PostgreSQL - Perl Interface

Advertisements

Previous Page Next Page

Installation

The PostgreSQL can be integrated with Perl using Perl DBI module, which is a database access module for the Perl programming language. It defines a set of methods, variables and conventions that provide a standard database interface.

Here are simple steps to install DBI module on your Linux/Unix machine –

```
$ wget http://search.cpan.org/CPAN/authors/id/T/TI/TIMB/DBI-1.625.tar.gz
$ tar xvfz DBI-1.625.tar.gz
$ cd DBI-1.625
$ perl Makefile.PL
$ make
$ make install
```

If you need to install SQLite driver for DBI, then it can be installed as follows –

```
$ wget http://search.cpan.org/CPAN/authors/id/T/TU/TURNSTEP/DBD-Pg-2.19.3.tar.gz
$ tar xvfz DBD-Pg-2.19.3.tar.gz
$ cd DBD-Pg-2.19.3
$ perl Makefile.PL
$ make
$ make install
```

Before you start using Perl PostgreSQL interface, find the **pg_hba.conf** file in your PostgreSQL installation directory and add the following line –

```
# IPv4 local connections:
host    all         all         127.0.0.1/32          md5
```

You can start/restart the postgres server, in case it is not running, using the following command –

```
[root@host]# service postgresql restart
Stopping postgresql service:                               [  OK  ]
Starting postgresql service:                               [  OK  ]
```

DBI Interface APIs

Following are the important DBI routines, which can suffice your requirement to work with SQLite database from your Perl program. If you are looking for a more sophisticated application, then you can look into Perl DBI official documentation.

S. No.	API & Description
1	**DBI→connect($data_source, "userid", "password", \%attr)** Establishes a database connection, or session, to the requested $data_source. Returns a database handle object if the connection succeeds. Datasource has the form like : **DBI:Pg:dbname=$database;host=127.0.0.1;port=5432** Pg is PostgreSQL driver name and testdb is the name of database.
2	**$dbh→do($sql)** This routine prepares and executes a single SQL statement. Returns the number of rows affected or undef on error. A return value of -1 means the number of rows is not known, not applicable, or not available. Here $dbh is a handle returned by DBI→connect() call.
3	**$dbh→prepare($sql)** This routine prepares a statement for later execution by the database engine and returns a reference to a statement handle object.
4	**$sth→execute()** This routine performs whatever processing is necessary to execute the prepared statement. An undef is returned if an error occurs. A successful execute always returns true regardless of the number of rows affected. Here $sth is a statement handle returned by $dbh→prepare($sql) call.
5	**$sth→fetchrow_array()** This routine fetches the next row of data and returns it as a list containing the field values. Null fields are returned as undef values in the list.
6	**$DBI::err** This is equivalent to $h→err, where $h is any of the handle types like $dbh, $sth, or $drh. This returns native database engine error code from the last driver method called.
7	**$DBI::errstr**

	This is equivalent to $h→errstr, where $h is any of the handle types like $dbh, $sth, or $drh. This returns the native database engine error message from the last DBI method called.
8	**$dbh->disconnect()** This routine closes a database connection previously opened by a call to DBI→connect().

Connecting to Database

The following Perl code shows how to connect to an existing database. If the database does not exist, then it will be created and finally a database object will be returned.

```
#!/usr/bin/perl

use DBI;
use strict;

my $driver  = "Pg";
my $database = "testdb";
my $dsn = "DBI:$driver:dbname = $database;host = 127.0.0.1;port = 5432";
my $userid = "postgres";
my $password = "pass123";
my $dbh = DBI->connect($dsn, $userid, $password, { RaiseError => 1 })
   or die $DBI::errstr;

print "Opened database successfully\n";
```

Now, let us run the above given program to open our database **testdb**; if the database is successfully opened then it will give the following message –

```
Open database successfully
```

