Modellezés DODS segítségével

Az Enhydra-kiszolgáló részeként a DODS az objektumok és a relációs adatbázisok között próbál kapcsolatot teremteni.

Mivel az adatok tárolását, lekérdezését egyszerűvé, rugalmassá és biztonságossá teszik, a legtöbb komoly webalkalmazás gerincét a relációs adatbázisok alkotják. Ez a felállás többnyire egészen addig tökéletesen működik, amíg a fejlesztők olyan objektumokkal nem kezdenek el dolgozni, amelyek teljesen más szemléletmódot követelnek. Vajon lehetőség van-e arra, hogy áthidaljuk az objektumközpontú és relációs világok közti szakadékot?

Valójában számos olyan módszer létezik, amellyel a relációs modellt objektumokká és eljárásokká formálhatjuk, és a legtöbb programozó már rég tervezett magának egy ilyen rendszert. Ahogyan a múlt hónapban láthattuk, a Perl-programozók egy kis segítséget kaphatnak az Alzabo modultól – lehetőséget ad nekik, hogy megtervezzék a táblákat, elérésükhöz pedig eljárásalapú csatolófelületet nyújt.

Ebben a hónapban a DODS-ra (Data Object Design Studio, azaz adatobjektum tervezőstúdió) vetünk egy pillantást, amely szellemében az Alzabóra hasonlít, ezt azonban Java-felhasználóknak szánták. A DODS az Enhydra központi eleme, amelynek hamarosan megjelenő változata (Enhydra Enterprise) várhatóan az első olyan nyílt forrású alkalmazáskiszolgáló lesz, amely támogatja a J2EE-t (Java 2, Enterprise Edition).

Jelenleg az Enhydra Enterprise még kipróbálás alatt áll, és bár úgy tűnik, a DODS-támogatás sokat fejlődött az utolsó változat óta, ***David Young*** a Lutris Enhydra-prófétája szerint az Enhydra 3.x DODS-változata azért üzembiztosabb. Hogy a dolgokat könnyen kipróbálhassam, a Lutris küldött nekem egy EAS (Enhydra Applicattion Service) példányt, amely az Enhydra bővített, kereskedelmi változata.

Nem vagyok benne teljesen biztos, mi a különbség az EAS és a nyílt forrású Enhydra-kiszolgáló között. Az ***Enhydra.org*** azt írja, hogy az EAS az Enhydrán alapul, de az EAS és egy Enhydra-példány vásárlása közötti különbség nem teljesen nyilvánvaló. Azt fogom feltételezni, hogy az általam telepített EAS nagyjából azonos a 3.1-változattal, bár meglehet, hogy ez nem teljesen pontos feltételezés.

A DODS áttekintése

A DODS-nak, akárcsak a múlt hónapban megismert Alzabo-rendszernek, kettős célja van: magas szintű felületet nyújt adatbázisok tervezéséhez, illetve eljárás- és objektumkészletet nyújt, amellyel azután az adatbázis elérhető. Míg az Alzabo kiszolgálóoldali, a DODS ügyféloldali Javában íródott alkalmazás, amelylyel adatbázisainkat építhetjük fel, illetve szerkeszthetjük.

A DODS elsődleges célja, hogy párhuzamosan készítsen SQL-meghatározásokat és Java-osztályokat, amelyek ugyanazt az adatbázist írják le. Ezután adatbázisba tölthetjük az SQL-meghatározásokat, a Java-osztályokkal pedig elérhetjük őket. Ezenkívül a DODS többfajta adatbázissal való munkára is fel lett készítve. Jelenleg PostgreSQL-, MySQL-, Sybase- és Oracle-rendszerekkel működik, de ez a kör a jövőben valószínűleg tovább bővül. Mivel a tényleges SQL-lekérdezések egy objektum-középrétegben íródtak, az Enhydra-programok átírás nélkül vihetők át egyik adatbázis-kezelőről a másikra. A valóságban természetesen kicsit bonyolultabb a dolog. Például az Enhydra PostgreSQL-támogatása nem éppen lenyűgöző, ugyanis figyelmen kívül hagyja a `SERIAL` adattípust (ez valójában egy számláló, más néven szekvencia), és nem kezeli a hivatkozásiépség-megkötéseket (referential integrity constrations), így például az idegen kulcsokat sem. Mindenesetre a cél becsülendő, és örömmel látnám, ha az Enhydra 4.x már kezelné ezt a gondot. Idővel a DODS várhatóan egyre több különféle adatbázist lesz képes kezelni, illetve a megfelelő lekérdezést az egyes SQL-nyelvjárásokhoz elkészíteni.

