MySQL Improved Extension

Introduction

The mysqli extension allows you to access the functionality provided by MySQL 4.1 and above. More information about the MySQL Database server can be found at » http://www.mysql.com/

Documentation for MySQL can be found at » http://dev.mysql.com/doc/.

Parts of this documentation included from MySQL manual with permissions of MySQL AB.

Examples

All Examples in the MySQLI documentation use the world database from MySQL AB. The world database can be found at » http://dev.mysql.com/get/Downloads/Manual/world.sql.gz/from/pick

Installing/Configuring

Requirements

In order to have these functions available, you must compile PHP with support for the mysqli extension.

Note
The mysqli extension is designed to work with the version 4.1.3 or above of MySQL. For previous versions, please see the MySQL extension documentation.

Installation

To install the mysqli extension for PHP, use the *--with-mysqli=mysql_config_path/mysql_config* configuration option where *mysql_config_path* represents the location of the *mysql_config* program that comes with MySQL versions greater than 4.1.

If you would like to install the mysql extension along with the mysqli extension you have to use the same client library to avoid any conflicts.

Installation on Windows Systems

MySQLi is not enabled by default, so the *php_mysqli.dll* DLL must be enabled inside of *php.ini*. Also, PHP needs access to the MySQL client library. A file named *libmysql.dll* is included in the Windows PHP distribution and in order for PHP to talk to MySQL this file needs to be available to the Windows systems PATH. See the FAQ titled " How do I add my PHP directory to the PATH on Windows " for information on how to do this. Although copying *libmysql.dll* to the Windows system directory also works (because the system directory is by default in the system's PATH), it's not recommended.

As with enabling any PHP extension (such as *php_mysqli.dll*), the PHP directive extension_dir should be set to the directory where the PHP extensions are located. See also the Manual Windows Installation Instructions. An example extension_dir value for PHP 5 is *c:\php\ext*

Note
If when starting the web server an error similar to the following occurs: *"Unable to load dynamic library './php_mysqli.dll'"*, this is because *php_mysqli.dll* and/or *libmysql.dll* cannot be found by the system.

Runtime Configuration

The behaviour of these functions is affected by settings in *php.ini*.

MySQLi Configuration Options

Name	Default	Changeable	Changelog
mysqli.max_links	"-1"	PHP_INI_SYSTEM	Available since PHP 5.0.0.
mysqli.default_port	"3306"	PHP_INI_ALL	Available since PHP 5.0.0.
mysqli.default_socket	NULL	PHP_INI_ALL	Available since PHP 5.0.0.
mysqli.default_host	NULL	PHP_INI_ALL	Available since PHP 5.0.0.
mysqli.default_user	NULL	PHP_INI_ALL	Available since PHP 5.0.0.
mysqli.default_pw	NULL	PHP_INI_ALL	Available since PHP 5.0.0.

For further details and definitions of the above PHP_INI_* constants, see the chapter on configuration changes.

Here's a short explanation of the configuration directives.

`mysqli.max_links` integer
: The maximum number of MySQL connections per process.

`mysqli.default_port` string
: The default TCP port number to use when connecting to the database server if no other port is specified. If no default is specified, the port will be obtained from the *MYSQL_TCP_PORT* environment variable, the *mysql-tcp* entry in */etc/services* or the compile-time *MYSQL_PORT* constant, in that order. Win32 will only use the *MYSQL_PORT* constant.

`mysqli.default_socket` string
: The default socket name to use when connecting to a local database server if no other socket name is specified.

`mysqli.default_host` string

The default server host to use when connecting to the database server if no other host is specified. Doesn't apply in safe mode.

mysqli.default_user string
The default user name to use when connecting to the database server if no other name is specified. Doesn't apply in safe mode.

mysqli.default_pw string
The default password to use when connecting to the database server if no other password is specified. Doesn't apply in safe mode.

Resource Types

This extension has no resource types defined.

Predefined Constants

MYSQLI_READ_DEFAULT_GROUP
Read options from the named group from *my.cnf* or the file specified with **MYSQLI_READ_DEFAULT_FILE**

MYSQLI_READ_DEFAULT_FILE
Read options from the named option file instead of from *my.cnf*

MYSQLI_OPT_CONNECT_TIMEOUT
Connect timeout in seconds

MYSQLI_OPT_LOCAL_INFILE
Enables command *LOAD LOCAL INFILE*

MYSQLI_INIT_COMMAND
Command to execute when connecting to MySQL server. Will automatically be re-executed when reconnecting.

MYSQLI_CLIENT_SSL
Use SSL (encrypted protocol). This option should not be set by application programs; it is set internally in the MySQL client library

MYSQLI_CLIENT_COMPRESS
Use compression protocol

MYSQLI_CLIENT_INTERACTIVE
Allow interactive_timeout seconds (instead of wait_timeout seconds) of inactivity before closing the connection. The client's session wait_timeout variable will be set to the value of the session interactive_timeout variable.

MYSQLI_CLIENT_IGNORE_SPACE
Allow spaces after function names. Makes all functions names reserved words.

MYSQLI_CLIENT_NO_SCHEMA
Don't allow the *db_name.tbl_name.col_name* syntax.

MYSQLI_CLIENT_MULTI_QUERIES
Allows multiple semicolon-delimited queries in a single mysqli_query() call.

MYSQLI_STORE_RESULT
For using buffered resultsets

MYSQLI_USE_RESULT
For using unbuffered resultsets

MYSQLI_ASSOC
Columns are returned into the array having the fieldname as the array index.

MYSQLI_NUM
Columns are returned into the array having an enumerated index.

MYSQLI_BOTH
Columns are returned into the array having both a numerical index and the fieldname as the associative index.

MYSQLI_NOT_NULL_FLAG
Indicates that a field is defined as *NOT NULL*

MYSQLI_PRI_KEY_FLAG
Field is part of a primary index

MYSQLI_UNIQUE_KEY_FLAG
Field is part of a unique index.

MYSQLI_MULTIPLE_KEY_FLAG
Field is part of an index.

MYSQLI_BLOB_FLAG
Field is defined as *BLOB*

MYSQLI_UNSIGNED_FLAG
Field is defined as *UNSIGNED*

MYSQLI_ZEROFILL_FLAG
Field is defined as *ZEROFILL*

MYSQLI_AUTO_INCREMENT_FLAG
Field is defined as *AUTO_INCREMENT*

MYSQLI_TIMESTAMP_FLAG
Field is defined as *TIMESTAMP*

MYSQLI_SET_FLAG
Field is defined as *SET*

MYSQLI_NUM_FLAG
Field is defined as *NUMERIC*

MYSQLI_PART_KEY_FLAG
Field is part of an multi-index

MYSQLI_GROUP_FLAG
Field is part of *GROUP BY*

MYSQLI_TYPE_DECIMAL
Field is defined as *DECIMAL*

MYSQLI_TYPE_NEWDECIMAL
Precision math *DECIMAL* or *NUMERIC* field (MySQL 5.0.3 and up)

MYSQLI_TYPE_BIT
Field is defined as *BIT* (MySQL 5.0.3 and up)

