# Jelentések a weboldalon MySQL adatbázisból, CSS és Perl segítségével

A Maypole Futballklub alkalmazás kiegészítése egy egyszerű jelentéskészítő módszerrel.

A *Linux Journal 2005 márciusi* számában megjelent cikkben a *Maypole* rendszer használatával készítettünk webes adatbázis alkalmazást mindössze 18 sor *Perl* kód megírásával. A *Maypole* nyújtotta funkcionalitás egészen elképesztő egy fontos területet leszámítva, ez pedig a jelentések készítése. Így hát nekifogtam olyan technológiák kereséséhez, amelyekkel jelentéseket tudok készíteni a Futballklub alkalmazásból kinyert adatokkal. A célom az volt, hogy létrehozzak egy garnitúra gyakran használt jelentést, amelyeket a webes felületről lehet lekérdezni.

## Webes jelentések? Mi a teendő?

Webes jelentések számtalan módon készíthetők a szokásos kiszolgáló oldali programnyelvek használatával, mint például a *PHP*, *JSP*, *Perl* parancsfájlok, és még sok más egyéb. Más, önálló munkaállomásokon futó jelentéskészítő eszközöket is használhatunk, amelyek még arra is képesek, hogy *MySQL* adatbázison alapuló jelentést készítsünk *OpenOffice.org* segítségével. Minthogy a jelentéskészítéssel kapcsolatosan csak alapvető elvárásaim vannak, így a szellemi ráfordítást is próbáltam minimalizálni. Azt nem bánom, ha a jelentést előállító *SQL* lekérdezés reszelgetésével kell tölteni az időt, de ha már meg van írva, azt szeretném, ha egyből egy *HTML* táblát adna eredményül.
Ezt persze megoldhatjuk *Perl*-ben, a *DBI* és *DBD::mysql* modulok segítségével, a kód kézi hegesztésével elküldött *SQL* lekérdezéseken keresztül. Ezt követően újabb kódok megírásával „utófeldolgozhatjuk”, mielőtt újabb kódrészletek felhasználásával végre elkészítenénk a táblázatot.
Az én egyszerű elvárásaimnak ez túl sok munkát jelentett. Amire tényleg szükségem volt, az egy gyors és igénytelen megoldás. A cikk hátralévő részében ezt a bizonyos általam készített megoldást fogom részletezni.

## MySQL-t az adatok kinyeréséhez!

Miközben *Paul DuBouis* kiváló *MySQL* szakácskönyvét böngésztem, találtam egy parancssori kapcsolót, amely a szintén parancssorból átadott lekérdezés eredményét *HTML* táblázattá alakítja át (1.23-as recept, 33. oldal). Példa gyanánt nézzük a következő parancsot:

```
mysql  -e  "select name from
➥player"              \
    -u  manager  -ppwhere  CLUB
```

Ez a következő szöveges kimenetet eredményezi:

```
+-------------+
|name         |
+-------------+
|Robert Plant |
|Tim Finn     |
|James Taylor |
|Bryan Adams  |
|Ian Gillen   |
|Mick Jagger  |
|Neil Young   |
|Bob Dylan    |
+-------------+
```

Az eredményből a Futballklub összes játékosának neve kiderül, s úgy tűnik, hogy a klub játékosait híres folk és rock énekesek után nevezték el.
Ha a fenti parancsot újrafuttatjuk a *HTML* készítő kapcsolót alkalmazva,

```
mysql  -H -e  "select name from
➥player"              \
    -u  manager  -ppwhere  CLUB
```

akkor a következő szöveghalmazt kapjuk, ami – higgyék el nekem – egy *HTML* táblázat:

```
<TABLE BORDER=1><TR><TH>name
➥</TH></TR><TR><TD>
Robert Plant</TD></TR><TR><TD>
➥Tim Finn</TD></TR>
<TR><TD>James Taylor</TD></TR>
➥<TR><TD>Bryan Adams
</TD></TR><TR><TD>Ian Gillen
➥</TD></TR><TR><TD>
Mick Jagger</TD></TR>
➥<TR><TD>Neil Young</TD></TR>
<TR><TD>Bob Dylan</TD>
➥</TR></TABLE>
```