Az XMLC-dokumentum elkészítése

Hogy a működésükkel megismerkedhessünk, készítsünk az Enhydrával és a DODS-sal egyszerű adatbázis-, illetve webalkalmazás-együttest. Példáimban a PostgreSQL-t fogom használni, két okból kifolyólag is. Egyrészt, mert kitűnő nyílt forrású adatbázis-kezelő, másrészt mert DODS támogatja. Példánk, akárcsak a múlt hónapban, egy egyszerű webnapló lesz (más néven blog, egy olyan napló, amely az adatbázis bejegyzéseit fordított időrendben mutatja be). Egy ilyen program megírása nem különösképpen bonyolult, viszont annál inkább vonzó a DODS és az Enhydra kipróbálásaként.

Az első megállónk az Enhydra Appwizard lesz, amely elkészíti az alkalmazásunkhoz szükséges vázlatokat és könyvtárakat. Az Appwizard a ***$ENHYDRA/bin*** könyvtárban található, ahol az `ENHYDRA` az Enhydra telepítéskönyvtárának megfelelő környezeti változó. (Amikor RPM-csomagokból a saját RedHat-gépemhez telepítettem CD-ről, az `ENHYDRA` értéke ***/usr/local/lutrys-enhydra3.5.2*** volt.)

Az első `appwizard` képernyőn a hagyományos webalkalmazás és az Enhydra szuperservlet között választhattam, az utóbbi mellett döntöttem. A következő képen a HTML projektre voksoltam (a vezeték nélküli WML projekt helyett), majd a projektet elneveztem „blog”-nak és behelyeztem az ***il.co.lerner*** csomagosztályba. Elfogadtam az Enhydra-alkalmazásokhoz rendelt, alapértelmezett ***~/enhydraApps/*** alkalmazás saját könyvtárát. A forráskódomhoz nem szándékoztam szerzői jogi üzenetet rendelni, így végezetül a ***Finish***-re kattintottam, amely a ***~/enhydraApps/*** könyvtárban 18 új állományt hozott létre.

Most, hogy elkészítettük az alkalmazás vázát, módosíthatjuk az Enhydrával érkező alapértelmezett üdvözlő (Welcome) oldalt. Ezt két lépésben kell megtennünk: először a ***Welcome.html*** HTML-fájlt kell megváltoztatnunk, amely az én gépemen a ***~/enhydraApps/blog/src/il/co/lerner/presentation/Welcome.html*** helyen található.

Érdemes odafigyelni rá, mivel ez a fájl nem csak alap-HTML, hanem az XMLC által feldolgozandó további tagokat is tartalmaz (lásd ***Linuxvilág*** 2001. október, 67. oldal). Amint az *1. listában* látható, úgy fogjuk megváltoztatni, hogy az eredeti egyszerű oldal helyett blogunk legfrissebb adatait jelenítse

meg. Az XMLC és a hagyományos HTML-oldal közötti egyetlen különbség az, hogy a módosítani kívánt részeket `id` tulajdonsággal felruházott `<span>` tagok közé helyezzük. Például

```
<p><b><span id="date">Date</span></b></p>
<p><span id="text">Text</span></p>
```

amennyiben ezt a fájlt most egyszerűen csak megjelenítjük a böngészőben, a ***Date*** és ***Text*** szavakat fogjuk látni. A felhasználók azonban nem fogják ezt az oldalt közvetlenül elérni. Az XMLC ugyanis Java-osztállyá fogja őket fordítani. Ezután a ***WelcomePresentation*** osztályt használhatjuk a dokumentum egy példányának előállítására, miközben a ***text*** és a ***date*** mezők értékét önműködően létrehozott eljárásokkal állíthatjuk be.