MYSQLI_TYPE_TINY
Field is defined as *TINYINT*

MYSQLI_TYPE_SHORT
Field is defined as *INT*

MYSQLI_TYPE_LONG
Field is defined as *INT*

MYSQLI_TYPE_FLOAT
Field is defined as *FLOAT*

MYSQLI_TYPE_DOUBLE
Field is defined as *DOUBLE*

MYSQLI_TYPE_NULL
Field is defined as *DEFAULT NULL*

MYSQLI_TYPE_TIMESTAMP
Field is defined as *TIMESTAMP*

MYSQLI_TYPE_LONGLONG
Field is defined as *BIGINT*

MYSQLI_TYPE_INT24
Field is defined as *MEDIUMINT*

MYSQLI_TYPE_DATE
Field is defined as *DATE*

MYSQLI_TYPE_TIME
Field is defined as *TIME*

MYSQLI_TYPE_DATETIME
Field is defined as *DATETIME*

MYSQLI_TYPE_YEAR
Field is defined as *YEAR*

MYSQLI_TYPE_NEWDATE
Field is defined as *DATE*

MYSQLI_TYPE_ENUM
Field is defined as *ENUM*

MYSQLI_TYPE_SET
Field is defined as *SET*

MYSQLI_TYPE_TINY_BLOB

Field is defined as *TINYBLOB*

MYSQLI_TYPE_MEDIUM_BLOB
Field is defined as *MEDIUMBLOB*

MYSQLI_TYPE_LONG_BLOB
Field is defined as *LONGBLOB*

MYSQLI_TYPE_BLOB
Field is defined as *BLOB*

MYSQLI_TYPE_VAR_STRING
Field is defined as *VARCHAR*

MYSQLI_TYPE_STRING
Field is defined as *CHAR*

MYSQLI_TYPE_GEOMETRY
Field is defined as *GEOMETRY*

MYSQLI_NEED_DATA
More data available for bind variable

MYSQLI_NO_DATA
No more data available for bind variable

MYSQLI_DATA_TRUNCATED
Data truncation occurred. Available since PHP 5.1.0 and MySQL 5.0.5.

The MySQLi class

Introduction

Represents a connection between PHP and a MySQL database.

Class synopsis

MySQLi

```
MySQLi {

    /*  Properties */

    int affected_rows;

    string connect_errno;

    string connect_error;

    int errno;

    string error;

    int field_count;

    string host_info;

    string protocol_version;

    string server_info;

    int server_version;

    string info;

    int insert_id;

    string sqlstate;

    int thread_id;

    int warning_count;
```

/* Methods */

int **mysqli_affected_rows** (mysqli `$link`)

bool **mysqli::autocommit** (bool `$mode`)

bool **mysqli::change_user** (string `$user`, string `$password`, string `$database`)

string **mysqli::character_set_name** (void)

bool **mysqli::close** (void)

bool **mysqli::commit** (void)

int **mysqli_connect_errno** (void)

string **mysqli_connect_error** (void)

mysqli **mysqli_connect** ([string `$host` [, string `$username` [, string `$passwd` [, string `$dbname` [, int `$port` [, string `$socket`]]]]]])

bool **mysqli::debug** (string `$message`)

bool **mysqli::dump_debug_info** (void)

int **mysqli_errno** (mysqli `$link`)

string **mysqli_error** (mysqli `$link`)

int **mysqli_field_count** (mysqli `$link`)

object **mysqli::get_charset** (void)

string **mysqli::get_client_info** (void)

int **mysqli::get_client_version** (void)

string **mysqli_get_host_info** (mysqli `$link`)

int **mysqli_get_proto_info** (mysqli `$link`)

string **mysqli_get_server_info** (mysqli `$link`)

int **mysqli_get_server_version** (mysqli `$link`)

object **mysqli::get_warnings** (void)

string **mysqli_info** (mysqli `$link`)

mysqli **init** (void)

int **mysqli_insert_id** (mysqli `$link`)

bool **mysqli::kill** (int `$processid`)

bool **mysqli::more_results** (void)

bool **mysqli::multi_query** (string `$query`)

bool **mysqli::next_result** (void)

bool **mysqli::options** (int `$option`, mixed `$value`)

bool **mysqli::ping** (void)

mysqli_stmt **prepare** (string `$query`)

mixed **mysqli::query** (string `$query` [, int `$resultmode`])

bool **mysqli::real_connect** ([string `$host` [, string `$username` [, string `$passwd` [, string `$dbname` [, int `$port` [, string `$socket` [, int `$flags`]]]]]])

string **mysqli::escape_string** (string `$escapestr`)

bool **real_query** (string `$query`)

bool **mysqli::rollback** (void)

bool **mysqli::select_db** (string `$dbname`)

bool **mysqli::set_charset** (string `$charset`)

void **mysqli_set_local_infile_default** (mysqli `$link`)

bool **mysqli_set_local_infile_handler** (mysqli `$link`, callback `$read_func`)

string **mysqli_sqlstate** (mysqli `$link`)

bool **mysqli::ssl_set** (string `$key`, string `$cert`, string `$ca`, string `$capath`, string `$cipher`)

string **mysqli::stat** (void)

mysqli_stmt **stmt_init** (void)

mysqli_result **store_result** (void)

int **mysqli_thread_id** (mysqli `$link`)

bool **mysqli_thread_safe** (void)

mysqli_result **use_result** (void)

int **mysqli_warning_count** (mysqli `$link`)
}

mysqli->affected_rows

mysqli_affected_rows

mysqli->affected_rows -- mysqli_affected_rows -- Gets the number of affected rows in a previous MySQL operation

Description

Object oriented style (property):

mysqli

int *affected_rows*;

Procedural style:

int **mysqli_affected_rows** (mysqli `$link`)

Returns the number of rows affected by the last *INSERT*, *UPDATE*, *REPLACE* or *DELETE* query.

For SELECT statements mysqli_affected_rows() works like mysqli_num_rows().

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

An integer greater than zero indicates the number of rows affected or retrieved. Zero indicates that no records where updated for an UPDATE statement, no rows matched the *WHERE* clause in the query or that no query has yet been executed. -1 indicates that the query returned an error.

Note
If the number of affected rows is greater than maximal int value, the number of affected rows will be returned as a string.

Examples

Example #1 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* Insert rows */
$mysqli->query("CREATE TABLE Language SELECT * from CountryLanguage");
printf("Affected rows (INSERT): %d\n", $mysqli->affected_rows);

$mysqli->query("ALTER TABLE Language ADD Status int default 0");

/* update rows */
$mysqli->query("UPDATE Language SET Status=1 WHERE Percentage > 50");
printf("Affected rows (UPDATE): %d\n", $mysqli->affected_rows);

/* delete rows */
$mysqli->query("DELETE FROM Language WHERE Percentage < 50");
printf("Affected rows (DELETE): %d\n", $mysqli->affected_rows);

/* select all rows */
$result = $mysqli->query("SELECT CountryCode FROM Language");
printf("Affected rows (SELECT): %d\n", $mysqli->affected_rows);

$result->close();

/* Delete table Language */
$mysqli->query("DROP TABLE Language");