Az *SQL* lekérdezést eltehetjük egy fájlba is, aztán később hivatkozhatunk erre a fájlra parancssorból. Az alábbi példa – amelyben feltételeztük, hogy a fenti lekérdezés a *name.sql* fájlban található – ugyanazt a *HTML* táblát adja eredményül:

```
mysql  -H  -u  manager
➥-ppwhere  CLUB  <  name.sql
```

Ennek ismeretében azt találtam ki, hogy ha a *HTML* táblázatot előállító parancsot webes felületen keresztül indítanám el, akkor a nehezén rögtön túl is volnék, már ami a webes jelentéskészítő megoldásommal kapcsola-

tos problémákat illeti. Így hát készítettem egy *Perl* nyelven írt kis *CGI* parancsfájlt, amely a parancssoros utasításokat indítja.

## A CGI szkript

A *CGI* szkriptben használt stratégia lényegre törő: miután megállapítottuk, hogy mi a neve a futtatandó lekérdezésnek, egy parancsot rakunk össze, majd elindítjuk azt a programból. Ezt követően a parancs visszatérési érékét a *CGI* által létrehozott *HTML* fájl body részébe illesztjük. A szokásos *Perl* indítósorokat leszámítva a ***runquery.cgi*** parancsfájl egy sor konstans meghatározásával kezdődik:

```
#! /usr/bin/perl -w
use strict;
use constant MYSQL  => '/usr/
➥bin/mysql';
use constant USERID =>
➥'manager';
use constant PASSWD =>
➥'pwhere';
use constant DBNAME => 'CLUB';
```

Elképzelhető, hogy a *MySQL* ügyfélprogram helye máshol van, mint az én számítógépemen, ebben az esetben javítsuk a konstans értékét. Fontos még, hogy beledrótoztam a felhasználó (`USERID`), jelszó (`PASSWD`) és adatbázis (`DBNAME`) értékeket a kódba. Ez ugyan nem a legszebb megoldás, de szeretném kimagyarázni magamat: ez az igénytelen része a bevezetőben említett gyors és igénytelen megoldásomnak. A megadott konstansokból már látszik, hogy a *Perl CGI* programnyelv szabványos felületét fogjuk használni:

```
use CGI qw( :standard );
```

Két *Perl* változót definiálunk, amelyek a webes felületről a parancsfájlnak átadott paraméterek értékét veszik fel. Az első paraméter neve `query`, ez határozza meg, hogy melyik *sql* fájlt fogjuk használni. A másik neve `title`, ebben van a jelentés címe amit az eredmények megjelenítésénél használunk.

```
my $query = param( 'query' );
my $title = param( 'title' );
```

A szkript ezután elkészíti a parancsot, amely lefuttatja a megadott lekérdezést a *MySQL* kliensen keresztül. Ne feledjük, hogy a *Perl*-ben a ponttal jelölt művelet a karakterláncok összefűzését jelenti.

```
my $cmdline = MYSQL .
              ' -H -u ' .
              USERID .
              ' -p' .
              PASSWD .
              ' ' .
              DBNAME .
              "< $query ";
```

Ezek után felépítjük a *HTML* oldalt. A `header` (fejléc) függvény elkészíti a helyes `Content-Type` fejléceket, aztán a `start_html` függvény létrehozza a *HTML* oldalt, amelynek címe a paraméterként átadott oldalcím lesz:

```
print header;
print start_html( -title =>
➥$title );
```

A következő kódsor a *Perl* `qx` műveletét használja a parancs elindításához, s a parancs kimeneti értékével tér vissza, amelyet a `$result` nevű változóba teszünk.

```
my $results = qx/ $cmdline /;
```

A parancsfájl további része egy 3. szintű címsort tesz a weboldalra, a lekérdezés eredményével együtt, majd egy *HTML* hivatkozást, amely visszamutat a jelentések oldalra. Az `end_html` függvény lezárja a *HTML* oldal generálását, és befejezi a szkript futását.

```
print "<h3>$title</h3>";
print $results;
print p, "Return to the ",
      a( { -href => "/Club/
➥Reports.html" },
      "List of Reports" );
print end_html;
```

