Acceso a bases de datos en Perl

Martín Ferrari

16 de Julio de 2003

Resumen

Una introducción a la interfaz de bases de datos DBI de Perl.

1. Qué es una Base de Datos

El concepto de base de datos es simplemente una forma de guardar información y de obtener datos.

2. Qué es DBI

DBI significa: DataBase Interface. Es decir, interfaz de base de datos.

Las principales características de DBI son: que provee una interfaz homogénea de acceso a bases de datos, y que es modular -cualquiera puede agregar soporte para una nueva base de datos fácilmente-.

Gracias a esto no necesitamos hacer grandes cambios en un programa si decidimos, por ejemplo, migrar una aplicación que accedía una base de datos MySQL a PostgreSQL. O si queremos un cambio más radical podemos decidir dejar de depender de un motor de base de datos y utilizar la interfaz de acceso a archivos CSV (*Comma Separated Values* o valores separados por comas) que con los mismos comandos SQL de MySQL o PostgreSQL nos permite manipular los datos guardados en un archivo de texto!

3. Estructura de DBI

DBI nos proveé una interfaz de objetos; no debemos preocuparnos mucho por ello, ya que dentro de perl el acceso a objetos es sencillo, e intuitivo para cualquiera que haya usado objetos de otro lenguaje.

Es importante tener en cuenta un par de datos:

- Los constructores sólo se diferencian de los métodos comunes en que devuelven una nueva instancia del objeto. Su nombre no está prefijado por el lenguaje, por ejemplo en DBI el constructor se llama *connect*.
- La invocación de métodos y constructores se hace con el operador flecha (`->`), a la izquierda va la referencia al objeto o clase y a la derecha el método o contructor.

  Por ejemplo: `$dbh = DBI->connect(...)`, `$dbh->disconnect()`

El objeto que maneja la clase DBI es un manejador de conección a la base de datos por lo tanto es lógico que el constructor sea *connect*.

Para más referencia, en sistemas UNIX se puede consultar las páginas man de: perlobj(1), perltoot(1) y perltooc(1). [1]

DBI en sí no puede conectarse a ninguna base de datos, sólo proveé la interfaz unificada. Para el acceso real a base de datos, carga en el momento de conexión el controlador específico solicitado por el programador. Los controladores son llamados DBD (*Data Base Driver* o controlador de base de datos) y todos se llaman `DBD::xxxx`.

En la siguiente lista se pueden ver algunos controladores DBD, no es una lista extensiva de controladores disponibles.

`DBD::CSV`	Acceso a archivos CSV con consultas SQL
`DBD::Excel`	Acceso a archivos de Excel con consultas SQL
`DBD::LDAP`	Acceso a directorios LDAP con consultas SQL
`DBD::mysql`	Acceso a bases MySQL
`DBD::ODBC`	Acceso a bases de datos usando ODBC
`DBD::Pg`	Acceso a bases PostgreSQL
`DBD::RAM`	Acceso generalizado a estructuras en memoria o archivos utilizando consultas SQL
`DBD::SQLite`	Acceso a bases SQLite, un motor relacional autocontenido en una biblioteca
`DBD::Sybase`	Acceso a bases Sybase y MS-SQL

4. Usando DBI

Aunque DBI no exige el uso de SQL como lenguaje de consultas, está muy influenciado por él y todos los manejadores que conocemos usan SQL (emulándo-

[1] En entornos no-UNIX, por lo general esta documentación se encuentra en otro formato; en última instancia siempre se puede consultar en http://www.perldoc.com/

lo si hace falta, como en los casos de `DBD::Excel`, `DBD::LDAP` o `DBD::CSV`, o simplemente pasando los comandos al motor respectivo).

4.1. Conexión a la base de datos

Connect espera al menos un parámetro que es la especificación de conexión (DSN) que es una cadena formada por `"dbi:"` segido del nombre del manejador deseado y luego un string dependiente del manejador, que especifica otros datos como nombre de servidor, nombre de base de datos, etc. Opcionalmente, luego del DSN, van el nombre de usuario y clave para acceder a los datos.

Ejemplo:

```
$dbh = DBI->connect("dbi:mysql:database=testdb;host=localhost",
$user, $password);

$dbh = DBI->connect("dbi:SQLite:dbname=/var/lib/base", "", "");

$dbh = DBI->connect("dbi:CSV:f_dir=/var/lib/basecsv/", "", "");
```

4.2. Lectura de datos

Una vez que obtuvimos una conexión a la base, podemos empezar a recuperar datos.

Depende de cómo querramos recuperar los datos, cuantas veces tengamos que repetir la operación y otras variables, podremos usar una de varias maneras de trabajar.

Veremos una manera simple de recuperar los datos de un select de a una fila. Primero componemos el comando, lo que nos da un manejador de comando (`statement handler`) y con ese manejador, ejecutamos el comando y luego recuperamos los datos.

```
$sth = $dbh->prepare("
    SELECT nombre, apellido, cumple
    FROM amigos
");
$sth->execute;
while(@datos = $sth->fetchrow_array()) {
    print "$datos[0] $datos[1] cumple años el $datos[2]\n";
}
```

4.3. Modificación de datos y otros comandos

Cuando no necesitamos recuperar datos, la secuencia es más sencilla, invocamos un sólo método que hace todo el trabajo: do.

```
$filas_cambiadas = $dbh->do("
    UPDATE amigos
    SET sexo = 'm'
    WHERE nombre = 'Juan'
    ");

$dbh->do("DROP TABLE amigos");
```

4.4. Más poder

DBI es mucho más que todo esto. Algunas características importantes:

- Sistema para capturar y reportar errores configurable según las necesidades del programador.
- Manejo de transacciones, *commit*s y *rollback*s.
- Diversas maneras de recuperar datos de SELECT que se adaptan a distintas situaciones, y consumos de memoria.
- Manejo de múltiples conexiones a bases distintas o a la misma base, utilizando instancias independientes del objeto.

5. Ejemplo

Un ejemplo de uso de DBI: extrae los datos de una tabla de MySQL y los ingresa en una "tabla" CSV (usando el manejador de CSV, cada tabla es un archivo).

```
#!/usr/bin/perl

use warnings;
use strict;
use DBI;

my($host, $base, $usuario, $clave, $tabla) =
    qw/localhost test apache telecom9 tabla1/;
```

```
my($dircsv, $archcsv) = ("./", "datos_de_mysql");

my($dbh1, $sth1, $dbh2, $sth2, @fila, @columnas);

$dbh1 = DBI->connect("dbi:mysql:host=$host;database=$base", $usuario, $clave,
    { RaiseError => 1, AutoCommit => 1 });

# DESCRIBE devuelve Field, Type, Null, Key, Default, Extra
$sth1 = $dbh1->prepare("DESCRIBE $tabla");
$sth1->execute();
while(@fila = $sth1->fetchrow_array()) {
    push(@columnas, $fila[0]);
}

$dbh2 = DBI->connect("dbi:CSV:f_dir=$dircsv", "", "",
    { RaiseError => 1, AutoCommit => 1 });

$dbh2->do("CREATE TABLE $archcsv (" . join(", ", map("$columnas[$_] varchar(255)",

$sth2 = $dbh2->prepare("INSERT INTO $archcsv (" . join(', ', @columnas) .
    ") VALUES (" . join(", ", map("?", @columnas)) . ")");

$sth1 = $dbh1->prepare("SELECT * FROM $tabla");
$sth1->execute();
while(@fila = $sth1->fetchrow_array()) {
    $sth2->execute(@fila);
}
```