/* close connection */
$mysqli->close();
?>
```

Example #2 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

if (!$link) {
   printf("Can't connect to localhost. Error: %s\n",
mysqli_connect_error());
   exit();
}

/* Insert rows */
mysqli_query($link, "CREATE TABLE Language SELECT * from CountryLanguage");
printf("Affected rows (INSERT): %d\n", mysqli_affected_rows($link));

mysqli_query($link, "ALTER TABLE Language ADD Status int default 0");
```

```
/* update rows */
mysqli_query($link, "UPDATE Language SET Status=1 WHERE Percentage > 50");
printf("Affected rows (UPDATE): %d\n", mysqli_affected_rows($link));

/* delete rows */
mysqli_query($link, "DELETE FROM Language WHERE Percentage < 50");
printf("Affected rows (DELETE): %d\n", mysqli_affected_rows($link));

/* select all rows */
$result = mysqli_query($link, "SELECT CountryCode FROM Language");
printf("Affected rows (SELECT): %d\n", mysqli_affected_rows($link));

mysqli_free_result($result);

/* Delete table Language */
mysqli_query($link, "DROP TABLE Language");

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Affected rows (INSERT): 984
Affected rows (UPDATE): 168
Affected rows (DELETE): 815
Affected rows (SELECT): 169
```

See Also

- mysqli_num_rows()
- mysqli_info()

mysqli::autocommit

mysqli_autocommit

mysqli::autocommit -- mysqli_autocommit -- Turns on or off auto-commiting database modifications

Description

Object oriented style (method)

bool **mysqli::autocommit** (bool `$mode`)

Procedural style:

bool **mysqli_autocommit** (mysqli `$link`, bool `$mode`)

Turns on or off auto-commit mode on queries for the database connection.

To determine the current state of autocommit use the SQL command *SELECT @@autocommit*.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

`mode`
Whether to turn on auto-commit or not.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
This function doesn't work with non transactional table types (like MyISAM or ISAM).

Examples

Example #3 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* turn autocommit on */
$mysqli->autocommit(TRUE);

if ($result = $mysqli->query("SELECT @@autocommit")) {
   $row = $result->fetch_row();
   printf("Autocommit is %s\n", $row[0]);
   $result->free();
}

/* close connection */
$mysqli->close();
?>
```

Example #4 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

if (!$link) {
   printf("Can't connect to localhost. Error: %s\n",
mysqli_connect_error());
   exit();
}

/* turn autocommit on */
mysqli_autocommit($link, TRUE);

if ($result = mysqli_query($link, "SELECT @@autocommit")) {
   $row = mysqli_fetch_row($result);
   printf("Autocommit is %s\n", $row[0]);
   mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Autocommit is 1
```

See Also

- mysqli_commit()
- mysqli_rollback()

mysqli::change_user

mysqli_change_user

mysqli::change_user -- mysqli_change_user -- Changes the user of the specified database connection

Description

Object oriented style (method):

bool **mysqli::change_user** (string `$user`, string `$password`, string `$database`)

Procedural style:

bool **mysqli_change_user** (mysqli `$link`, string `$user`, string `$password`, string `$database`)

Changes the user of the specified database connection and sets the current database.

In order to successfully change users a valid *`username`* and *`password`* parameters must be provided and that user must have sufficient permissions to access the desired database. If for any reason authorization fails, the current user authentication will remain.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

`user`
The MySQL user name.

`password`
The MySQL password.

`database`
The database to change to. If desired, the **NULL** value may be passed resulting in only changing the user and not selecting a database. To select a database in this case use the mysqli_select_db() function.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note

Using this command will always cause the current database connection to behave as if was a completely new database connection, regardless of if the operation was completed successfully. This reset includes performing a rollback on any active transactions, closing all temporary tables, and unlocking all locked tables.

Examples

Example #5 - Object oriented style

```
<?php

/* connect database test */
$mysqli = new mysqli("localhost", "my_user", "my_password", "test");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* Set Variable a */
$mysqli->query("SET @a:=1");

/* reset all and select a new database */
$mysqli->change_user("my_user", "my_password", "world");

if ($result = $mysqli->query("SELECT DATABASE()")) {
   $row = $result->fetch_row();
   printf("Default database: %s\n", $row[0]);
   $result->close();
}

if ($result = $mysqli->query("SELECT @a")) {
   $row = $result->fetch_row();
   if ($row[0] === NULL) {
       printf("Value of variable a is NULL\n");
   }
   $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Example #6 - Procedural style

```
<?php
/* connect database test */
$link = mysqli_connect("localhost", "my_user", "my_password", "test");
```

```
/* check connection */
if (!$link) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* Set Variable a */
mysqli_query($link, "SET @a:=1");

/* reset all and select a new database */
mysqli_change_user($link, "my_user", "my_password", "world");

if ($result = mysqli_query($link, "SELECT DATABASE()")) {
   $row = mysqli_fetch_row($result);
   printf("Default database: %s\n", $row[0]);
   mysqli_free_result($result);
}

if ($result = mysqli_query($link, "SELECT @a")) {
   $row = mysqli_fetch_row($result);
   if ($row[0] === NULL) {
       printf("Value of variable a is NULL\n");
   }
   mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Default database: world
Value of variable a is NULL
```

See Also

- mysqli_connect()
- mysqli_select_db()

mysqli::character_set_name

mysqli_character_set_name

mysqli::character_set_name -- mysqli_character_set_name -- Returns the default character set for the database connection

Description

Object oriented style (method):

string **mysqli::character_set_name** (void)

Procedural style:

string **mysqli_character_set_name** (mysqli `$link`)

Returns the current character set for the database connection.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

The default character set for the current connection

Examples

Example #7 - Object oriented style

```
<?php
/* Open a connection */
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* Print current character set */
$charset = $mysqli->character_set_name();
printf ("Current character set is %s\n", $charset);
```

```
$mysqli->close();
?>
```

Example #8 - Procedural style

```
<?php
/* Open a connection */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (!$link) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* Print current character set */
$charset = mysqli_character_set_name($link);
printf ("Current character set is %s\n",$charset);

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Current character set is latin1_swedish_ci
```

See Also

- mysqli_client_encoding()
- mysqli_real_escape_string()

mysqli::close

mysqli_close

mysqli::close -- mysqli_close -- Closes a previously opened database connection

Description

Object oriented style (method):

bool **mysqli::close** (void)

Procedural style:

bool **mysqli_close** (mysqli `$link`)

Closes a previously opened database connection.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- mysqli_connect()
- mysqli_init()
- mysqli_real_connect()

mysqli::commit

mysqli_commit

mysqli::commit -- mysqli_commit -- Commits the current transaction

Description

Object oriented style (method)

bool **mysqli::commit** (void)

Procedural style:

bool **mysqli_commit** (mysqli `$link`)

Commits the current transaction for the database connection.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #9 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$mysqli->query("CREATE TABLE Language LIKE CountryLanguage Type=InnoDB");

/* set autocommit to off */
$mysqli->autocommit(FALSE);
```

```
/* Insert some values */
$mysqli->query("INSERT INTO Language VALUES ('DEU', 'Bavarian', 'F',
11.2)");
$mysqli->query("INSERT INTO Language VALUES ('DEU', 'Swabian', 'F', 9.4)");

/* commit transaction */
$mysqli->commit();

/* drop table */
$mysqli->query("DROP TABLE Language");

/* close connection */
$mysqli->close();
?>
```

Example #10 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "test");

/* check connection */
if (!$link) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* set autocommit to off */
mysqli_autocommit($link, FALSE);

mysqli_query($link, "CREATE TABLE Language LIKE CountryLanguage
Type=InnoDB");

/* Insert some values */
mysqli_query($link, "INSERT INTO Language VALUES ('DEU', 'Bavarian', 'F',
11.2)");
mysqli_query($link, "INSERT INTO Language VALUES ('DEU', 'Swabian', 'F',
9.4)");

