalmas, biztonságos és gyors. Az adatbázisok használata lehetővé teszi, hogy a webalkalmazásunkhoz szükséges adatokat anélkül tárolhassuk és kérhessük le, hogy saját megmaradó tárolóréteget (persistent storage layer) kellene létrehoznunk. Ennek eredménye kevesebb hiba, nagyobb gyorsaság és sokkal nagyobb biztonság.

A Zope nevű objektumközpontú alkalmazáskiszolgáló, amelyet az utóbbi néhány hónapban vizsgáltunk, egy ZODB nevű beépített objektum-adatbázissal rendelkezik. A ZODB egyszerre hatékony és könnyen használható; a Zope-ban minden, a DTML-dokumentumokat és könyvtárakat is ideértve, ZODB objektumokként tárolódik. A valóságban a ZODB ezeket az adatbázis-elképzeléseket tranzakciók formájában támogatja, ami azt jelenti, hogy fontos adatok tárolására is felhasználhatjuk, és eközben biztosak lehetünk benne, hogy adatainkat egy hosszú, összetett lekérdezés alatt sem fogja senki módosítani. Számos esetben azonban a tárolandó és lekérdezendő adatok kezelésére nem a ZODB a legjobb választás. Ennek sokszor egyszerűen az az oka, hogy az adat már létezik, és mi azt szeretnénk, hogy a Zope elérje. Az is megeshet, hogy állandó tárolóréteget akarunk készíteni, de azt szeretnénk, hogy az emberek a Zope-on kívülről is el tudják érni. Esetleg adataink egyszerűen jobban illeszkednek a relációs, mint az objektumközpontú adatbázismodellhez. Végül előfordulhat, hogy szervezetünk IT-részlege azt akarja, hogy minden adat relációs adatbázisban legyen tárolva.

E felsorolt okok és helyzetek miatt az alap-Zope-telepítés meghatározza a ZSQL tagfüggvény-objektumot. Ebben a hónapban a ZSQL tagfüggvényeit és a Zope általános relációsadatbázis-illesztési lehetőségeit fogjuk megvizsgálni. Mint látni fogjuk, egyszerű Zope-lapjainkat igen könnyű úgy átalakítani, hogy az adatokat relációs adatbázisból olvassa, és oda is írja vissza.

Adatbázis-kapcsolatok

Mielőtt elkezdhetnénk egy adatbázissal dolgozni, hozzá kell kapcsolódnunk. A Zope alatt ezt egy adatbázis-kapcsolati objektum létrehozásával tehetjük meg. A Zope-oldalak tetszőleges számú ilyen objektumot tartalmazhatnak, és ezek mindegyike képes SQL-lekérdezéseket küldeni az adatbázisnak.

A Zope alapesetben egyetlen típusú adatbázis-kapcsolatot ismer, amely az egyszerű ***Gadfly*** relációs adatbázissal való kapcsolatot teszi lehetővé. Csakhogy míg a Gadfly nagyon jól megfelel a Zope adatbázis-kapcsolatainak bemutatására, sebesség és képességek terén semmilyen más relációs adatbázissal nem versenyezhet. Azt javaslom, a Gadflyt egy az egyben hagyjuk is ki, és azt az adatbázis-csatolót telepítsük, amelyhez majd csatlakozni szeretnénk.

Mivel én az irodai adatbázis-kiszolgálón PostgreSQL-kiszolgálót futtatok, úgy döntöttem, hogy az Interneten jelenleg elérhető számos PostgreSQL-átalakító közül a `psycopg` adatbázis-átalakítót telepítem fel (a `psycopg`-ról további tájékozódási pontot a ***Kapcsolódó címek*** között találhatunk). Amikor ezt (vagy valamelyik másik) csomagot telepítjük, ne feledjük, hogy a Zope általában saját Python-változattal érkezik, ami a rendszerünkön esetleg fent lévő minden más másolattól független. Ez azt jelenti, hogy a `psycopg`-t a Zope-ban (a `$ZOPE/bin/python` által) megadott Python-könyvtárba kell telepítenünk, nem pedig a ***/usr/local/bin/python*** vagy a ***/usr/bin/python*** könyvtárak alá.