Create a Table

The following Perl program will be used to create a table in previously created database –

```
#!/usr/bin/perl

use DBI;
use strict;

my $driver   = "Pg";
my $database = "testdb";
my $dsn = "DBI:$driver:dbname=$database;host=127.0.0.1;port=5432";
my $userid = "postgres";
my $password = "pass123";
my $dbh = DBI->connect($dsn, $userid, $password, { RaiseError => 1 })
   or die $DBI::errstr;
print "Opened database successfully\n";

my $stmt = qq(CREATE TABLE COMPANY
```

```
      (ID INT PRIMARY KEY     NOT NULL,
      NAME           TEXT    NOT NULL,
      AGE            INT     NOT NULL,
      ADDRESS        CHAR(50),
      SALARY         REAL););
my $rv = $dbh->do($stmt);
if($rv < 0) {
   print $DBI::errstr;
} else {
   print "Table created successfully\n";
}
$dbh->disconnect();
```

When the above given program is executed, it will create COMPANY table in your **testdb** and it will display the following messages –

```
Opened database successfully
Table created successfully
```

INSERT Operation

The following Perl program shows how we can create records in our COMPANY table created in above example –

```
#!/usr/bin/perl

use DBI;
use strict;

my $driver   = "Pg";
my $database = "testdb";
my $dsn = "DBI:$driver:dbname = $database;host = 127.0.0.1;port = 5432";
my $userid = "postgres";
my $password = "pass123";
my $dbh = DBI->connect($dsn, $userid, $password, { RaiseError => 1 })
   or die $DBI::errstr;
print "Opened database successfully\n";

my $stmt = qq(INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
   VALUES (1, 'Paul', 32, 'California', 20000.00 ));
my $rv = $dbh->do($stmt) or die $DBI::errstr;

$stmt = qq(INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
   VALUES (2, 'Allen', 25, 'Texas', 15000.00 ));
$rv = $dbh->do($stmt) or die $DBI::errstr;

$stmt = qq(INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
   VALUES (3, 'Teddy', 23, 'Norway', 20000.00 ));
$rv = $dbh->do($stmt) or die $DBI::errstr;

$stmt = qq(INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
   VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00 ););
$rv = $dbh->do($stmt) or die $DBI::errstr;

print "Records created successfully\n";
$dbh->disconnect();
```

When the above given program is executed, it will create given records in COMPANY table and will display the following two lines –

```
Opened database successfully
Records created successfully
```

SELECT Operation

The following Perl program shows how we can fetch and display records from our COMPANY table created in above example –

```
#!/usr/bin/perl

use DBI;
use strict;

my $driver   = "Pg";
my $database = "testdb";
my $dsn = "DBI:$driver:dbname = $database;host = 127.0.0.1;port = 5432";
my $userid = "postgres";
my $password = "pass123";
my $dbh = DBI->connect($dsn, $userid, $password, { RaiseError => 1 })
   or die $DBI::errstr;
print "Opened database successfully\n";

my $stmt = qq(SELECT id, name, address, salary  from COMPANY;);
my $sth = $dbh->prepare( $stmt );
my $rv = $sth->execute() or die $DBI::errstr;
if($rv < 0) {
   print $DBI::errstr;
}
while(my @row = $sth->fetchrow_array()) {
      print "ID = ". $row[0] . "\n";
      print "NAME = ". $row[1] ."\n";
      print "ADDRESS = ". $row[2] ."\n";
      print "SALARY =  ". $row[3] ."\n\n";
}
print "Operation done successfully\n";
$dbh->disconnect();
```

When the above given program is executed, it will produce the following result –

```
Opened database successfully
ID = 1
NAME = Paul
ADDRESS = California
SALARY =  20000

ID = 2
NAME = Allen
ADDRESS = Texas
SALARY =  15000

ID = 3
NAME = Teddy
ADDRESS = Norway
SALARY =  20000
```

```
ID = 4
NAME = Mark
ADDRESS = Rich-Mond
SALARY =  65000


Operation done successfully
```