A DODS használata

A ***WelcomePresentation*** a ***date*** és ***text*** mezőkbe szánt adatot a relációs adatbázistáblából kérdezi le. Mielőtt folytatnánk, készítenünk kell tehát egy táblát, és be kell népesítenünk néhány adattal.
Ez az a pont, ahol a DODS belép a képbe. A ***$ENHYDRA/bin/dods*** helyen található `dods` program szintén ügyféloldali, grafikus, Javában írt alkalmazás. Amennyiben a DODS-sal dolgozunk, soha ne feledjük, hogy olyan alkalmazást használunk, amely két különböző szemléletmódot köt össze, így a kifejezésmódja néha bizony furcsának tűnhet.
A DODS egy csomag készítésével indul, amely majd az általunk készített összes tábla és tulajdonság tárolására szolgál. Amint azt a kezdeti DODS-képernyőn is láthatjuk, e csomag neve alapértelmezés szerint ***root***. Ezt én ***blog***-ra változtattam, rákattintottam a ***root*** mappára, majd az ***Edit*** menüből a ***Package*** pontot választottam.
Az adattáblát (***BlogEntries***) két tulajdonsággal hozzuk létre: ***date*** (***dátum***), és ***text***, amelyek egyébként megegyeznek a ***Welcome.html*** változatunkban használt ***id*** tagokkal. Első lépésként az ***Insert*** menüponttal egy új táblát adunk hozzá a BlogEntrieshez: kiválasztjuk a ***data object*** pontot és a ***BlogEntries*** nevet adjuk neki.
Ezután a táblánkhoz két mezőt (***date*** és ***text***) kell adnunk. Ehhez a DODS-ablak bal felső sarkában kattintsunk a ***BlogEntries*** szóra és az ***Insert*** menüpont használatával szúrjuk be a két új tulajdonságot. Mindkét tulajdonságunk ***varchar*** típusú lesz – jelölvén, hogy szöveget szeretnénk bennük tárolni. Bár a mi esetünkben ez tökéletesen megfelel, kár, hogy DODS a PostgreSQL `TIMESTAMP` adattípusát nem kezeli, ez ugyanis ügyes és kifinomult idő- és dátumadat-kezelést tesz lehetővé. Így a dátumokat szöveges formában fogjuk tárolni, tudván, hogy az `ORDER BY` segítségével sorrendben kaphatjuk őket vissza – és nekünk ennyi elég is.
Mivel web-, illetve adatbázis-alkalmazásunkkal a lehető legnagyobb sebességet szeretnénk elérni, és mivel lényegesen kevesebbszer fogunk szöveget beszúrni, mint lekérdezni, közöljük a DODS-sal, hogy mindkét mezőnket tegye indexelhetővé és lekérdezhetővé. Az első az SQL-meghatározást fogja módosítani azáltal, hogy indexet készít a mezőkhöz. A második egy további eljárást készít, amivel az oszlopban tárolt adatokhoz férhetünk hozzá.
Végül az adatbázismenüből kiválasztjuk a PostgreSQL-t. Ezáltal a DODS kifejezetten PostgreSQL-stílusú SQL-t készít.
Most, hogy táblánkat elkészítettük a DODS-sal, nincs más hátra, minthogy (a ***File, Save as*** menüpont segítségével XML-formátumú DOML formában) mentsük, amely leírja a táblánkat, és mind a Java, mind az SQL elkészítéséhez felhasználható. Mentsük a DOML-fájlt a projektcsomag forráskönyvtárába; az én esetemben ez a ***blog/src/il/co/lerner*** könyvtárat jelenti. Ha DOML-fájlunk elkészült (lásd a *2. listát*), SQL- és Java-parancsokká alakíthatjuk át a ***File*** menü ***Build all*** parancsával. E választás eredményeképpen számos fájl keletkezik az adatkönyvtárban, ezért amikor válaszolnunk kell, hogy a fájlokat hova szeretnénk telepíteni, válasszuk azt az adatkönyvtárat, ahová a ***blog.doml*** fájlt helyeztük. Egy ablak fog megjelenni, amely tájékoztat bennünket, hogy a DODS mit is csinál éppen. Ha minden simán ment, a DODS-ból akár ki is léphetünk.