Mielőtt telepítenénk a `psycopg`-ot, telepítenünk kell az eGenix által írt és terjesztett `mxDateTime` osztályt. Ez a csomag lehetővé teszi, hogy a jelenlegi Unix-időkorlátokon (ami 1970-ben kezdődik és 2038-ban végződik) túli dátumokkal és időpontokkal is dolgozni tudjunk, illetve számos olyan kényelmes eljárást tesz elérhetővé, amelyek segítségével különféle dátum- és időformátumokkal dolgozhatunk. Ezt a modult akkor is fel kell telepítenünk, ha nem akarjuk használni, máskülönben a `psycopg` nem fog helyesen települni. A `mxDateTime` a ➲ http://www.egenix.com/files/python/eGenix-mx-Extensions.html címről tölthető le.

Figyeljünk rá, hogy nekünk a ***base*** (alap) csomagra van szükségünk (amely ingyenes), és nem az üzleti csomagra. Jobb, ha nem az `mxDateTime` RPM-változatát töltjük le, még akkor sem, ha RPM-alapú terjesztésünk van. Ennek az az oka, hogy a fordítást és a könyvtárak telepítését a Zope Python-fájában kell elvégeznünk, és nem a rendszer Python-fájában.

Az `mxBase` csomag letöltése és kicsomagolása után a telepítéshez az ***mxBase*** könyvtárba kell váltanunk, majd ki kell adnunk a következő parancsot:

```
$ZOPE/bin/python setup.py install
```

Az előbbi sor lefordítja az `mx`-modult, és Python-változatunkba telepíti.

A psycopg telepítése

Már közel járunk a `psycopg` telepítéséhez, amely Python- és C-kód keverékében íródott, és szüksége van a PostgreSQL fejlesztési (development) könyvtáraira. Ha a PostgreSQL-t RPM-ből telepítjük, szükségünk lesz az általunk használt PostgreSQL-változatnak megfelelő `postgresql-devel` RPM-csomagra is. A folyamat során az új fájlok általában a ***/usr/local/pgsql*** és a ***/usr/include/pgsql*** könyvtárakba kerülnek, bár néhány telepítés mindkét elérési útban `postgresql`-t használ `pgsql` helyett.

Most töltsük le a `psycopg` forráskódját a ➲ http://initd.org/pub/software/psycopg címről! Én a 1.0.4-es változatot töltöttem le, azonban úgy láttam, pár hetente kijön egy-egy új változat, ezért ha lehetséges, a legfrissebbet töltsük le. A `psycopg`

kicsomagolásához és telepítéséhez el kell készítenünk a `makesetup` parancsfájlt (ez jelenleg a legfrissebb Zope 2.5b1-ben a `$ZOPE/lib/python2.1/config` alá települ):

```
chmod 775 $ZOPE/lib/python2.1/config
```

A `psycopg` beállításához váltsunk a forráskönyvtárába, és gépeljük be a következőket:

```
./configure
  --with-python=$ZOPE/bin/python
  --with-zope=$ZOPE
  --with-mxdatetime-includes=$ZOPE/lib/python2.1/
    ➥site-packages/mx/DateTime/mxDateTime
  --with-postgres-includes=/usr/include/pgsql
```

Természetesen az elérési utakat a saját telepítésünknek megfelelően meg kell változtatnunk, különös figyelmet szentelve a Python-változatszámoknak (jelen esetben ez 2.1) és a PostgreSQL ***include*** könyvtárának.
Bár továbbra is meg vagyok győződve róla, hogy valamilyen `configure` kapcsolóval is meg lehetne oldani a dolgot, egyelőre úgy tűnik, a `Makefile`-t kézzel kell átszerkesztenünk, hogy az új header könyvtárat a `CFLAGS` változóhoz hozzáadhassuk. Kedvenc szövegszerkesztőnkkel nyissuk meg a `Makefile`-t, és a `CFLAGS` meghatározáshoz (az én változatomban a 90. sor) adjuk hozzá `$ZOPE/include/python2.1`-ben található headereket. Ha tehát a `$ZOPE` nálunk */**usr**/**local**/**zope***, a `CFLAGS`-hoz a következőket kell adnunk:

```
-I/usr/local/zope/include/python2.1
```