/* commit transaction */
mysqli_commit($link);

/* close connection */
mysqli_close($link);
?>
```

See Also

- mysqli_autocommit()
- mysqli_rollback()

mysqli->connect_errno

mysqli_connect_errno

mysqli->connect_errno -- mysqli_connect_errno -- Returns the error code from last connect call

Description

mysqli

string *connect_errno*;

int **mysqli_connect_errno** (void)

Returns the last error code number from the last call to mysqli_connect().

Note

Client error message numbers are listed in the MySQL *errmsg.h* header file, server error message numbers are listed in *mysqld_error.h*. In the MySQL source distribution you can find a complete list of error messages and error numbers in the file *Docs/mysqld_error.txt*.

Return Values

An error code value for the last call to mysqli_connect(), if it failed. zero means no error occurred.

Examples

Example #11 - mysqli_connect_errno() example

```
<?php

$link = @mysqli_connect("localhost", "nonexisting_user", "");

if (!$link) {
   printf("Can't connect to localhost. Errorcode: %d\n",
mysqli_connect_errno());
}
```

```
?>
```

See Also

- mysqli_connect()
- mysqli_connect_error()
- mysqli_errno()
- mysqli_error()
- mysqli_sqlstate()

mysqli->connect_error

mysqli_connect_error

mysqli->connect_error -- mysqli_connect_error -- Returns a string description of the last connect error

Description

mysqli

string *connect_error*;

string **mysqli_connect_error** (void)

Returns the last error message string from the last call to mysqli_connect().

Return Values

A string that describes the error. An empty string if no error occurred.

Examples

Example #12 - mysqli_connect_error() example

```
<?php

$link = @mysqli_connect("localhost", "nonexisting_user", "");

if (!$link) {
   printf("Can't connect to localhost. Error: %s\n",
mysqli_connect_error());
}
?>
```

See Also

- mysqli_connect()
- mysqli_connect_errno()

- mysqli_errno()
- mysqli_error()
- mysqli_sqlstate()

mysqli::__construct

mysqli_connect

mysqli::__construct -- mysqli_connect -- Open a new connection to the MySQL server

Description

Object oriented style (constructor):

mysqli::__construct ([string `$host` [, string `$username` [, string `$passwd` [, string `$dbname` [, int `$port` [, string `$socket`]]]]])

Procedural style

mysqli **mysqli_connect** ([string `$host` [, string `$username` [, string `$passwd` [, string `$dbname` [, int `$port` [, string `$socket`]]]]])

Opens a connection to the MySQL Server running on.

Parameters

`host`
: Can be either a host name or an IP address. Passing the **NULL** value or the string "localhost" to this parameter, the local host is assumed. When possible, pipes will be used instead of the TCP/IP protocol.

`username`
: The MySQL user name.

`passwd`
: If not provided or **NULL**, the MySQL server will attempt to authenticate the user against those user records which have no password only. This allows one username to be used with different permissions (depending on if a password as provided or not).

`dbname`
: If provided will specify the default database to be used when performing queries.

`port`
: Specifies the port number to attempt to connect to the MySQL server.

`socket`
: Specifies the socket or named pipe that should be used.

Note
Specifying the *`socket`* parameter will not explicitly determine the type of connection

to be used when connecting to the MySQL server. How the connection is made to the MySQL database is determined by the *host* parameter.

Return Values

Returns a object which represents the connection to a MySQL Server or **FALSE** if the connection failed.

Examples

Example #13 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

printf("Host information: %s\n", $mysqli->host_info);

/* close connection */
$mysqli->close();
?>
```

Example #14 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (!$link) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

printf("Host information: %s\n", mysqli_get_host_info($link));

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Host information: Localhost via UNIX socket
```

Notes

Note

Error "Can't create TCP/IP socket (10106)" usually means that the variables_order configure directive doesn't contain character *E*. On Windows, if the environment is not copied the *SYSTEMROOT* environment variable won't be available and PHP will have problems loading Winsock.

mysqli::debug

mysqli_debug

mysqli::debug -- mysqli_debug -- Performs debugging operations

Description

Object oriented style (method):

bool **mysqli::debug** (string `$message`)

Procedural style:

bool **mysqli_debug** (string `$message`)

Performs debugging operations using the Fred Fish debugging library.

Parameters

`message`
A string representing the debugging operation to perform

Return Values

Returns **TRUE**.

Notes

Note
To use the mysqli_debug() function you must complile the MySQL client library to support debugging.

Examples

Example #15 - Generating a Trace File

```
<?php

/* Create a trace file in '/tmp/client.trace' on the local (client) machine:
*/
```

```
mysqli_debug("d:t:0,/tmp/client.trace");

?>
```

See Also

- mysqli_dump_debug_info()
- mysqli_report()

mysqli::dump_debug_info

mysqli_dump_debug_info

mysqli::dump_debug_info -- mysqli_dump_debug_info -- Dump debugging information into the log

Description

Object oriented style (method):

bool **mysqli::dump_debug_info** (void)

Procedural style:

bool **mysqli_dump_debug_info** (mysqli `$link`)

This function is designed to be executed by an user with the SUPER privilege and is used to dump debugging information into the log for the MySQL Server relating to the connection.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- mysqli_debug()

mysqli->errno

mysqli_errno

mysqli->errno -- mysqli_errno -- Returns the error code for the most recent function call

Description

Object oriented style (property):

mysqli

int *errno*;

Procedural style:

int **mysqli_errno** (mysqli `$link`)

Returns the last error code for the most recent MySQLi function call that can succeed or fail.

Client error message numbers are listed in the MySQL *errmsg.h* header file, server error message numbers are listed in *mysqld_error.h*. In the MySQL source distribution you can find a complete list of error messages and error numbers in the file *Docs/mysqld_error.txt*.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

An error code value for the last call, if it failed. zero means no error occurred.

Examples

Example #16 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");
```

```
/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

if (!$mysqli->query("SET a=1")) {
   printf("Errorcode: %d\n", $mysqli->errno);
}

/* close connection */
$mysqli->close();
?>
```

Example #17 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

if (!mysqli_query($link, "SET a=1")) {
   printf("Errorcode: %d\n", mysqli_errno($link));
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Errorcode: 1193
```

See Also

- mysqli_connect_errno()
- mysqli_connect_error()
- mysqli_error()
- mysqli_sqlstate()

mysqli->error

mysqli_error

mysqli->error -- mysqli_error -- Returns a string description of the last error

Description

Object oriented style (property):

mysqli

string *error*;

Procedural style:

string **mysqli_error** (mysqli `$link`)

Returns the last error message for the most recent MySQLi function call that can succeed or fail.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

A string that describes the error. An empty string if no error occurred.

Examples

Example #18 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
```

```
}

if (!$mysqli->query("SET a=1")) {
   printf("Errormessage: %s\n", $mysqli->error);
}

/* close connection */
$mysqli->close();
?>
```

