Using Perl with MySQL

Originally written by Larry Wall as a supplement to the USENET reader, Perl ("Practical Extraction and Report Language") is used widely for scripts that run from operating system prompts, that send and receive email, and that generate dynamic web pages. It combines the syntax of many traditional command-line languages—C, C++, sed, awk, grep, sh, and csh—into a tool that is famously flexible, and compact or even terse. It is useful especially for list processing.

If you are new to Perl, you might want to supplement reading this chapter with working through an *introductory tutorial*. Perl scripts that generate web content require a web server, of course. If you will use Apache but have yet to *install it*, now is the time. And if you have not yet *created the tracker database*, do it before trying this chapter's examples.

Installing Perl

The simplest approach is to install *XAMPP*. For independent Perl installation, *ActiveState Perl* is now pretty much the standard implementation. Download the current build for your operating system from *http://www.activestate.com/Products/ActivePerl/*. Build numbers `6xx` are for version `5.6.x`; builds `8xx` are for versions `5.8.x`, which is recommended. Under Red Hat Linux 6.2 or later, install ActiveState Perl with

```
% rpm -i ActivePerl-<VERSION_NUMBER>.<BUILD_NUMBER>-i686-linux.rpm
```

For other Linux versions, you may find you have the generic installer, which can be installed by an unprivileged user:

```
% tar zxf ActivePerl-<VERSION_NUMBER>.<BUILD_NUMBER>-i686-linux.tar.gz
% cd ActivePerl-<VERSION_NUMBER>
% ./install.sh
```

whatever location you specified. Call the Perl installation path *`<PERL_INSTALL_DIR>`*. Add *`<PERL_INSTALL_DIR>/bin`* to the `PATH` environment variable, for example:

```
% setenv PATH /usr/local/ActivePerl-<VERSION_NUMBER>/bin:${PATH}
```

ActivePerl for Windows comes as a .MSI file, *ActivePerl-`<VERSION_ NUMBER>`-`<WIN_VERSION_NUMBER>`-x86.msi*. If Apache is running, bring it down. Double-click on the installer file and let it install to its default folder, *%systemdrive% \perl*. If you will be running IIS rather than Apache, be sure to enable installation of ActivePerl/Perl/ ISAPI. ActivePerl will automatically add *%systemdrive%\perl\bin* to the environment `PATH` variable and will associate *.pl* files with Perl.

ActiveState Perl documentation is at *`<PERL_INSTALL_DIR>/html/index.html`*. Installation details are in *`<PERL_INSTALL_DIR>/html/index.html`*.

No matter what your operating system is, you need several Programmer's Package Manager (PPM) modules. From a command prompt, run …

```
ppm query *
```

and search the list for `Data-Dump`, `DBD-mysql` and `DBI`. For missing packages, run …

```
ppm install <packagename>
```

`DBI` is the Perl *Database Interfac*e layer, which liberates Perl code from having to deal with the details of particular database drivers. `DBD-mysql` is the Perl *Database Driver* for MySQL, and oddly it is often missing not only from ActivePerl installations but from the current PPM repository `PPM3`, so for `DBD-mysql` you may have to run …

```
  ppm rep add PPM2 http://ppm.activestate.com/PPMPackages/5.6plus/
  ppm rep 2
  ppm install DBD-mysql
```

If this fails because your operating system cannot find *ppm*, *`<PERL_INSTALL_DIR>`/bin* is not yet in the `PATH`. Correct that, reboot and try again.

Perl and CGI

Perl is a scripting language. Under Linux/UNIX, a Perl script can run if its first line says

```
#!<PERL_INSTALL_DIR>/bin/perl
```

for example

```
#! /usr/bin/perl
```

and if the following command has executed against a Perl script:

```
chmod +x <scriptName>
```

Under Windows, the script's first line should also point at the Perl executable, for example

```
#! c:/perl/bin/perl.exe
```

and the script runs with the command

```
perl <scriptName>
```

When a Perl script runs in a command-line environment, the Perl interpreter's output to `stdout` comes to the terminal window. Thus running *test.pl*,

```
#! /usr/bin/perl
print "This is from source file test.pl\n";
```

from a command prompt produces

```
This is from source file test.pl
```

Web deployment

If we format *test.pl*'s output for HTML,

```
#! /usr/bin/perl
print "Content-type: text/html\n\n";
print "<html><body>This is from source file test.pl</body></html>";
```

we *deploy the script as a web page* for Apache 2 by dropping the script into the Apache 2 *cgi-bin* directory. On Linux/UNIX, but not under Windows, deployment also requires that we execute these two commands:

```
chmod 500 test.pl
chown <apache_user> test.pl
```

where `<apache_user>` is the Linux/UNIX username under which Apache runs. Then simply browsing `http://localhost/cgi-bin/test.pl` produces the script's output in your web browser.

Common Gateway Interface (CGI) was the first standard for generating dynamic Web content. Web servers recognise CGI requests in various ways, for example by finding */cgi-bin/* in the request URL, or by finding a file extension of *.cgi.* In response to a CGI request, the Web server spawns a child thread to process the script and send HTML output to `stdout`. The Web server captures this output and redirects it to the client's browser.

The CGI model has severe drawbacks. The request language, usually C or Perl, must be procedural. These languages are compact, but development of complex data-driven web content with either of them tends to be numbingly literal, and is code-intensive. A program crash can bring down the machine. Every request spawns a new thread. CGI provides for no user-centric services, for example remembering who the client is, or authentication protocols. New, more powerful, more flexible, and less computationally expensive arrangements therefore followed quickly, notably Java servlets and JavaServer Pages, PHP, ASP and .NET. A lot of Perl/CGI is still running out there, though, and more is being written.

Perl:DBI

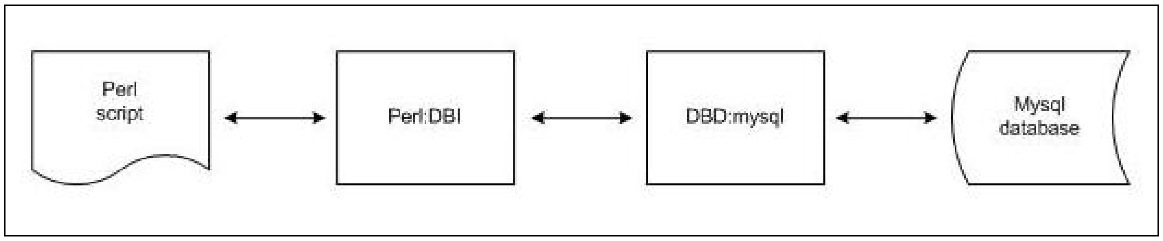

Fig 13-1: Perl Connection with MySQL

A Perl/MySQL script writes to, and reads from, the Perl DBI layer, thus insulating itself from the details of the DBD layer (Fig 13-1). Provided you do not call DBI methods that are specific to the DBD for a particular database, changing a script from using one database driver to another, for example from Sybase to MySQL, entails changing exactly one line of code.

DBI queries

The usual sequence is: connect to the database, prepare an SQL statement, execute it, process the result, report the result, and tidy up. Here is perhaps the simplest Perl DBI query script you can write. In it, change `USR` and `PSWD` to appropriate values for your

TOC ***Previous*** ***Next***