Mentsük a `Makefile`-t, majd telepítsük a `psycopg`-ot:

```
make && make install && make install-zope
```

A fentiek lefordítják a `psycopg`-ot, majd ***$ZOPE*** könyvtárunkba telepítik.
Végül a `psycopg` megosztott programkönyvtárát (***psycopgmodule.so***) a ***$ZOPE**/**lib**/**python2.1**/**site-packages*** könyvtárból helyezzük át a ***$ZOPE**/**lib**/**python2.1**/**lib-dynload***/-ba.

A psycopg beállítása

A Zope újraindítása után a `psycopg`-ot mindjárt ki is próbálhatjuk – egy új termék saját könyvtárba való telepítésével (sajnos a Zope újraindítása az egyetlen lehetőségünk, hogy a rendszerrel egy új termék telepítésének megtörténtét közöljük). A telepítendő termék neve ***Z Psycopg Database Connection***, és a Zope kezelőfelület jobb felső sarkában az ***Add product*** menüben találjuk.
Minden adatbáziskapcsolat-objektum egy távoli gép egyetlen adatbázishoz enged hozzáférést, egyetlen felhasználónévvel és jelszóval. Ez azt jelenti, hogy ha az adatot két különböző adatbázis között osztottuk meg (avagy különböző adatbázis-kiszolgálók között), akkor két kapcsolatobjektumra lesz szükségünk.
Amikor a ***Z Psycopg Database Connection***-t kiválasztjuk az ***Add product*** menüből, a rendszer feltesz néhány, adatbázis-kapcsolatunkra vonatkozó alapvető kérdést. Az ID-t (amelynek minden könyvtárban egyedinek kell lennie) és a címet (ez a kezelőfelületen jelenik majd meg), valamint az adatbázis-kapcsolat karaktersorozatát be kell írnunk. Ez a kapcsolatkaraktersorozat adja meg a Zope-nak, hogyan tud kapcsolatba lépni a PostgreSQL-kiszolgálóval. Irodámban az atf adatbázis a `databases`-en található PostgreSQL-kiszolgálón helyezkedik el, ahová `reuven` néven, jelszó nélkül kapcsolódhatok. Ennek megfelelően a következő kapcsolódási karaktersorozatot használom:

```
host=databases dbname=atf user=reuven
```

Ha kívánjuk, a maradék részeket meghagyhatjuk az alapértelmezett értéken. Kattintsunk az ***Add*** gombra, ezáltal visszatérünk abba a könyvtárba, ahová az új kapcsolatobjektumunkat illesztettük be.
A kapcsolatobjektumra kattintva néhány Zope-tábla jelenik meg, segítségükkel elvégezhetjük az adatbázis karbantartását. Ezek közül a négy legérdekesebb:

- ***Status***: ebből a táblából megtudhatjuk, hogy az adatbázis-kapcsolat nyitott-e (vagyis éppen kapcsolódik-e a PostgreSQL-kiszolgálóhoz vagy sem). Itt – ha szükséges – lehetőségünk nyílik ezt a kapcsolatot lezárni.
- ***Properties***: megváltoztathatjuk azokat a beállításokat, amelyeket az adatbáziskapcsolat-objektum eredeti létrehozásakor írtunk be. Különösen hasznos lehet ez a rész, ha az adatbázist másik kiszolgálóra visszük át, vagy megváltoztatjuk az eléréshez szükséges jelszót.
- ***Test***: az adatbázis-kapcsolatot próbálhatjuk ki azáltal, hogy tetszőleges SQL-lekérdezést küldünk ki rá. Természetesen a lekérdezésnek érvényesnek kell lennie; ha hibás SQL-lekérdezést küldünk ki, vagy olyan táblát címzünk meg, amely nem létezik, a PostgreSQL-kiszolgáló által visszaadott megfelelő hibajelzést kapjuk vissza. Például begépelhetjük a `SELECT * FROM pg_database;` parancsot. Bármilyen SQL-utasítást bevihetünk ezen a dobozon keresztül, ami jól jöhet az adatbázis kipróbálásakor, főleg, ha nem rendelkezünk közvetlen Telnet- vagy SSH-kapcsolattal. Amikor `INSERT` vagy `UPDATE` lekérdezést gépelünk be, a Zope jelzi, hogy a lekérdezés nem ad vissza eredményt. Mint általában, most sem érdemes `SELECT *` formájú lekérdezéseket írni, legfeljebb egyértelmű esetben, nehogy végül meglepődjünk az eredményoszlopok sorrendje vagy neve miatt.
- ***Browse***: a böngésző (browse) táblán a PostgreSQL adatbázistábláit nézegethetjük végig, amely Zope-fastílusban listázza a táblákat és azok mezőit.

ZSQL tagfüggvények

Most, hogy az adatbázis kapcsolat már él, egy vagy több ZSQL tagfüggvényt is készíthetünk. Minden ZSQL tagfüggvény egyetlen SQL-lekérdezés (ha kívánjuk, változó számú értékkel), amely az adott kapcsolattal dolgozik.
Készítsünk egy ZSQL tagfüggvényt, amellyel új nevet adhatunk a telefonkönyvhöz. Természetesen ehhez az adatbázisban előbb meg kell határoznunk a megfelelő táblát. A táblát könnyen elkészíthetjük, ha az *1. lista* tartalmát elküldjük a PostgreSQL-nek – akár az adatbázis kapcsolatunk próbatábláján, akár a hagyományos `psql` parancssoros felületen.
Ha olyankor akarunk ZSQL tagfüggvényeket beilleszteni, amikor nincs elérhető adatbázis-kapcsolat, a Zope hibaüzenetet fog küldeni, jelezvén, hogy egyetlen megfelelő adatbázis-kapcsolatot sem talált.
A Zope beépített „szerzeményezés” rendszerének megfelelően a ZSQL tagfüggvényei bármely adatbázis-kapcsolatot felhasználhatják, amely a Zope rendszerében felettük áll. A felhasz-

1. lista A tábla létrehozása

```
CREATE TABLE AddressBook (
  person_id         SERIAL,
  first_name        TEXT    NOT NULL,
  last_name         TEXT    NOT NULL,
  address1          TEXT    NOT NULL,
  address2          TEXT    NULL,
  ➥- Nem mindenkinek kell a mÆsodik sor
  city              TEXT    NOT NULL,
  state_province    TEXT    NULL,
  ➥- Nem mindenki Øl az USA-ban
  postal_code       TEXT    NULL,
  ➥- Nem minden orszÆgban van
  phone_number      TEXT    NOT NULL,
  ➥- Nem mindenki Øl az USA-ban
  fax_number        TEXT    NULL,
  ➥- Nem mindenkinek van faxgØpe
  cell_number       TEXT    NULL,
  ➥- Nem mindenkinek van mobiltelefonja
  PRIMARY KEY(person_id)
);
```