Az adatbázis elkészítése

A DODS `build all` parancsának futtatása a DOML-fájlt jó néhány új fájlra bontotta szét az adatkönyvtárban. A könyvtár immár nemcsak egy (korábban üres) Makefile-t tartalmaz, hanem egy `blog` alkönyvtárat is, amely a következő négy Javaosztályt tartalmazza: `BlogEntriesBO`, amely az Enhydra megjelenítést végző bemutató objektumához hasonlít; `BlogEntriesDataStruct`, amely tulajdonképpen táblánk adatait tárolja; `BlogEntriesDOI` amely a `BlogEntriesDO` objektumhoz tartozó csatolófelület; végül a `BlogEntriesQuery`, amely lehetővé teszi, hogy az előzőleg lekérdezhetőnek megjelölt mezőket lekérdezzük.
Az elkészült Java-forráskódon túl néhány olyan fájlt is találunk itt, amely SQL-utasításokat tartalmaz. Nevezetesen rálelhetünk egy ***create_tables.sql*** fájlra, amelynek segítségével az adatbázisunkat hozhatjuk létre.
Mindehhez a `CREATEDB` parancsot fogjuk használni, amelyet általában valamilyen erre jogosult felhasználó hajt végre Unix-héjprogramból (ami nem feltétlenül a rendszergazdát jelenti; nézzük végig a PostgreSQL leírását, hogy megtudjuk, miképpen készítsünk PostgreSQL-felügyelőket).
Kiadhatjuk a következő parancsot:

```
CREATEDB blog
```

Ezekután az adatbázisnak interaktív módon küldhetünk lekérdezéseket a következő paranccsal:

```
psql blog
```

Ha tábláinkat az önműködően létrehozott DODS parancsfájl segítségével szeretnénk létrehozni, a `psql \i` parancsát kell használnunk:

```
\i
➥/home/reuven/enhydraApps/blog/src/il/co/
➥lerner/data/create_tables.sql
```

Láthatunk néhány `CREATE` üzenetet, majd végül ismét a psql parancssora jelentkezik be. A `\d` parancs használatával kideríthetjük, hogy a DODS nem készítette el a ***BlogEntries*** táblát. Ehelyett két másik táblát hozott létre, az egyiket ***objectid***, a másik (elsődleges) táblát pedig ***newdbtable*** néven. Az ***objectid*** tábla a `PostgreSQL` sequence függvényeit hivatott helyettesíteni, jól szemléltetve az ilyesfajta általános eszközök korlátjait. A tábla egy ***next*** oszloppal rendelkezik, amely megmutatja, milyen azonosító lesz a következő. Ennek megfelelően az adatokat a ***newdbtable*** táblába szúrjuk be, ilyenkor az ***objectid*** táblát mindig egy-egy sorral bővítjük.
Adjunk is mindjárt néhány elemet a táblákhoz, hogy legyen mit lekérdezni:

1. lista A Welcome.html XMLC-bemeneti fájl, amit a Weblog megjelenítéséhez fogunk használni

```
<html>
  <head>
      <title>Weblog</title>
  </head>

  <body bgcolor="#FFFFFF">

      <H1>Weblog</H1>

      <P>Welcome to our Weblog!  Here is the
            ➥latest entry:</P>

      <p><b>
    <span id="date">Date</span>
    </b></p><p><span id="text">Text</span></p>

  </body>
</html>
```