Example #19 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

if (!mysqli_query($link, "SET a=1")) {
   printf("Errormessage: %s\n", mysqli_error($link));
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Errormessage: Unknown system variable 'a'
```

See Also

- mysqli_connect_errno()
- mysqli_connect_error()
- mysqli_errno()
- mysqli_sqlstate()

mysqli->field_count

mysqli_field_count

mysqli->field_count -- mysqli_field_count -- Returns the number of columns for the most recent query

Description

Object oriented style (property):

mysqli_result

int *field_count*;

Procedural style:

int **mysqli_field_count** (mysqli `$link`)

Returns the number of columns for the most recent query on the connection represented by the *`link`* parameter. This function can be useful when using the mysqli_store_result() function to determine if the query should have produced a non-empty result set or not without knowing the nature of the query.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

An integer representing the number of fields in a result set.

Examples

Example #20 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "test");
```

```
$mysqli->query( "DROP TABLE IF EXISTS friends");
$mysqli->query( "CREATE TABLE friends (id int, name varchar(20))");

$mysqli->query( "INSERT INTO friends VALUES (1,'Hartmut'), (2, 'Ulf')");


$mysqli->real_query($HTTP_POST_VARS['query']);

if ($mysqli->field_count) {
   /* this was a select/show or describe query */
   $result = $mysqli->store_result();

   /* process resultset */
   $row = $result->fetch_row();

   /* free resultset */
   $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Example #21 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "test");

mysqli_query($link, "DROP TABLE IF EXISTS friends");
mysqli_query($link, "CREATE TABLE friends (id int, name varchar(20))");

mysqli_query($link, "INSERT INTO friends VALUES (1,'Hartmut'), (2, 'Ulf')");

mysqli_real_query($link, $HTTP_POST_VARS['query']);

if (mysqli_field_count($link)) {
   /* this was a select/show or describe query */
   $result = mysqli_store_result($link);

   /* process resultset */
   $row = mysqli_fetch_row($result);

   /* free resultset */
   mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

mysqli::get_charset

mysqli_get_charset

mysqli::get_charset -- mysqli_get_charset -- Returns a character set object

Description

object **mysqli::get_charset** (void)

object **mysqli_get_charset** (mysqli `$link`)

Returns a character set object providing several properties of the current active characer set.

Parameters

`link`
: Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

The function returns a character set object with the following properties:

`charset`
: Character set name

`collation`
: Collation name

`dir`
: Directory the charset description was fetched from (?) or "" for builtin character sets

`min_length`
: Minimum character lenght in bytes

`max_length`
: Maximum character length in bytes

`number`
: Internal character set number

`state`
: Characer set status (?)

Examples

Example #22 - Object oriented style

```
<?php
 $db = mysqli_init();
 $db->real_connect("localhost","root","","test");
 var_dump($db->get_charset());
?>
```

Example #23 - Procedural style

```
<?php
 $db = mysqli_init();
 mysqli_real_connect($db, "localhost","root","","test");
 var_dump($db->get_charset());
?>
```

The above example will output:

```
object(stdClass)#2 (7) {
 ["charset"]=>
 string(6) "latin1"
 ["collation"]=>
 string(17) "latin1_swedish_ci"
 ["dir"]=>
 string(0) ""
 ["min_length"]=>
 int(1)
 ["max_length"]=>
 int(1)
 ["number"]=>
 int(8)
 ["state"]=>
 int(801)
}
```

See Also

- **mysqli_characters_set_name()**
- mysqli_set_charset()

mysqli::get_client_info

mysqli_get_client_info

mysqli::get_client_info -- mysqli_get_client_info -- Returns the MySQL client version as a string

Description

string **mysqli::get_client_info** (void)

string **mysqli_get_client_info** (void)

The mysqli_get_client_info() function is used to return a string representing the client version being used in the MySQLi extension.

Return Values

A string that represents the MySQL client library version

Examples

Example #24 - mysqli_get_client_info

```
<?php

/* We don't need a connection to determine
  the version of mysql client library */

printf("Client library version: %s\n", mysqli_get_client_info());
?>
```

See Also

- mysqli_get_client_version()
- mysqli_get_server_info()
- mysqli_get_server_version()

mysqli::get_client_version

mysqli_get_client_version

mysqli::get_client_version -- mysqli_get_client_version -- Get MySQL client info

Description

int **mysqli::get_client_version** (void)

int **mysqli_get_client_version** (void)

Returns client version number as an integer.

Return Values

A number that represents the MySQL client library version in format: *main_version*10000 + minor_version *100 + sub_version*. For example, 4.1.0 is returned as 40100.

This is useful to quickly determine the version of the client library to know if some capability exits.

Examples

Example #25 - mysqli_get_client_version

```
<?php

/* We don't need a connection to determine
  the version of mysql client library */

printf("Client library version: %d\n", mysqli_get_client_version());
?>
```

See Also

- mysqli_get_client_info()
- mysqli_get_server_info()
- mysqli_get_server_version()

mysqli->host_info

mysqli_get_host_info

mysqli->host_info -- mysqli_get_host_info -- Returns a string representing the type of connection used

Description

Object oriented style (property):

mysqli

string *host_info*;

Procdural style:

string **mysqli_get_host_info** (mysqli `$link`)

The mysqli_get_host_info() function returns a string describing the connection represented by the *`link`* parameter is using (including the server host name).

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

A character string representing the server hostname and the connection type.

Examples

Example #26 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
```

```
   exit();
}

/* print host information */
printf("Host info: %s\n", $mysqli->host_info);

/* close connection */
$mysqli->close();
?>
```

Example #27 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* print host information */
printf("Host info: %s\n", mysqli_get_host_info($link));

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Host info: Localhost via UNIX socket
```

See Also

- mysqli_get_proto_info()

mysqli->protocol_version

mysqli_get_proto_info

mysqli->protocol_version -- mysqli_get_proto_info -- Returns the version of the MySQL protocol used

Description

Object oriented style (property):

mysqli

string *protocol_version*;

Procedural style:

int **mysqli_get_proto_info** (mysqli `$link`)

Returns an integer representing the MySQL protocol version used by the connection represented by the *`link`* parameter.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

Returns an integer representing the protocol version.

Examples

Example #28 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
```

```
   exit();
}

/* print protocol version */
printf("Protocol version: %d\n", $mysqli->protocol_version);

/* close connection */
$mysqli->close();
?>
```

Example #29 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* print protocol version */
printf("Protocol version: %d\n", mysqli_get_proto_info($link));

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Protocol version: 10
```

See Also

- mysqli_get_host_info()

mysqli->server_info

mysqli_get_server_info

mysqli->server_info -- mysqli_get_server_info -- Returns the version of the MySQL server

Description

Object oriented style (property):

mysqli

string *server_info*;

Procedural style:

string **mysqli_get_server_info** (mysqli `$link`)

Returns a string representing the version of the MySQL server that the MySQLi extension is connected to.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

A character string representing the server version.

Examples

Example #30 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
```

```
}

/* print server version */
printf("Server version: %s\n", $mysqli->server_info);

/* close connection */
$mysqli->close();
?>
```

Example #31 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* print server version */
printf("Server version: %s\n", mysqli_get_server_info($link));

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Server version: 4.1.2-alpha-debug
```

See Also

- mysqli_get_client_info()
- mysqli_get_client_version()
- mysqli_get_server_version()

mysqli->server_version

mysqli_get_server_version

mysqli->server_version -- mysqli_get_server_version -- Returns the version of the MySQL server as an integer

Description

Object oriented style (property):

mysqli

int *server_version*;

Procedural style:

int **mysqli_get_server_version** (mysqli `$link`)

The mysqli_get_server_version() function returns the version of the server connected to (represented by the `link` parameter) as an integer.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

An integer representing the server version.

The form of this version number is *main_version * 10000 + minor_version * 100 + sub_version* (i.e. version 4.1.0 is 40100).

Examples

Example #32 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password");
```

```
/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* print server version */
printf("Server version: %d\n", $mysqli->server_version);

/* close connection */
$mysqli->close();
?>
```