## A parancsfájl hívása

A szkript futtatásához két dolgot kell tennünk: el kell helyezni a parancsfájlt olyan helyre, ahonnan a webkiszolgáló eléri, valamint létre kell hozni az *SQL* lekérdezést tartalmazó fájlt. Az általam használt *Fedora Core 3* terjesztés *Apache 2* webkiszolgálót futtat, és a */var/www/cgi-bin/* könyvtárban tárolja a *CGI* parancsfájlokat. Egyszerűen csak át kell másolni a *CGI* fájlt erre a helyre, majd futtathatóvá kell tenni:

```
cp  runquery.cgi  /var/www/
➥cgi-bin/
chmod  +x  /var/www/cgi-bin/
➥runquery.cgi
```

A fenti elérési út lehet, hogy nem ugyanaz, mint amit az olvasó rendszere is használ, legelőször ennek járjunk utána. Lekérdezés gyanánt pedig használjuk az alábbit, amelyet a ***conditions.sql*** fájlba mentsünk:

```
select player.name as 'Player',
       condition.name as
➥'Medical Condition'
from   player, condition
where  player.medical_condition
➥= condition.id and
       player.medical_condition
➥!= 1;
```

Ez a lekérdezés összekapcsolja a `player` (játékos) és `condition` (erőnlét) táblákat, hogy ki tudja listázni az egyes játékosokat és a hozzájuk tartozó egészségügyi erőnlétre vonatkozó információt – feltételezve, hogy mindenkihez csak egy ilyen adat tartozik. A lekérdezést tartalmazó fájlt még át kell másolnunk a webkiszolgáló *CGI* könyvtárába:

```
cp conditions.sql /var/www/
➥cgi-bin
```

A lekérdezés *CGI* szkripten keresztül történő indításához gépeljük be a böngészőnk címsorába a következőt (a `localhost` tagot helyettesítsük a webkiszolgálónk tartománynevével):

```
http://localhost/cgi-bin/
➥runquery.cgi?           \
     title=Results&query=
➥conditions.sql
```

Az *URL* az ***1. ábrán*** látható eredményt állítja elő, amely a kevés előkészület ellenére is teljesen jónak tűnik, igaz lehetne szebb is.

## Tegyük szebbé a dolgokat CSS használatával

Ahhoz, hogy egy tökéletesebb kinézetű jelentéshez jussak, készítettem egy kis *CSS* (*Cascading Style Sheet*) fájlt,

amelyet ***reports.css***-nek neveztem. Ez majd rendbe hozza a jelentésünk általános kinézetét.

```
body {
    font-family:    sans-serif;
}
table {
    font-family:    sans-serif;
    background-color:
➥LIGHTYELLOW;
}
table th {
    background-color:
➥LIGHTCYAN;
    font-size:      75%;
}
h3 {
    font-family:    sans-serif;
    color:          BLUE;
}
```

A stíluslapok működéséből adódóan a fájl meglehetősen egyszerű. Először is meghatározzuk az oldal alap betűtípusát, aztán trükközünk kicsit a jelentést alkotó táblázat hátterével és betűtípusával. A táblázatok fejléce 75%-a az alapértelmezett betűméretnek, és a táblázat adatainak háttérszíne is eltérő. Aztán megadjuk, hogy a jelentésben használt 3-as szintű címsor kék színű legyen.
A *CSS* fájlt a webkiszolgáló gyökérmappájába kell másolni, ahol a webalkalmazások is láthatják:

```
cp reports.css /var/www/html
```