UPDATE Operation

The following Perl code shows how we can use the UPDATE statement to update any record and then fetch and display updated records from our COMPANY table –

```
#!/usr/bin/perl

use DBI;
use strict;

my $driver   = "Pg";
my $database = "testdb";
my $dsn = "DBI:$driver:dbname = $database;host = 127.0.0.1;port = 5432";
my $userid = "postgres";
my $password = "pass123";
my $dbh = DBI->connect($dsn, $userid, $password, { RaiseError => 1 })
   or die $DBI::errstr;
print "Opened database successfully\n";

my $stmt = qq(UPDATE COMPANY set SALARY = 25000.00 where ID=1;);
my $rv = $dbh->do($stmt) or die $DBI::errstr;
if( $rv < 0 ) {
   print $DBI::errstr;
}else{
   print "Total number of rows updated : $rv\n";
}
$stmt = qq(SELECT id, name, address, salary  from COMPANY;);
my $sth = $dbh->prepare( $stmt );
$rv = $sth->execute() or die $DBI::errstr;
if($rv < 0) {
   print $DBI::errstr;
}
while(my @row = $sth->fetchrow_array()) {
      print "ID = ". $row[0] . "\n";
      print "NAME = ". $row[1] ."\n";
      print "ADDRESS = ". $row[2] ."\n";
      print "SALARY =  ". $row[3] ."\n\n";
}
print "Operation done successfully\n";
$dbh->disconnect();
```

When the above given program is executed, it will produce the following result –

```
Opened database successfully
Total number of rows updated : 1
ID = 1
NAME = Paul
ADDRESS = California
SALARY =  25000
```

```
ID = 2
NAME = Allen
ADDRESS = Texas
SALARY =  15000

ID = 3
NAME = Teddy
ADDRESS = Norway
SALARY =  20000

ID = 4
NAME = Mark
ADDRESS = Rich-Mond
SALARY =  65000

Operation done successfully
```

DELETE Operation

The following Perl code shows how we can use the DELETE statement to delete any record and then fetch and display the remaining records from our COMPANY table –

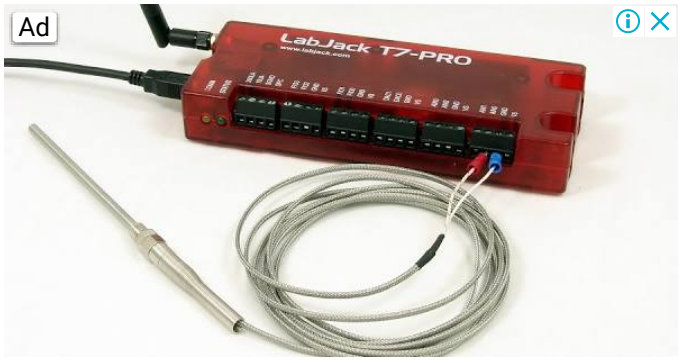
```
#!/usr/bin/perl

use DBI;
use strict;

my $driver   = "Pg";
my $database = "testdb";
my $dsn = "DBI:$driver:dbname = $database;host = 127.0.0.1;port = 5432";
my $userid = "postgres";
my $password = "pass123";
my $dbh = DBI->connect($dsn, $userid, $password, { RaiseError => 1 })
   or die $DBI::errstr;
print "Opened database successfully\n";

my $stmt = qq(DELETE from COMPANY where ID=2;);
my $rv = $dbh->do($stmt) or die $DBI::errstr;
if( $rv < 0 ) {
   print $DBI::errstr;
} else{
   print "Total number of rows deleted : $rv\n";
}
$stmt = qq(SELECT id, name, address, salary  from COMPANY;);
my $sth = $dbh->prepare( $stmt );
$rv = $sth->execute() or die $DBI::errstr;
if($rv < 0) {
   print $DBI::errstr;
}
while(my @row = $sth->fetchrow_array()) {
      print "ID = ". $row[0] . "\n";
      print "NAME = ". $row[1] ."\n";
      print "ADDRESS = ". $row[2] ."\n";
      print "SALARY =  ". $row[3] ."\n\n";
}
print "Operation done successfully\n";
$dbh->disconnect();
```

When the above given program is executed, it will produce the following result –

```
Opened database successfully
Total number of rows deleted : 1
ID = 1
NAME = Paul
ADDRESS = California
SALARY =  25000

ID = 3
NAME = Teddy
ADDRESS = Norway
SALARY =  20000

ID = 4
NAME = Mark
ADDRESS = Rich-Mond
SALARY =  65000

Operation done successfully
```

Previous Page Next Page

Advertisements



Thermocouple DAQ Devices

LabJack VISIT SITE

Tutorials Point (India) Pvt. Ltd.

YouTube 58K

Write for us FAQ's Helping Contact
© Copyright 2018. All Rights Reserved.

Enter email for newsletter go