Example #33 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* print server version */
printf("Server version: %d\n", mysqli_get_server_version($link));

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Server version: 40102
```

See Also

- mysqli_get_client_info()
- mysqli_get_client_version()
- mysqli_get_server_info()

mysqli::get_warnings

mysqli_get_warnings

mysqli::get_warnings -- mysqli_get_warnings --

Description

object **mysqli::get_warnings** (void)

object **mysqli_get_warnings** (mysqli `$link`)

Warning
This function is currently not documented; only its argument list is available.

mysqli->info

mysqli_info

mysqli->info -- mysqli_info -- Retrieves information about the most recently executed query

Description

Object oriented style (property)

mysqli

string *info*;

Procedural style:

string **mysqli_info** (mysqli `$link`)

The mysqli_info() function returns a string providing information about the last query executed. The nature of this string is provided below:

Possible mysqli_info return values

Query type	**Example result string**
INSERT INTO...SELECT...	Records: 100 Duplicates: 0 Warnings: 0
INSERT INTO...VALUES (...),(...),(...)	Records: 3 Duplicates: 0 Warnings: 0
LOAD DATA INFILE ...	Records: 1 Deleted: 0 Skipped: 0 Warnings: 0
ALTER TABLE ...	Records: 3 Duplicates: 0 Warnings: 0
UPDATE ...	Rows matched: 40 Changed: 40 Warnings: 0

Note

Queries which do not fall into one of the above formats are not supported. In these situations, mysqli_info() will return an empty string.

Parameters

link
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

A character string representing additional information about the most recently executed query.

Examples

Example #34 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$mysqli->query("CREATE TEMPORARY TABLE t1 LIKE City");

/* INSERT INTO .. SELECT */
$mysqli->query("INSERT INTO t1 SELECT * FROM City ORDER BY ID LIMIT 150");
printf("%s\n", $mysqli->info);

/* close connection */
$mysqli->close();
?>
```

Example #35 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}
```

```
mysqli_query($link, "CREATE TEMPORARY TABLE t1 LIKE City");

/* INSERT INTO .. SELECT */
mysqli_query($link, "INSERT INTO t1 SELECT * FROM City ORDER BY ID LIMIT
150");
printf("%s\n", mysqli_info($link));

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Records: 150  Duplicates: 0  Warnings: 0
```

See Also

- mysqli_affected_rows()
- mysqli_warning_count()
- mysqli_num_rows()

mysqli::init

mysqli_init

mysqli::init -- mysqli_init -- Initializes MySQLi and returns a resource for use with mysqli_real_connect()

Description

Object oriented style (method):

mysqli **init** (void)

Procedural style:

mysqli **mysqli_init** (void)

Allocates or initializes a MYSQL object suitable for mysqli_options() and mysqli_real_connect().

Note
Any subsequent calls to any mysqli function (except mysqli_options()) will fail until mysqli_real_connect() was called.

Return Values

Returns an object.

See Also

- mysqli_options()
- mysqli_close()
- mysqli_real_connect()
- mysqli_connect()

mysqli->insert_id

mysqli_insert_id

mysqli->insert_id -- mysqli_insert_id -- Returns the auto generated id used in the last query

Description

Object oriented style (property):

mysqli

int *insert_id*;

Procedural style:

int **mysqli_insert_id** (mysqli `$link`)

The mysqli_insert_id() function returns the ID generated by a query on a table with a column having the AUTO_INCREMENT attribute. If the last query wasn't an INSERT or UPDATE statement or if the modified table does not have a column with the AUTO_INCREMENT attribute, this function will return zero.

Note
Performing an INSERT or UPDATE statement using the LAST_INSERT_ID() function will also modify the value returned by the mysqli_insert_id() function.

Parameters

`link`
: Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

The value of the *AUTO_INCREMENT* field that was updated by the previous query. Returns zero if there was no previous query on the connection or if the query did not update an *AUTO_INCREMENT* value.

Note

If the number is greater than maximal int value, mysqli_insert_id() will return a string.

Examples

Example #36 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$mysqli->query("CREATE TABLE myCity LIKE City");

$query = "INSERT INTO myCity VALUES (NULL, 'Stuttgart', 'DEU', 'Stuttgart',
617000)";
$mysqli->query($query);

printf ("New Record has id %d.\n", $mysqli->insert_id);

/* drop table */
$mysqli->query("DROP TABLE myCity");

/* close connection */
$mysqli->close();
?>
```

Example #37 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

mysqli_query($link, "CREATE TABLE myCity LIKE City");

$query = "INSERT INTO myCity VALUES (NULL, 'Stuttgart', 'DEU', 'Stuttgart',
617000)";
mysqli_query($link, $query);

printf ("New Record has id %d.\n", mysqli_insert_id($link));

/* drop table */
mysqli_query($link, "DROP TABLE myCity");
```

```
/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
New Record has id 1.
```

mysqli::kill

mysqli_kill

mysqli::kill -- mysqli_kill -- Asks the server to kill a MySQL thread

Description

Object oriented style (method)

bool **mysqli::kill** (int `$processid`)

Procedural style:

bool **mysqli_kill** (mysqli `$link`, int `$processid`)

This function is used to ask the server to kill a MySQL thread specified by the *`processid`* parameter. This value must be retrieved by calling the mysqli_thread_id() function.

To stop a running query you should use the SQL command *KILL QUERY processid*.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #38 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* determine our thread id */
$thread_id = $mysqli->thread_id;
```

```
/* Kill connection */
$mysqli->kill($thread_id);

/* This should produce an error */
if (!$mysqli->query("CREATE TABLE myCity LIKE City")) {
   printf("Error: %s\n", $mysqli->error);
   exit;
}

/* close connection */
$mysqli->close();
?>
```

Example #39 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* determine our thread id */
$thread_id = mysqli_thread_id($link);

/* Kill connection */
mysqli_kill($link, $thread_id);

/* This should produce an error */
if (!mysqli_query($link, "CREATE TABLE myCity LIKE City")) {
   printf("Error: %s\n", mysqli_error($link));
   exit;
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Error: MySQL server has gone away
```

See Also

- mysqli_thread_id()

mysqli::more_results

mysqli_more_results

mysqli::more_results -- mysqli_more_results -- Check if there are any more query results from a multi query

Description

bool **mysqli::more_results** (void)

bool **mysqli_more_results** (mysqli `$link`)

Indicates if one or more result sets are available from a previous call to mysqli_multi_query().

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

See mysqli_multi_query().

See Also

- mysqli_multi_query()
- mysqli_next_result()
- mysqli_store_result()
- mysqli_use_result()

mysqli::multi_query

mysqli_multi_query

mysqli::multi_query -- mysqli_multi_query -- Performs a query on the database

Description

Object oriented style (method):

bool **mysqli::multi_query** (string `$query`)

Procedural style:

bool **mysqli_multi_query** (mysqli `$link`, string `$query`)

Executes one or multiple queries which are concatenated by a semicolon.

To retrieve the resultset from the first query you can use mysqli_use_result() or mysqli_store_result(). All subsequent query results can be processed using mysqli_more_results() and mysqli_next_result().

Parameters

link
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

query
The query, as a string.

Return Values

Returns **FALSE** if the first statement failed. To retrieve subsequent errors from other statements you have to call mysqli_next_result() first.

Examples

Example #40 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
```

```
   exit();
}

$query  = "SELECT CURRENT_USER();";
$query .= "SELECT Name FROM City ORDER BY ID LIMIT 20, 5";

/* execute multi query */
if ($mysqli->multi_query($query)) {
   do {
       /* store first result set */
       if ($result = $mysqli->store_result()) {
           while ($row = $result->fetch_row()) {
               printf("%s\n", $row[0]);
           }
           $result->free();
       }
       /* print divider */
       if ($mysqli->more_results()) {
           printf("-----------------\n");
       }
   } while ($mysqli->next_result());
}

/* close connection */
$mysqli->close();
?>
```