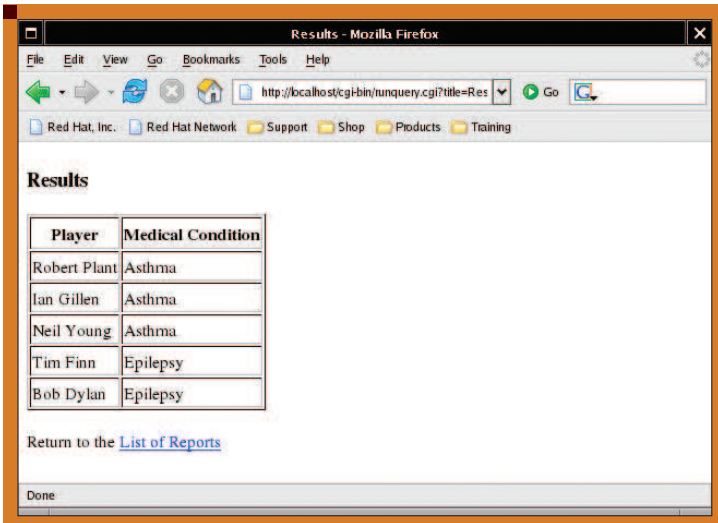
A *CSS* fájl használatához meg kell változtatnunk a ***runquery.html*** fájlban található `start_html` függvény paraméterezését, hogy az hivatkozzon a stíluslapra:

```
print start_html( -title =>
➥$title,
                  -style => {
➥-src => "/reports.css" } );
```

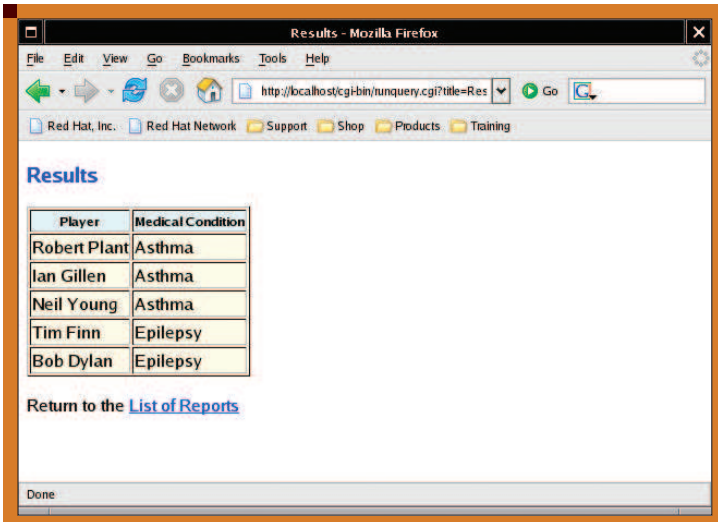
A *CGI* szkriptet újratöltve a *2. ábrán* látható eredményt kapjuk. Talán nem fog webdizájn díjat nyerni, de sokkal jobban néz ki, mint az *1. ábrán* látható egyszerű kimenet.

## A webes felület elkészítése

A megoldásnak ez a része igen egyszerű. Egy sima weblapra van szükségem, amely a jelentések listáját tartalmazza. Hasonlóan, mint az előállított jelentések esetében, az előbbi nem túl bonyolult stíluslapot fogom használni a kinézet szebbé tételéhez. Íme a ***HTML***, amelyet készítettem:

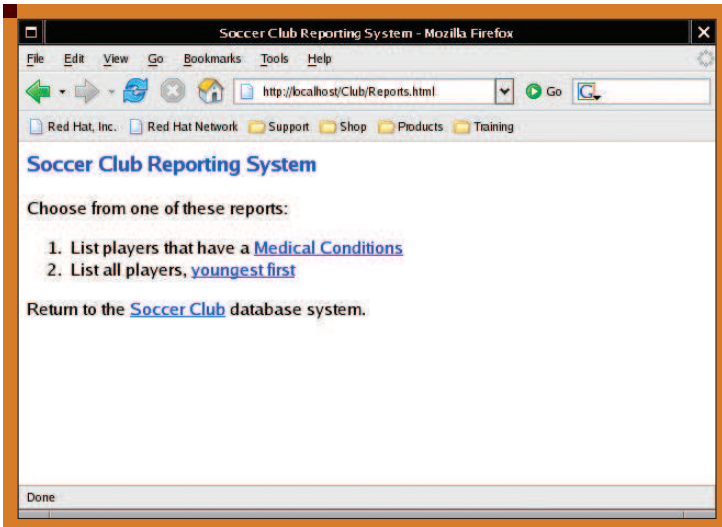



*1. ábra* Egy működőképes, bár nem túl szép HTML jelentés




*2. ábra* Egy sokkal tökéletesebb HTML jelentés

```
<HTML>
<HEAD>
    <TITLE>Soccer Club
➥Reporting System</TITLE>
    <LINK rel="stylesheet"
➥type="text/css"
          href="/reports.css" />
</HEAD>
<BODY>
<H3>Soccer Club Reporting
➥System</H3>
Choose from one of these
➥reports:
<OL>
    <LI>Mutasd azokat a játé-
➥kosokat, akiknek elérhető az
    <a href="/cgi-bin/
➥runquery.cgi?
    title=Players with a
➥Medical Condition&
    query=conditions.sql">
➥erőnléti eredménye</a>
    <LI>Mutasd az összes
➥játékost,
    <a href="/cgi-bin/
➥runquery.cgi?
    title=Listing of all
➥Players (Youngest First)&
    query=desc_dob.sql">kezdd a
➥fiatalokkal</a>
```




*3. ábra* A Jelentéskészítő felülete

```
</OL>
Vissza <A HREF="/Club">Soccer
➥Club</A>
az adatbázishoz.
</BODY>
</HTML>
```

Ahogy már láttuk, minden jelentést két paraméterrel hívunk meg: az egyik a `title`, amely a jelentés címét tartalmazza, a másik a `query`, amely azonosítja a lekérdezést, amelyet a *MySQL* ügyfélprogram segítségével le kell futtatni. A weblapot az elkészítés után másoljuk a Futballklub weboldalunk gyökérmappájába:

```
cp Reports.html /var/vvv/html/
➥Club
```

A böngészőből betöltve a jelentéskészítő webes felület a *3. ábrán* látható formában jelenik meg.
Ezen a ponton azt hiszem, készen vagyunk. Van egy egyszerű webfelületünk egy alap jelentéskészítő mechanizmushoz. Ha később több lekérdezést is írunk, elhelyezhetjük őket ***sql*** fájlokban egyenként, majd bemásolhatjuk a ***cgi-bin*** könyvtárunkban, frissíthetjük a Jelentéskészítő *HTML* kódját, hogy el tudjuk indítani ezeket a lekérdezéseket. A megoldás gyors és igénytelen, de több, mint elegendő.

## Vagy mégsem?

A megoldásom biztonsági vonatkozását tekintve igen gyenge. Két problémára kell megoldást találnom: meg kell védenem a CGI és SQL parancsfájlokat attól, hogy a felhasználók trükközhessenek vele, valamint meg kell védenem a rendszert a CGI parancsfájloktól

## Biztonság: Védelem a felhasználói belebabrálásoktól

Bár a *CGI* és *SQL* fájlokkal kapcsolatos felhasználói piszkálódásokról beszélünk, tudni kell, hogy a probléma minden olyan fájl esetében megvan, amelyek olvashatók a webkiszolgálón héj fiókkal rendelkező felhasználók számára. Egy sima `cat` vagy `less` paranccsal megnézhetjük a tartalmát. Minden felhasználó belenézhet a ***runquery.cgi*** fájlba, és kiolvashatja az adatbázishoz tartozó felhasználónév - jelszó párost , ami nem szerencsés.
Az ***User*** és ***Group*** tulajdonságok az ***Apache httpd.conf*** beállítási fájljában meghatározzák, hogy melyik felhasználó és csoport nevében fut a webkiszolgáló. A saját rendszeremen ez a felhasználónév és csoportnév az `apache`. Ennek ismeretében adjunk ki olyan parancsot, amely beállítja, hogy a *CGI* és *SQL* fájljaink az `apache` felhasználó legyen a tulajdonosa, valamint csak ez a bizonyos `apache` felhasználó tudja ezeket a fájlokat írni és olvasni. Ez a rendszergazda (root) felhasználó kivételével mindenki számára megakadályozza, hogy hozzáférjen a fájlok tartalmához.

```
cd  /var/www/cgi-bin
chown  apache:apache  *
chmod  600  *
chmod  700  *.cgi
```

Az első `chmod` utasítás hatására a ***cgi-bin*** könyvtárban található tartalom kizárólag az `apache` felhasználó számára lesz olvasható. A második `chmod` utasítás az összes *CGI* fájlra bebillenti a futtatható bitet, de csak a fájl tulajdonosa számára. Ezzel az egyszerű elővigyázatossággal a jelentéskezelőnk biztonságban van az illetéktelen felhasználói hozzáférésektől.

