Working with PHP & Postgresql

pg_connect (string *$connection_string* [, int *$connect_type*])

pg_connect() opens a connection to a PostgreSQL database specified by connection_string

If a second call is made to pg_connect() with the same connection_string as an existing connection, the existing connection will be returned unless you pass PGSQL_CONNECT_FORCE_NEW as connect_type.

Example #1 Using pg_connect()

```
<?php
$dbconn = pg_connect("dbname=mary");
//connect to a database named "mary"

$dbconn2 = pg_connect("host=localhost port=5432 dbname=mary");
// connect to a database named "mary" on "localhost" at port "5432"

$dbconn3 = pg_connect("host=sheep port=5432 dbname=mary user=lamb password=foo");
//connect to a database named "mary" on the host "sheep" with a username and password

$conn_string = "host=sheep port=5432 dbname=test user=lamb password=bar";
$dbconn4 = pg_connect($conn_string);
//connect to a database named "test" on the host "sheep" with a username and password
?>
```

- pg_pconnect() - Open a persistent PostgreSQL connection
- pg_close() - Closes a PostgreSQL connection
- pg_host() - Returns the host name associated with the connection
- pg_port() - Return the port number associated with the connection
- pg_tty() - Return the TTY name associated with the connection
- pg_options() - Get the options associated with the connection
- pg_dbname() - Get the database name

pg_query ([resource *$connection*], string *$query*)
pg_query() executes the *query* on the specified database *connection*.
Return Values
A query result resource on success or **FALSE** on failure.
Example #1 pg_query() example

```
<?php
    $conn = pg_pconnect("dbname=publisher");
    if (!$conn) {
      echo "An error occured.\n";
      exit;
    }
    $result = pg_query($conn, "SELECT author, email FROM authors");
    if (!$result) {
      echo "An error occured.\n";
      exit;
    }
    while ($row = pg_fetch_row($result)) {
      echo "Author: $row[0]  E-mail: $row[1]";
      echo "<br />\n";
    }?>
```

Example #2 Using pg_query() with multiple statements

```
<?php

    $conn = pg_pconnect("dbname=publisher");

    // these statements will be executed as one transaction

    $query = "UPDATE authors SET author=UPPER(author) WHERE id=1;";

    $query .= "UPDATE authors SET author=LOWER(author) WHERE id=2;";

    $query .= "UPDATE authors SET author=NULL WHERE id=3;";

    pg_query($conn, $query);?>
```

- pg_fetch_array() - Fetch a row as an array
- pg_fetch_object() - Fetch a row as an object
- pg_num_rows() - Returns the number of rows in a result
- pg_affected_rows() - Returns number of affected records (tuples

array **pg_fetch_row** (resource *$result* [, int *$row*])

pg_fetch_row() fetches one row of data from the result associated with the specified *result* resource.

Parameters

result

PostgreSQL query result resource, returned by pg_query(), pg_query_params() or pg_execute() (among others).

row

Row number in result to fetch. Rows are numbered from 0 upwards. If omitted, next row is fetched.

Example #1 pg_fetch_row() example

```
<?php
    $conn = pg_pconnect("dbname=publisher");
    if (!$conn) {
      echo "An error occured.\n";
      exit;
    }
    $result = pg_query($conn, "SELECT author, email FROM authors");
    if (!$result) {
      echo "An error occured.\n";
      exit;
    }
    while ($row = pg_fetch_row($result)) {
      echo "Author: $row[0]  E-mail: $row[1]";
      echo "<br />\n";
    }
    ?>
```

- pg_query() - Execute a query
- pg_fetch_array() - Fetch a row as an array
- pg_fetch_object() - Fetch a row as an object
- pg_fetch_result() - Returns values from a result resource

array **pg_fetch_array** (resource *$result* [, int *$row* [, int *$result_type*]])

pg_fetch_array() returns an array that corresponds to the fetched row (record).

Parameters

result PostgreSQL query result resource, returned by pg_query(), pg_query_params() or pg_execute() (among others).

row Row number in result to fetch. Rows are numbered from 0 upwards. If omitted, next row is fetched.

Return Values An array indexed numerically (beginning with 0) or associatively (indexed by field name), or both. Each value in the array is represented as a string .