Example #41 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query  = "SELECT CURRENT_USER();";
$query .= "SELECT Name FROM City ORDER BY ID LIMIT 20, 5";

/* execute multi query */
if (mysqli_multi_query($link, $query)) {
   do {
       /* store first result set */
       if ($result = mysqli_store_result($link)) {
           while ($row = mysqli_fetch_row($result)) {
               printf("%s\n", $row[0]);
           }
           mysqli_free_result($result);
       }
       /* print divider */
       if (mysqli_more_results($link)) {
           printf("-----------------\n");
       }
   } while (mysqli_next_result($link));
}
```

```
/* close connection */
mysqli_close($link);
?>
```

The above example will output something similar to:

```
my_user@localhost
-----------------
Amersfoort
Maastricht
Dordrecht
Leiden
Haarlemmermeer
```

See Also

- mysqli_use_result()
- mysqli_store_result()
- mysqli_next_result()
- mysqli_more_results()

mysqli::next_result

mysqli_next_result

mysqli::next_result -- mysqli_next_result -- Prepare next result from multi_query

Description

bool **mysqli::next_result** (void)

bool **mysqli_next_result** (mysqli `$link`)

Prepares next result set from a previous call to mysqli_multi_query() which can be retrieved by mysqli_store_result() or mysqli_use_result().

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

See mysqli_multi_query().

See Also

- mysqli_multi_query()
- mysqli_more_results()
- mysqli_store_result()
- mysqli_use_result()

mysqli::options

mysqli_options

mysqli::options -- mysqli_options -- Set options

Description

Object oriented style (method)

bool **mysqli::options** (int `$option`, mixed `$value`)

Procedural style:

bool **mysqli_options** (mysqli `$link`, int `$option`, mixed `$value`)

Used to set extra connect options and affect behavior for a connection.

This function may be called multiple times to set several options.

mysqli_options() should be called after mysqli_init() and before mysqli_real_connect().

Parameters

link
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

option
The option that you want to set. It can be one of the following values:

Valid options

Name	Description
MYSQLI_OPT_CONNECT_TIMEOUT	connection timeout in seconds
MYSQLI_OPT_LOCAL_INFILE	enable/disable use of *LOAD LOCAL INFILE*
MYSQLI_INIT_COMMAND	command to execute after when connecting to MySQL server
MYSQLI_READ_DEFAULT_FILE	Read options from named option file instead of *my.cnf*
MYSQLI_READ_DEFAULT_GROUP	Read options from the named group from *my.cnf* or the file specified with **MYSQL_READ_DEFAULT_FILE**.

value
: The value for the option.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

See mysqli_real_connect().

See Also

- mysqli_init()
- mysqli_real_connect()

mysqli::ping

mysqli_ping

mysqli::ping -- mysqli_ping -- Pings a server connection, or tries to reconnect if the connection has gone down

Description

Object oriented style (method):

bool **mysqli::ping** (void)

Procedural style:

bool **mysqli_ping** (mysqli `$link`)

Checks whether the connection to the server is working. If it has gone down, and global option *mysqli.reconnect* is enabled an automatic reconnection is attempted.

This function can be used by clients that remain idle for a long while, to check whether the server has closed the connection and reconnect if necessary.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #42 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* check if server is alive */
```

```
if ($mysqli->ping()) {
   printf ("Our connection is ok!\n");
} else {
   printf ("Error: %s\n", $mysqli->error);
}

/* close connection */
$mysqli->close();
?>
```

Example #43 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* check if server is alive */
if (mysqli_ping($link)) {
   printf ("Our connection is ok!\n");
} else {
   printf ("Error: %s\n", mysqli_error($link));
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Our connection is ok!
```

mysqli::prepare

mysqli_prepare

mysqli::prepare -- mysqli_prepare -- Prepare a SQL statement for execution

Description

Object oriented style (method)

mysqli_stmt **prepare** (string `$query`)

Procedure style:

mysqli_stmt **mysqli_prepare** (mysqli `$link`, string `$query`)

Prepares the SQL query pointed to by the null-terminated string query, and returns a statement handle to be used for further operations on the statement. The query must consist of a single SQL statement.

The parameter markers must be bound to application variables using mysqli_stmt_bind_param() and/or mysqli_stmt_bind_result() before executing the statement or fetching rows.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

`query`
The query, as a string.

Note
You should not add a terminating semicolon or \g to the statement.

This parameter can include one or more parameter markers in the SQL statement by embedding question mark (*?*) characters at the appropriate positions.

Note
The markers are legal only in certain places in SQL statements. For example, they are allowed in the *VALUES()* list of an *INSERT* statement (to specify column values for a row), or in a comparison with a column in a *WHERE* clause to specify a comparison value. However, they are not allowed for identifiers (such as table or column names), in the

select list that names the columns to be returned by a *SELECT* statement, or to specify both operands of a binary operator such as the = equal sign. The latter restriction is necessary because it would be impossible to determine the parameter type. It's not allowed to compare marker with *NULL* by *? IS NULL* too. In general, parameters are legal only in Data Manipulation Languange (DML) statements, and not in Data Defination Language (DDL) statements.

Return Values

mysqli_prepare() returns a statement object or **FALSE** if an error occured.

Examples

Example #44 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$city = "Amersfoort";

/* create a prepared statement */
if ($stmt = $mysqli->prepare("SELECT District FROM City WHERE Name=?")) {

   /* bind parameters for markers */
   $stmt->bind_param("s", $city);

   /* execute query */
   $stmt->execute();

   /* bind result variables */
   $stmt->bind_result($district);

   /* fetch value */
   $stmt->fetch();

   printf("%s is in district %s\n", $city, $district);

   /* close statement */
   $stmt->close();
}

/* close connection */
$mysqli->close();
?>
```

Example #45 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$city = "Amersfoort";

/* create a prepared statement */
if ($stmt = mysqli_prepare($link, "SELECT District FROM City WHERE Name=?")) {

   /* bind parameters for markers */
   mysqli_stmt_bind_param($stmt, "s", $city);

   /* execute query */
   mysqli_stmt_execute($stmt);

   /* bind result variables */
   mysqli_stmt_bind_result($stmt, $district);

   /* fetch value */
   mysqli_stmt_fetch($stmt);

   printf("%s is in district %s\n", $city, $district);