## Biztonság: Védelem a CGI parancsfájlokkal szemben

A fenti `chmod` utasítások megvédik a fájlokat azoktól a felhasználóktól, akiknek héj fiókjuk van a kiszolgálón, de a jelentéskezelőnk még mindig sebezhető. Sajnos ezt minden felhasználó kihasználhatja egy egyszerű webböngésző segítségével. Tekintsük például, hogy mi történik, ha az alábbi *URL* használatával futtatjuk a *CGI* parancsfájlt:

```
http://localhost/cgi-bin/
➥runquery.cgi?            \
title=Ha!&query=conditions.sql
➥| cat runquery.cgi
```

A *CGI* parancsfájl tartalma megjelenik a böngészőnkben, és a támadó könnyedén kiolvashatja az adatbázis nevét, valamint a felhasználónév illetve jelszó párost. Ez már önmagában elég baj, de képzeljük el mi történik, ha a fenti *URL*-ben szereplő `cat runquery.cgi` szakaszt erre cseréljük:

```
cat /etc/passwd
```

vagy egy katasztrófával fenyegető változatra:

```
rm -rf /
```

Az általunk írt *CGI* szkripttel az a baj, hogy vakon megbízik a használójában, hogy az nem fog trükközni az *URL*-lel. Egy egyszerű csővezeték (pipe) karakter, és egy bármilyen héj parancs hozzáírásával a felhasználó kihasználhatja az átgondolatlanul megírt *CGI*-ből eredő biztonsági hibát: szó szerint bármilyen parancsot futtathat a kiszolgálón. A paraméterlista változatlan formában történő átadása az operációs rendszer számára nagyon sebezhetővé teszi a *CGI* parancsfájlokat.

Hála az égnek, a *Perl*-nek van egy különleges működési módja, amely segíthet nekünk. Ennek a neve ***taint*** (fertőzés) mód. A legtöbb *Perl*-lel foglalkozó könyv ismerteti a ***taint*** mód használatát, sőt, ***Christiansen*** és ***Torkington Perl*** szakácskönyve egy kényelmes megoldást is bemutat (19.4 recept, 767. oldal). A ***taint*** mód bekapcsolásával arra utasítjuk a *Perl* értelmezőt, hogy ne bízzon meg abban az adatban, amely a parancsfájlon kívülről származik. Mivel az adat megbízhatatlan, fertőzöttnek tekinti, a *Perl* a nem biztonságos változó felhasználása során egy futásidejű kivételt generál.
A ***Perl taint*** módja a *CGI* parancsfájlunk legelső során megváltoztatásával, a ***taint*** mód kapcsoló hozzáadásával érhető el.

```
#! /usr/bin/perl -wT
```

Ha újratöltjük a trükkös ***URL***-t, az eredményoldal üres lesz, és az ***Apache*** hibanaplója is gazdagodik egy 'nem biztonságos függőségi hiba' bejegyzéssel. A *Perl* így adja tudtunkra, hogy a szkript futása meghiúsult fertőzéses hiba okán. A futás meghiúsulása kétségtelenül megszünteti a rendszer biztonsági kockázatát, bár hozzá kell tenni, hogy a szkript így már nem látja el azt a feladatát, amire terveztük, azaz a megoldás minden, csak épp nem használható. Hogy újra életet leheljünk bele, meg kell tisztítanunk a bemeneti adatokat a *Perl* szabályos kifejezések használatával. Az ötlet igen egyszerű: meghatározunk egy a biztonságos adatnak megfelelő mintát, ezt a mintát illesztjük aztán a bejövő adatra, s abban az esetben, ha a minta illeszkedik, megbízhatóként kezelhetjük tovább. A *CGI* parancsfájlunknak két bemeneti adata van: a `query` és a `title` nevű paraméterek. A következő szabályos kifejezés hozzáadásával ellenőrizhetjük a bemenetek megbízhatóságát.

```
$query =~ /^([-\w]+\.sql)$/;
$query = $1;
$title =~ /^([\w:.?! ]+)$/;
$title = $1;
```