2. lista A blog.doml, a DODS által önműködően készített fájl

```
<?xml version="1.0" encoding="UTF-8"?>
<doml>
  <database database="PostgreSQL"
   ➥legal_values="Standard,InstantDB,Oracle,
   ➥Informix,Msql,Sybase, PostgreSQL">
    <package id="blog">
      <table id="blog.BlogEntries"
       ➥dbTableName="NewDBTable">
        <column id="entrydate"
              ➥isIndex="true"
         ➥usedForQuery="true">
          <type dbType="VARCHAR"
              ➥javaType="String"/>
        </column>
        <column id="text" isIndex="true"
         ➥usedForQuery="true">
          <type dbType="VARCHAR"
              ➥javaType="String"/>
        </column>
      </table>
    </package>
  </database>
</doml>
```

```
INSERT INTO newdbtable (entrydate, text,
➥objectid, objectversion)
➥VALUES (NOW(),'First blog entry', 1, 1);
INSERT INTO objectid ( next ) VALUES ('2');

INSERT INTO newdbtable (entrydate, text,
➥objectid, objectversion)
➥VALUES (NOW(),'Second blog entry', 2, 1);
INSERT INTO objectid ( next ) VALUES ('3');
```

Most, hogy már van két blogbejegyzésünk, ki is léphetünk a psql-ből (\q), és elkezdhetjük módosítani a Java-osztályokat. Rakjuk össze mindezt!
A varázslás nagy része a ***WelcomePresentation.java*** fájlban zajlik. Itt fog elkészülni a ***Welcome.html*** és az adatbázisobjektumok egy-egy példánya, majd a lekérdezések eredményének megszerzése után a HTML-fájl itt töltődik fel az adatokkal. Miután a 3. listának (24. CD Magazin/DODS könyvtár) megfelelően módosítottuk a ***WelcomePresentation.java*** fájlt, futtassuk le a `make`-et a projekt sajátkönyvtárából. Az Enhydra lefordítja a Java-osztályainkat, ellenőrzi, hogy minden szükséges a helyén van-e, és futásra kész állapotba hozza az alkalmazásunkat.
Figyeljük meg, hogy a 3. listában módosítanunk kellett a `run` eljárást, hogy két új kivételt adjon vissza: a `NonUniqueQueryException`-t és `DataObjectException`-t. Ezeket az általunk létrehozott különféle adatobjektumok hozzák létre, és mivel ezeket a kivételeket nem szándékozunk elfogni, a hívónak jeleznünk kell, hogy lehet, hogy kivételt vált ki.
A 3. lista a DODS-eljárásokat használó SQL-lekérdezéseket az Enhydra QueryBuilder segítségével hozta létre. Első lépésként elkészítjük az egyik önműködően létrejövő osztály, a `BlogEntriesQuery` egy példányát:

```
BlogEntriesQuery blogq = new
      BlogEntriesQuery();
```

A jelen pillanatig az összes sort le szeretnénk kérdezni, fordított időrendben:

```
blogq.setQueryEntrydate("NOW()",
      ➥QueryBuilder("LESS_THAN");
blogq.addOrderByEntrydate(false);
```

Olyan eljárások is léteznek, amelyekkel a `WHERE` kifejezést is a lekérdezésünkhöz szúrhatjuk, így már meglehetősen összetett lekérdezéseket alkothatunk.
Végül az egyező sorok halmazát kapjuk vissza, amelyek mindegyike egy-egy `BlogEntriesDO` objektum formájában jelenik meg:

```
BlogEntriesDO[] blogEntries =
      ➥blogq.getDOArray();
```

Mivel csak a legfrissebb adatot szeretnénk megjeleníteni, a tömbnek egyszerűen az első elemét vesszük. A szöveget az XMLC által létrehozott „welcome” objektum eljárásának segítségével illesztjük a dokumentumunkba:

```
Welcome.setTextDate(blogEntries[0].getEntrydate());
Welcome.setTextText(blogEntries[0].getText());
```