Example #1 pg_fetch_array() example

```
<?php
    $conn = pg_pconnect("dbname=publisher");

    if (!$conn) {
      echo "An error occured.\n";
      exit;
    }

    $result = pg_query($conn, "SELECT author, email FROM authors");

    if (!$result) {
      echo "An error occured.\n";
      exit;
    }

    $arr = pg_fetch_array($result, 0, PGSQL_NUM);

    echo $arr[0] . " <- Row 1 Author\n";

    echo $arr[1] . " <- Row 1 E-mail\n";

    // As of PHP 4.1.0, the row parameter is optional; NULL can be passed instead,
    // to pass a result_type.  Successive calls to pg_fetch_array will return the
    // next row.

    $arr = pg_fetch_array($result, NULL, PGSQL_ASSOC);

    echo $arr["author"] . " <- Row 2 Author\n";

    echo $arr["email"] . " <- Row 2 E-mail\n";

    $arr = pg_fetch_array($result);

    echo $arr["author"] . " <- Row 3 Author\n";

    echo $arr[1] . " <- Row 3 E-mail\n";

?>
```

array **pg_fetch_all** (resource *$result*)

pg_fetch_all() returns an array that contains all rows (records) in result resource.

PostgreSQL query result resource, returned by pg_query(), pg_query_params() or pg_execute() (among others).

Example #1 PostgreSQL fetch all

```
<?php
    $conn = pg_pconnect("dbname=publisher");
    if (!$conn) {
        echo "An error occured.\n";
        exit;
    }
    $result = pg_query($conn, "SELECT * FROM authors");
    if (!$result) {
        echo "An error occured.\n";
        exit;
    }
    $arr = pg_fetch_all($result);

    print_r($arr);
    ?>
```

The above example will output something similar to:

```
Array (
    [0] =>
            Array ( [id] => 1 [name] => Fred )
    [1] =>
            Array ( [id] => 2 [name] => Bob ) )
```

- pg_fetch_row() - Get a row as an enumerated array
- pg_fetch_array() - Fetch a row as an array
- pg_fetch_object() - Fetch a row as an object
- pg_fetch_result() - Returns values from a result resource

string **pg_last_error** ([resource *$connection*])
pg_last_error() returns the last error message for a given *connection*.

Parameters

connection

PostgreSQL database connection resource. When *connection* is not present, the default connection is used. The default connection is the last connection made by pg_connect() or pg_pconnect().

Return Values

A string containing the last error message on the given *connection*, or **FALSE** on error.

Example #1 pg_last_error() example

```
<?php
    $dbconn = pg_connect("dbname=publisher") or die("Could not connect");

    // Query that fails
    $res = pg_query($dbconn, "select * from doesnotexist");

    echo pg_last_error($dbconn);
?>
```

- pg_result_error() - Get error message associated with result
- pg_result_error_field() - Returns an individual field of an error report.

pg_execute ([resource *$connection*], string *$stmtname* , array *$params*)

Sends a request to execute a prepared statement with given parameters, and waits for the result.

pg_execute() is like pg_query_params(), but the command to be executed is specified by naming a previously-prepared statement, instead of giving a query string. This feature allows commands that will be used repeatedly to be parsed and planned just once, rather than each time they are executed. The statement must have been prepared previously in the current session.

Example #1 Using pg_execute()

```
<?php
// Connect to a database named "mary“
$dbconn = pg_connect("dbname=mary");
// Prepare a query for execution
$result = pg_prepare($dbconn, "my_query", 'SELECT * FROM shops WHERE name = $1');
// Execute the prepared query.  Note that it is not necessary to escape
// the string "Joe's Widgets" in any way
$result = pg_execute($dbconn, "my_query", array("Joe's Widgets"));
// Execute the same prepared query, this time with a different parameter
$result = pg_execute($dbconn, "my_query", array("Clothes Clothes Clothes"));
?>
```

- pg_prepare() - Submits a request to create a prepared statement with the given parameters, and waits for completion.
- pg_send_prepare() - Sends a request to create a prepared statement with the given parameters, without waiting for completion.
- pg_query_params() - Submits a command to the server and waits for the result, with the ability to pass parameters separately from the SQL command text.