POSTGRESQL - PHP INTERFACE

Installation

The PostgreSQL extension is enabled by default in the latest releases of PHP 5.3.x. It is possible to disable it by using **--without-pgsql** at compile time. Still you can use yum command to install PHP -PostgreSQL interface –

```
yum install php-pgsql
```

Before you start using the PHP 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  ]
```

Windows users must enable php_pgsql.dll in order to use this extension. This DLL is included with Windows distributions in the latest releases of PHP 5.3.x

For detailed installation instructions, kindly check our PHP tutorial and its official website.

PHP Interface APIs

The following are important PHP routines, which can suffice your requirement to work with PostgreSQL database from your PHP program. If you are looking for a more sophisticated application, then you can look into the PHP official documentation.

S. No.	API & Description
1	**resource pg_connect** $string\$connection_string[, int\$connect_type]$ This opens a connection to a PostgreSQL database specified by the connection_string. If PGSQL_CONNECT_FORCE_NEW is passed as connect_type, then a new connection is created in case of a second call to pg_connect, even if the connection_string is identical to an existing connection.
2	**bool pg_connection_reset** $resource\$connection$ This routine resets the connection. It is useful for error recovery. Returns TRUE on success or FALSE on failure.

3	**int pg_connection_status** *resource$connection* This routine returns the status of the specified connection. Returns PGSQL_CONNECTION_OK or PGSQL_CONNECTION_BAD.
4	**string pg_dbname** [*resource$connection*] This routine returns the name of the database that the given PostgreSQL connection resource.
5	**resource pg_prepare** [*resource$connection*], *string$stmtname*, *string$query* This submits a request to create a prepared statement with the given parameters and waits for completion.
6	**resource pg_execute** [*resource$connection*], *string$stmtname*, *array$params* This routine sends a request to execute a prepared statement with given parameters and waits for the result.
7	**resource pg_query** [*resource$connection*], *string$query* This routine executes the query on the specified database connection.
8	**array pg_fetch_row** *resource$result*[, *int$row*] This routine fetches one row of data from the result associated with the specified result resource.
9	**array pg_fetch_all** *resource$result* This routine returns an array that contains all rows *records* in the result resource.
10	**int pg_affected_rows** *resource$result* This routine returns the number of rows affected by INSERT, UPDATE, and DELETE queries.
11	**int pg_num_rows** *resource$result*

	This routine returns the number of rows in a PostgreSQL result resource for example number of rows returned by SELECT statement.
12	**bool pg_close** [*resource$connection*] This routine closes the non-persistent connection to a PostgreSQL database associated with the given connection resource.
13	**string pg_last_error** [*resource$connection*] This routine returns the last error message for a given connection.
14	**string pg_escape_literal** [*resource$connection*], *string$data* This routine escapes a literal for insertion into a text field.
15	**string pg_escape_string** [*resource$connection*], *string$data* This routine escapes a string for querying the database.

Connecting to Database

The following PHP code shows how to connect to an existing database on a local machine and finally a database connection object will be returned.

```
<?php
   $host        = "host = 127.0.0.1";
   $port        = "port = 5432";
   $dbname      = "dbname = testdb";
   $credentials = "user = postgres password=pass123";

   $db = pg_connect( "$host $port $dbname $credentials"  );
   if(!$db) {
      echo "Error : Unable to open database\n";
   } else {
      echo "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 –

```
Opened database successfully
```

Create a Table

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

```
<?php
   $host        = "host = 127.0.0.1";
   $port        = "port = 5432";
   $dbname      = "dbname = testdb";
   $credentials = "user = postgres password=pass123";

   $db = pg_connect( "$host $port $dbname $credentials"  );
   if(!$db) {
      echo "Error : Unable to open database\n";
   } else {
      echo "Opened database successfully\n";
   }

   $sql =<<<EOF
      CREATE TABLE COMPANY
      (ID INT PRIMARY KEY     NOT NULL,
      NAME           TEXT    NOT NULL,
      AGE            INT     NOT NULL,
      ADDRESS        CHAR(50),
      SALARY         REAL);
EOF;

   $ret = pg_query($db, $sql);
   if(!$ret) {
      echo pg_last_error($db);
   } else {
      echo "Table created successfully\n";
   }
   pg_close($db);
?>
```

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 PHP program shows how we can create records in our COMPANY table created in above example –

```
<?php
   $host        = "host=127.0.0.1";
   $port        = "port=5432";
   $dbname      = "dbname = testdb";
   $credentials = "user = postgres password=pass123";

   $db = pg_connect( "$host $port $dbname $credentials"  );
   if(!$db) {
      echo "Error : Unable to open database\n";
   } else {
      echo "Opened database successfully\n";
   }

   $sql =<<<EOF
      INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
```

```
      VALUES (1, 'Paul', 32, 'California', 20000.00 );

      INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
      VALUES (2, 'Allen', 25, 'Texas', 15000.00 );

      INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
      VALUES (3, 'Teddy', 23, 'Norway', 20000.00 );

      INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
      VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00 );
EOF;

   $ret = pg_query($db, $sql);
   if(!$ret) {
      echo pg_last_error($db);
   } else {
      echo "Records created successfully\n";
   }
   pg_close($db);
?>
```