Ha a módosítással készen vagyunk, futtassuk le a `make`-et a projekt legfelsőbb szintű könyvtárából. Ha Java programunkban bármilyen hibát találunk, ahányszor csak akarjuk, kijavíthatjuk, majd újrafuttathatjuk a `make`-et.
Elméletileg – a kimeneti könyvtárba lépve és a `./start` parancsot futtatva – most már elindíthatjuk a programot. Ha valóban megtesszük, láthatjuk, hogy próbálkozásunk kudarcot vall, mivel az alkalmazás még nem tudja, hol keresse a ***PostgreSQL .JAR*** fájlt. Az is hasznos, ha az első használatkor az Enhydrát (illetve bármilyen más alkalmazást) teljes körű hibakeresési kimenettel indítjuk, hiszen így az esetleges hibákat sokkal gyorsabban fel tudjuk deríteni és ki tudjuk javítani.

Három fájlt kell módosítanunk, hogy működésre bírjuk a dolgokat. Először módosítsuk az *$ENHYDRA/bin/multiserver* fájlt: a PostgreSQL JDBC meghajtó .JAR fájljára helyezzünk el egy hivatkozást. Ezt egyszerűen a multiserver fájl megváltoztatásával érhetjük el (ez tulajdonképpen egy héjprogram, ami a Java programot hívja meg). A `Build up classpath` megjegyzéssor alatti részeket a következőre változtatjuk:

```
# Where is the PostgreSQL JDBC .jar file?
PG_JDBC=/usr/share/pgsql/jdbc7.1-1.2.jar

if   [ X${CLASSPATH} = "X"] ; then
      ➥NEWCP=${ENHYDRA_CLASSES}${PS}${PG_JDBC}
else

NEWCP=${ENHYDRA_CLASSES}${PS}${CLASSPATH}${PS}
      ${PG_JDBC}
fi
```

Következő lépésként be kell állítanunk a *blog.conf* fájlt. Minden Enhydra-projekt tartalmaz egy beállításfájlt, amely számos más érték mellett meghatározza, hogy a rendszer milyen adatbázist használjon. Az én esetemben ez a beállításfájl a *blog/output/conf/blog.conf* névre hallgat, és az alkalmazással kapcsolatos rengeteg név-érték párost tartalmaz.
Módosítanunk kell a `Database manager` rész néhány részletét, hogy a mi programunkra mutasson. A 4. listában (24. CD Magazin/DODS könyvtár) megtaláljuk ezt a részt – abban a formában, ahogy annak lennie kell.
Végül módosítjuk a *servlet.conf*-ot. Annak ellenére, hogy nem kell feltétlenül megváltoztatnunk, hasznos dolognak tartom a következő két sorban a hibakereső részek bekapcsolását:

```
Server.LogToFile[] = EMERGENCY, ALERT,
      ➥CRITICAL, ERROR,WARNING, INFO, DEBUG
Server.LogToStderr[] = EMERGENCY, ALERT,
      ➥CRITICAL, ERROR,WARNING, INFO, DEBUG
```

A legfontosabb, amit a *blog.conf* és *servelet.conf* fájlokról tudni kell, hogy újra létrejönnek, valahányszor csak legfelsőbb szintű `make` parancsot adunk ki. Ezért ha ilyen módszerrel változtattuk meg a fájlokat, többet semmi esetre se adjuk ki a legfelsőbb szintű `make` parancsot. Ha megtesszük, meg fogjuk bánni (ahogy én is megbántam). Ehelyett inkább ezt a parancsot *presentation* könyvtárban adjuk ki.
Ha a beállításfájlokat megváltoztattuk, beléphetünk a *~/enhydraApps/blog/output* könyvtárba és futtathatjuk a `./start` parancsot. Láthatjuk, ahogy a kiszolgáló elindul, illetve (amennyiben a DEBUG lehetőséget beállítottuk a *servlet.conf*-ban, illetve a naplózást a *blog.conf*-ban) jó néhány hibakereső üzenetet tekinthetünk meg.
Ellenőrizhetjük is alkotásunkat. Állítsuk a böngészőt 9000-es kapura, amely az Enhydra-alkalmazások alapértelmezett kapuszáma: *http://localhost:9000/*. Ha minden jól megy, a böngészőben blogprogramunk kimenetét fogjuk látni.