náló így minden tagfüggvényhez különféle adatbázisokat rendelhet – ez a különböző adatbázisokban fellelhető adat egyetlen alkalmazásba egyesítését, vagy a teljes weblap átültetését egy másik adatbázistípusra teszi lehetővé.
A ZSQL tagfüggvények létrehozásához váltsunk abba könyvtárba, ahol adatbázis-kapcsolatunkat létrehoztuk, és válasszuk az ***Add product*** menü ***ZSQL method*** pontját. Ismét meg kell adnunk pár adatot: az ID-t (amely az objektum könyvtárán belüli egyedi azonosítását szolgálja), a címet (ez a kezelőfelületen fog látszani), a kapcsolókat (melyeket majd a következő részben tárgyaljuk), végül magát az SQL-t. Az SQL-lekérdezés egyszerű vagy tetszés szerinti bonyolultságú lehet, és `INSERT`, `UPDATE` vagy `DELETE` utasítást is végrehajthat.
Miután ZSQL tagfüggvényünket a rendszerhez adtuk, rákattintva számos Zope-tábla kerül elő. Az egyik ezek közül `test` címkével rendelkezik és – miként azt sejteni lehet – a lekérdezés kipróbálására szolgál. Ha lekérdezésünk értékeket is vár, egy HTML-űrlapon megadhatjuk őket. Ha nem, akkor egyszerűen csak a ***Submit Query*** gombra kell böknünk. Ezután, akárcsak adatbáziskapcsolat-objektumunk próbatábláján, egy HTML-formátumú táblát kapunk vissza, amely lekérdezésünk eredményeit hivatott megjeleníteni, vagy jelzi, hogy lekérdezésünk nem adott vissza eredményt.
Minden egyes kiadandó lekérdezéshez elkészíthetjük a megfelelő ZSQL tagfüggvényt. Bár ez elsőre kicsit nehézkesnek tűnhet, valójában igen rugalmas és elegáns megoldás, amelyet egyre inkább megtanulok értékelni. Ha körülbelül húsz lekérdezés kiadására számítok, egy webalkalmazásban mindegyiket külön ZSQL tagfüggvénybe helyezem, majd a DTML-lapokról ezeket a tagfüggvényeket hívom meg.
A DTML-lapokon a ZSQL tagfüggvények eredményét `<dtml-in>` tag formában kaphatjuk meg. Ha például a következő lekérdezést megvalósító ZSQL tagfüggvényt szeretném használni:

```
SELECT first_name, last_name, phone_number,
       fax_number, cell_number
  FROM AddressBook
 ORDER BY last_name, first_name
```

Ha ennek a ZSQL-nek a ***names-and-phone-numbers*** nevet adom, a 2. listában (36. CD Magazin/Zope könyvtár) látható kóddal hívhatom meg egy DTML-dokumentumból. Mindössze néhány soros DTML-kóddal képesek voltunk létrehozni egy egyszerű (de hasznos és rugalmas) ZSQL tagfüggvényt. Nézzük, hogyan is működik!
Amikor a Zope ehhez a DTML-dokumentumhoz tartozó kérelmet kap, értelmezi a DTML-t és a benne található összes tagot végrehajtja. A `<dtml-in>` ciklusszerkezet egy adatsort vár bemenetként – jelen esetben ez az adatsor a ***names-and-phone-numbers*** tagfüggvény meghívásából kapott eredmény. A `<dtml-in>` tag a visszaadott halmaz minden egyes oszlopához egyúttal egy-egy változót is rendel. Ez az oka annak, hogy a `<dtml-var first_name>` tagot később a felhasználó első nevének kiírásához fel tudjuk használni; a Zope a `first_name` oszlop értékét önműködően a `first_name` nevű változóhoz rendeli.
Hogy elkerülhessük a felesleges és az üres sorok kiírását, a `<dtml-if>` tagot használjuk annak eldöntésére, hogy a PostgreSQL értékes, vagy NULL, azaz üres értéket adott-e vissza.