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

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

SELECT Operation

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

```
<?php
   $host        = "host = 127.0.0.1";
   $port        = "port = 5432";
   $dbname      = "dbname = testdb";
   $credentials = "user = postgres password=pass123";

   $db = pg_connect( "$host $port $dbname $credentials"  );
   if(!$db) {
      echo "Error : Unable to open database\n";
   } else {
      echo "Opened database successfully\n";
   }

   $sql =<<<EOF
      SELECT * from COMPANY;
EOF;

   $ret = pg_query($db, $sql);
   if(!$ret) {
      echo pg_last_error($db);
      exit;
   }
   while($row = pg_fetch_row($ret)) {
      echo "ID = ". $row[0] . "\n";
      echo "NAME = ". $row[1] ."\n";
      echo "ADDRESS = ". $row[2] ."\n";
      echo "SALARY =  ".$row[4] ."\n\n";
   }
```

```
   echo "Operation done successfully\n";
   pg_close($db);
?>
```

When the above given program is executed, it will produce the following result. Keep a note that fields are returned in the sequence they were used while creating table.

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

```
<?php
   $host        = "host=127.0.0.1";
   $port        = "port=5432";
   $dbname      = "dbname = testdb";
   $credentials = "user = postgres password=pass123";

   $db = pg_connect( "$host $port $dbname $credentials"  );
   if(!$db) {
      echo "Error : Unable to open database\n";
   } else {
      echo "Opened database successfully\n";
   }
   $sql =<<<EOF
      UPDATE COMPANY set SALARY = 25000.00 where ID=1;
EOF;
   $ret = pg_query($db, $sql);
   if(!$ret) {
      echo pg_last_error($db);
      exit;
   } else {
      echo "Record updated successfully\n";
   }

   $sql =<<<EOF
      SELECT * from COMPANY;
```

```
EOF;

   $ret = pg_query($db, $sql);
   if(!$ret) {
      echo pg_last_error($db);
      exit;
   }
   while($row = pg_fetch_row($ret)) {
      echo "ID = ". $row[0] . "\n";
      echo "NAME = ". $row[1] ."\n";
      echo "ADDRESS = ". $row[2] ."\n";
      echo "SALARY =  ".$row[4] ."\n\n";
   }
   echo "Operation done successfully\n";
   pg_close($db);
?>
```

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

```
Opened database successfully
Record updated successfully
ID = 2
NAME = Allen
ADDRESS = 25
SALARY =  15000

ID = 3
NAME = Teddy
ADDRESS = 23
SALARY =  20000

ID = 4
NAME = Mark
ADDRESS = 25
SALARY =  65000

ID = 1
NAME = Paul
ADDRESS = 32
SALARY =  25000

Operation done successfully
```

DELETE Operation

The following PHP 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 –

```
<?php
   $host        = "host = 127.0.0.1";
   $port        = "port = 5432";
   $dbname      = "dbname = testdb";
   $credentials = "user = postgres password=pass123";

   $db = pg_connect( "$host $port $dbname $credentials"  );
   if(!$db) {
      echo "Error : Unable to open database\n";
   } else {
      echo "Opened database successfully\n";
   }
```

```
   $sql =<<<EOF
      DELETE from COMPANY where ID=2;
EOF;
   $ret = pg_query($db, $sql);
   if(!$ret) {
      echo pg_last_error($db);
      exit;
   } else {
      echo "Record deleted successfully\n";
   }

   $sql =<<<EOF
      SELECT * from COMPANY;
EOF;

   $ret = pg_query($db, $sql);
   if(!$ret) {
      echo pg_last_error($db);
      exit;
   }
   while($row = pg_fetch_row($ret)) {
      echo "ID = ". $row[0] . "\n";
      echo "NAME = ". $row[1] ."\n";
      echo "ADDRESS = ". $row[2] ."\n";
      echo "SALARY =  ".$row[4] ."\n\n";
   }
   echo "Operation done successfully\n";
   pg_close($db);
?>
```

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

```
Opened database successfully
Record deleted successfully
ID = 3
NAME = Teddy
ADDRESS = 23
SALARY =  20000

ID = 4
NAME = Mark
ADDRESS = 25
SALARY =  65000

ID = 1
NAME = Paul
ADDRESS = 32
SALARY =  25000

Operation done successfully
```