Összegzés

A DODS jobb az Alzabónál, mivel kitűnő objektumréteget nyújt az SQL felett. Továbbá úgy tűnik, jobb és megbízhatóbb eljárásokat kínál az egyes lekérdezések összeállítására és az eredmények kezelésére. Ugyanakkor a DODS néhány szempontból ugyanazokban a betegségekben szenved, mint az Alzabo, vagy bármely egyéb relációs-objektum kapcsolatteremtő rendszer.
Az egyik ilyen gond, hogy új módszert kell megtanulnunk az (évek óta megszokott) SQL-lekérdezések elkészítésére, illetve a hosszabb lekérdezéseket meglehetősen kényelmetlen megírni, hiszen eljáráshívásokból kell összerakni őket. A DODS-szerű rendszerek általános használhatósága egyben azt is jelenti, hogy kedvenc adatbázis-kezelőnk különleges képességeit nem használhatjuk ki. Így például a PostgreSQL esetében a DODS úgy tűnik, teljesen figyelmen kívül hagyja az idegen kulcsokat és a számlálókat, amelyek pedig sokkal tömörebb adatszerkezetet eredményeznének.
Az Enhydra egyéb részeivel társítása a DODS azonban kitűnően működik. Akárcsak az XMLC és a superservletek esetében, a DODS-t is előbb letaglózónak és ügyetlennek éreztem, utóbb viszont hasznosnak és okosnak. Első pillantásra az Enhydra DODS eszköze jó próbálkozás az objektum- és a relációs világ közötti szakadék áthidalására. Már alig várom az Enhydra Enterprise végső változatát, amely kétségtelenül újabb lökést ad majd a dolgoknak.
A következő hónapban belenézünk abba az egyre inkább előtérbe kerülő szabványba, amely nemcsak, hogy összeköti a relációs és az objektumvilágot, de kiszolgálóoldali Java-alkalmazásainkat tranzakció-kezelési képességekkel is felruházza. Az Enterprise JavaBeans szolgáltatásokat és adatokat nyújt a webalkalmazásoknak, és egyre népszerűbbé válik azok közt a webfejlesztők között, akik objektumokat szeretnének alkotni, használni, illetve adatbázisban tárolni, anélkül, hogy meg kellene erőltetniük magukat.

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

Az Enhydra fő honlapja a ➲ http://www.enhydra.org címen található. Az Enhydra 4.x más néven Enhydra Enterprise az ➲ http://www.enterprise.enhydra.org címen lelhető fel, a ➲ http://www.dods.enhydra.org címen pedig a DODS Projectről találunk néhány adatot.
Az Enhydráról szóló kitűnő bemutatót találunk a ➲ http://www.arsdigita.com/asj/enhydra címen, amelyet *Roger Metcalf*, az ArsDigita Corporation munkatársa írt.
Meglehetősen jó, bár kissé túlhaladott ismertetőt találunk az Enhydra, a DODS és a PostgreSQL használatáról az ➲ http://enhydra.enhydra.org/software/documentation/NewApps-DODS-Tutorial-PGSQL.html címen.
DODS és QueryBuilder témakörben két másik hasznos honlap: ➲ http://dods.enhydra.org/software/documentation/gettingStarted.html és ➲ http://www.dods.enhydra.org/project/faq/UsingTheGeneratedCode.html
Nagyon köszönöm a Lutris munkatársainak, különösképpen *David Young*-nak, hogy DODS-kérdéseimmel kapcsolatban ilyen segítőkészek voltak. Külön óriási köszönet illeti őket, amiért olyan nyugodtan tűrték a Federal Express és az izraeli iroda több mint egyhetes őrületét, akik egyszerűen nem hitték el, hogy egy program valóban ingyenes is lehet.