Az első szabályos kifejezés arra a karaktersorozatra illeszkedik, amely betűket és kötőjeleket tartalmaz tetszőleges elrendezésben, ponttal a végén, majd s q és l betűkkel zárva. Minden más, nem illeszkedő karaktersorozatot gyanúsnak tekintünk. Ha illeszkedik a mintára, a *Perl* a `$1` változóban fogja tárolni, amit aztán visszaírunk a most már megbízható `$query` változóba.
A cím paraméter mintája minden betűt elfogad, ezen kívül azokat a karaktereket is, amelyeket a fenti kifejezésben szögletes zárójelek között találunk. A `$title` változó szintén megbízhatóvá válik, ha a *Perl* illeszkedést észlel a futás során.
Mivel a *CGI* külső futtatható parancsot indít, nevezetesen a ***MySQL*** ügyfélprogramot, a futtatókörnyezet elérési útját szintén megbízhatóvá kell nyilvánítani. Ezt a `PATH` változó megfelelő értékre történő állításával érhetjük el, amely a biztonságos mappákat fogja tartalmazni:

```
$ENV{'PATH'} = "/usr/bin";
```

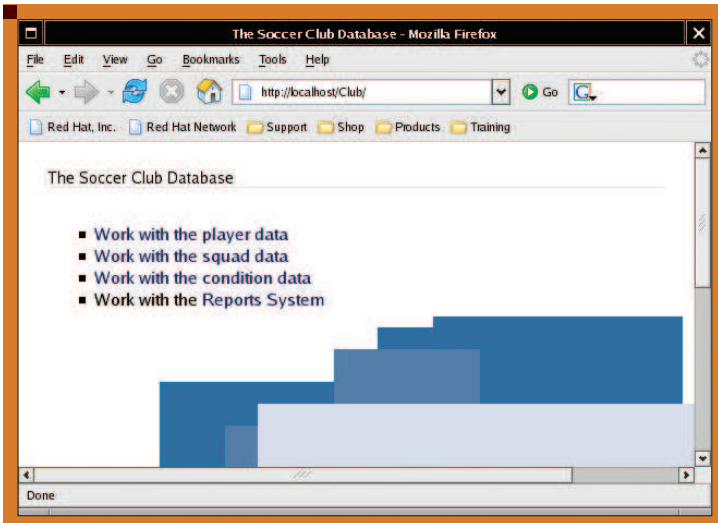
A fenti változtatásokkal biztonságossá tettük a ***runquery.cgi*** parancsfájlt, amely ismét használható. Ezen kívül egyszerű és igénytelen, és biztonságos a felhasználói támadásokkal szemben, a jelentéskezelő megoldásunk nem jelent többé potenciális veszélyt a rendszerünk számára.

## A jelentéskészítő illesztése a Maypole alkalmazáshoz

Ahhoz, hogy hivatkozhassunk a Futballklub alkalmazásból a jelentéskezelőre, meg kell változtatni a ***custom/frontpage*** sablont, egy új listával bővítve, amely a jelentéskezelő oldalára mutat.

```
<ul>
[% FOR table =
➥config.display_tables %]
    <li>
        <a href="[%table%]/
➥list">Munka a(z)
                    [%table %]
➥tábla adataival</a>
    </li>
[% END %]
    <li>Munka a <a href=
➥"Reports.html">
          Jelentéskezelővel</a>
</ul>
```

Amikor betöltjük az alkalmazást a böngészőnkben, a hivatkozás a ***Maypole*** menü részeként jelenik meg, ahogy az a *4. ábrán* is látható. A webalapú jelentéskezelő rendszerünk egyszerű, biztonságos és könnyen bővíthető. Ehhez csupán annyi dolgunk van, hogy írjunk még néhány ***SQL*** lekérdezést.




***4. ábra*** A Jelentéskezelő webfelületének beillesztése a Maypole alkalmazásba

*Linux Journal 2006., 149. szám*

***Paul Barry*** (paul.barry@itcarlow.ie) előadó a Carlow-i Műszaki Egyetemen, Írországban. Az előadásainak anyagai, valamint az általa írt könyvek és cikkek megtalálhatók a weboldalán: glasnost.itcarlow.ie/~barryp.