A ZSQL értékei

Az már nyilvánvaló, hogyan használhatjuk a ZSQL tagfüggvényeket és a DTML-t, ha mindig ugyanazt a lekérdezést akarjuk kiadni. Ha azonban az alap-lekérdezésünket minden egyes futás során módosítani szeretnénk, egy vagy több értéket is meg kell adnunk.
Ha valakiről vezetékneve (vagy vezetéknevének egy része alapján) szeretnénk adatott lekérni, az SQL szabványos kifejezéseit kihasználva a következő példához hasonló ZSQL tagfüggvényt kellene megadnunk:

```
SELECT first_name, last_name, phone_number,
       fax_number, cell_number
  FROM AddressBook
 WHERE last_name LIKE XXXXXX
ORDER BY last_name, first_name
```

A DTML-ben az XXXXXX helyére a `<dtml-sqlvar>` tag kerül, amely a helyes idézőjelezést is önműködően elvégzi helyettünk. A felhasznált SQL-változót és annak típusát is meg kell adnunk:

```
SELECT first_name, last_name, phone_number,
       fax_number, cell_number
  FROM AddressBook
  WHERE last_name LIKE <dtml-sqlvar
  name_sqlregexp type="string">
ORDER BY last_name, first_name
```

Ahhoz, hogy a fenti ZSQL-példa helyesen működjön, a tagfüggvény létrehozásakor az értéket (`name_sqlregexp`) a megfelelő szövegmezőben meg kell neveznünk. A Zope ennek a változónak veszi majd az értékét, lekérdezésünkbe helyezi, majd az eredményeket lekéri.
Még többet is kihozhatunk a Zope-ból, ha a `<dtml-sqltest>` tagot használjuk, ami a `<dtml-sqlvar>` tagjához hasonló módon működik:

```
SELECT first_name, last_name, phone_number,
       fax_number, cell_number
```

```
    FROM AddressBook
   WHERE <dtml-sqltest name_sqlregexp
         op="like" type="string">
ORDER BY last_name, first_name
```

Ha a fenti lekérdezést a `select_by_last_name` nevű ZSQL tagfüggvényben tároljuk, akkor a Zope önműködően el tudja készíteni a DTML-dokumentum vázát, azaz egy olyan dokumentumot, amelyben a felhasználók keresési feltételeket adhatnak meg, és az eredményt is megnézhetik. Ehhez egyszerűen csak ki kell választani a ***Z Search Interface*** terméket az ***Add product*** listából. Itt a rendszeren található összes kereshető objektumból választani tudunk, ideértve az éppen most elkészült ZSQL tagfüggvényt (`select_by_last_name`). Válasszuk ki, és adjunk neki egy ID-t (én a `search_by_last_name` nevet használtam). Névként az `input ID`-t adtam, ami elég beszédes egy olyan HTML-űrlap esetében, amely bemenetként szolgál a `search_by_last_name`-hez (ezt pedig `search_by_last_name_form`-nak neveztem el). A Zope korszerű változataiban azt is megadhatjuk, hogy a rendszer DTML tagfüggvényeket vagy lapsablonokat (page template) készítsen – mi most az előbbit szeretnénk.
Az ***Add*** gombra kattintva a pillanatnyi könyvtárban két új DTML tagfüggvény jön létre a korábban megadott neveknek megfelelően. Az `input ID` címre kattintva egy egyszerű HTML-űrlapot jelenít meg, melybe SQL-szabványos kifejezést gépelhetünk. A ***Submit*** gombra mutatva a lekérdezés elküldődik a `search_by_last_name` DTML tagfüggvényhez, amely viszont a mi ZSQL tagfüggvényünket hívja meg (`select_by_last_name`), így a lekérdezés végül a PostgreSQL-hez kerül. A PostgreSQL az eredményeket visszaadja a `select_by_last_name`-nek, amely az eredményhalmazt a `search_by_last_name`-hez továbbítja, ami ezután böngészőnkön megjeleníti őket.
Természetesen a létrehozott DTML tagfüggvényeket módosíthatjuk is, hogy jobban tükrözzék a lapunk stílusát. A Zope által önműködően készített DTML-lapokat át is másolhatjuk, példaként használva őket saját adatbázis-lekérdezéseinkhez.

Beszúrás

Az egyetlen nagyobb feladat, ami még hátravan, az `INPUT` lekérdezés megvalósítása, amely elemeket ad az adatbázishoz. Szerencsére a megoldás meglehetősen egyszerű: készítünk egy ZSQL tagfüggvényt, amely sorokat szúr az adatbázisba. Ezután egy DTML-dokumentumot kell megalkotnunk, amely a HTML-űrlap elemeit egy másik DTML-dokumentumnak adja át. Ez a második dokumentum meghívja a ZSQL tagfüggvényünkhöz tartozó `<dtml-call>`-t. Voilà, rekordunk máris bekerült az adatbázisba.
A 3. lista (36. CD Magazin/Zope könyvtár) mutatja be a létrehozandó ZSQL tagfüggvényt, amelyeket `insert_address_data` névre kereszteltem. Ezután egy egyszerű DTML-dokumentumot hozunk létre, amely egyetlen HTML-űrlapot tartalmaz (lásd a 4. listát, 36. CD Magazin/Zope könyvtár).
Végül elkészítjük az `insert_address` DTML-dokumentumot, amely az `insert_address_form` adatait fogadja, és az adatokat a `insert_address_data` ZSQL tagfüggvényhez továbbítja:

```
<dtml-var standard_html_header>
<h2><dtml-var title_or_id></h2>

<dtml-try>
    <dtml-call insert_address_data>
<dtml-except>
    <p>Sorry, but the INSERT didn·t work.</p>
<dtml-else>
    <p>Successfully inserted!
</dtml-try>

<dtml-var standard_html_footer>
```

A felhasználók ettől kezdve az `insert_address_form` által megadott HTML-űrlap felhasználásával képesek lesznek adatokat bevinni a PostgreSQL-táblába, és a `search_by_last_name_form` segítségével lehetőségük nyílik lekérdezni az adatokat. Elég lenyűgöző, hogy ilyen kevés fájllal ilyen sokat meg tudtunk csinálni, ráadásul ahhoz, hogy mindez működjön, még szövegszerkesztőre sem volt szükségük, mindössze a webböngészőnket kellett használnunk.

Összegzés

Bár nem tökéletesek, a ZSQL tagfüggvényeket igen elegáns módszernek tartom a HTML-lapok és a mögöttük elhelyezkedő adatbázis összekapcsolására. A ZSQL egy újabb példája annak, hogy a Zope milyen rugalmas, finom megközelítése a webfejlesztésnek, bár az is igaz, okoz némi fejtörést, mire mindez tisztává és világossá válik. Ha valaki már eleve ismeri a DTML-t és az SQL-t, annak nem fog nehézséget okozni Zope-alkalmazásaiba ZSQL tagfüggvény segítségével adatbázisokat beépíteni, sőt, az is megoldható, hogy a munkát egy honlapon belül megosszuk az SQL-t ismerő (és ZSQL tagfüggvényekkel dolgozó), illetve az őket meghívó DTML-tagfüggvényekkel foglalkozó munkatársak között.

Linux Journal 2002. május, 97. szám

Reuven M. Lerner (reuven@lerner.co.il) kisebb, webes és internetes módszerekkel foglalkozó tanácsadó cég tulajdonosa és vezetője. A cikk megjelenésének időpontjában valószínűleg már végleg elkészült Core Perl című könyvével, melyet idén jelentet meg a Prentice-Hall. Az ATF honlapon érhető el (➲ http://www.lerner.co.il/atf/).

Kapcsolódó címek

A Zope honlapja ➲ http://www.zope.org
Nagyszámú leírást rejt, többek között általános adatbázis- és ZSQL-felhasználási útmutatót. A Python nyelv, melyben a Zope nagy része íródott, a ➲ http://www.python.org címen érhető el.

A Pythonhoz és Zope-hoz írt psycopg-csatoló a ➲ http://initd.org címen érhető el. Az írásunkban olvasható telepítési utasítások nagyrészt a ➲ http://initd.org/pub/software/psycopg/FAQ megjegyzésein alapulnak. A PostgreSQL honlapja a ➲ http://www.postgresql.org címen található.