   /* close statement */
   mysqli_stmt_close($stmt);
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Amersfoort is in district Utrecht
```

See Also

- mysqli_stmt_execute()
- mysqli_stmt_fetch()
- mysqli_stmt_bind_param()
- mysqli_stmt_bind_result()
- mysqli_stmt_close()

mysqli::query

mysqli_query

mysqli::query -- mysqli_query -- Performs a query on the database

Description

Object oriented style (method):

mixed **mysqli::query** (string `$query` [, int `$resultmode`])

Procedural style:

mixed **mysqli_query** (mysqli `$link`, string `$query` [, int `$resultmode`])

Performs a *query* against the database.

Functionally, using this function is identical to calling mysqli_real_query() followed either by mysqli_use_result() or mysqli_store_result().

Parameters

link
: Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

query
: The query string.

resultmode
: Either the constant **MYSQLI_USE_RESULT** or **MYSQLI_STORE_RESULT** depending on the desired behavior. By default, **MYSQLI_STORE_RESULT** is used. If you use **MYSQLI_USE_RESULT** all subsequent calls will return error *Commands out of sync* unless you call mysqli_free_result()

Return Values

Returns **TRUE** on success or **FALSE** on failure. For *SELECT, SHOW, DESCRIBE* or *EXPLAIN* mysqli_query() will return a result object.

Examples

Example #46 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* Create table doesn't return a resultset */
if ($mysqli->query("CREATE TEMPORARY TABLE myCity LIKE City") === TRUE) {
   printf("Table myCity successfully created.\n");
}

/* Select queries return a resultset */
if ($result = $mysqli->query("SELECT Name FROM City LIMIT 10")) {
   printf("Select returned %d rows.\n", $result->num_rows);

   /* free result set */
   $result->close();
}

/* If we have to retrieve large amount of data we use MYSQLI_USE_RESULT */
if ($result = $mysqli->query("SELECT * FROM City", MYSQLI_USE_RESULT)) {

   /* Note, that we can't execute any functions which interact with the
      server until result set was closed. All calls will return an
      'out of sync' error */
   if (!$mysqli->query("SET @a:='this will not work'")) {
       printf("Error: %s\n", $mysqli->error);
   }
   $result->close();
}

$mysqli->close();
?>
```

Example #47 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* Create table doesn't return a resultset */
if (mysqli_query($link, "CREATE TEMPORARY TABLE myCity LIKE City") === TRUE) {
   printf("Table myCity successfully created.\n");
}

/* Select queries return a resultset */
```

```
if ($result = mysqli_query($link, "SELECT Name FROM City LIMIT 10")) {
   printf("Select returned %d rows.\n", mysqli_num_rows($result));

   /* free result set */
   mysqli_free_result($result);
}

/* If we have to retrieve large amount of data we use MYSQLI_USE_RESULT */
if ($result = mysqli_query($link, "SELECT * FROM City", MYSQLI_USE_RESULT)) {

   /* Note, that we can't execute any functions which interact with the
      server until result set was closed. All calls will return an
      'out of sync' error */
   if (!mysqli_query($link, "SET @a:='this will not work'")) {
       printf("Error: %s\n", mysqli_error($link));
   }
   mysqli_free_result($result);
}

mysqli_close($link);
?>
```

The above example will output:

```
Table myCity successfully created.
Select returned 10 rows.
Error: Commands out of sync;  You can't run this command now
```

See Also

- mysqli_real_query()
- mysqli_multi_query()
- mysqli_free_result()

mysqli::real_connect

mysqli_real_connect

mysqli::real_connect -- mysqli_real_connect -- Opens a connection to a mysql server

Description

Object oriented style (method)

bool **mysqli::real_connect** ([string `$host` [, string `$username` [, string `$passwd` [, string `$dbname` [, int `$port` [, string `$socket` [, int `$flags`]]]]]])

Procedural style

bool **mysqli_real_connect** (mysqli `$link` [, string `$host` [, string `$username` [, string `$passwd` [, string `$dbname` [, int `$port` [, string `$socket` [, int `$flags`]]]]]])

Establish a connection to a MySQL database engine.

This function differs from mysqli_connect():

- mysqli_real_connect() needs a valid object which has to be created by function mysqli_init().
- With function mysqli_options() you can set various options for connection.
- There is a *`flags`* parameter.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

`host`
Can be either a host name or an IP address. Passing the **NULL** value or the string "localhost" to this parameter, the local host is assumed. When possible, pipes will be used instead of the TCP/IP protocol.

`username`
The MySQL user name.

`passwd`
If provided or **NULL**, the MySQL server will attempt to authenticate the user against those user records which have no password only. This allows one username to be used with different permissions (depending on if a password as provided or not).

dbname
If provided will specify the default database to be used when performing queries.

port
Specifies the port number to attempt to connect to the MySQL server.

socket
Specifies the socket or named pipe that should be used.

Note
Specifying the *socket* parameter will not explicitly determine the type of connection to be used when connecting to the MySQL server. How the connection is made to the MySQL database is determined by the *host* parameter.

flags
With the parameter *flags* you can set different connection options:

Supported flags

Name	Description
MYSQLI_CLIENT_COMPRESS	Use compression protocol
MYSQLI_CLIENT_FOUND_ROWS	return number of matched rows, not the number of affected rows
MYSQLI_CLIENT_IGNORE_SPACE	Allow spaces after function names. Makes all function names reserved words.
MYSQLI_CLIENT_INTERACTIVE	Allow *interactive_timeout* seconds (instead of *wait_timeout* seconds) of inactivity before closing the connection
MYSQLI_CLIENT_SSL	Use SSL (encryption)

Note
For security reasons the **MULTI_STATEMENT** flag is not supported in PHP. If you want to execute multiple queries use the mysqli_multi_query() function.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #48 - Object oriented style

```
<?php

/* create a connection object which is not connected */
$mysqli = mysqli_init();

/* set connection options */
$mysqli->options(MYSQLI_INIT_COMMAND, "SET AUTOCOMMIT=0");
$mysqli->options(MYSQLI_OPT_CONNECT_TIMEOUT, 5);

/* connect to server */
$mysqli->real_connect('localhost', 'my_user', 'my_password', 'world');

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

printf ("Connection: %s\n.", $mysqli->host_info);

$mysqli->close();
?>
```

Example #49 - Procedural style

```
<?php

/* create a connection object which is not connected */
$link = mysqli_init();

/* set connection options */
mysqli_options($link, MYSQLI_INIT_COMMAND, "SET AUTOCOMMIT=0");
mysqli_options($link, MYSQLI_OPT_CONNECT_TIMEOUT, 5);

/* connect to server */
mysqli_real_connect($link, 'localhost', 'my_user', 'my_password', 'world');

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

printf ("Connection: %s\n.", mysqli_get_host_info($link));

mysqli_close($link);
?>
```

The above example will output:

```
Connection: Localhost via UNIX socket
```

See Also

- mysqli_connect()
- mysqli_init()
- mysqli_options()
- mysqli_ssl_set()
- mysqli_close()

mysqli::real_escape_string

mysqli_real_escape_string

mysqli::real_escape_string -- mysqli_real_escape_string -- Escapes special characters in a string for use in a SQL statement, taking into account the current charset of the connection

Description

Object oriented style (both methods are equivalent):

string **mysqli::escape_string** (string `$escapestr`)

string **real_escape_string** (string `$escapestr`)

Procedural style:

string **mysqli_real_escape_string** (mysqli `$link`, string `$escapestr`)

This function is used to create a legal SQL string that you can use in an SQL statement. The given string is encoded to an escaped SQL string, taking into account the current character set of the connection.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

`escapestr`
The string to be escaped. Characters encoded are *NUL (ASCII 0), \n, \r, \, ', ", and Control-Z*.

Return Values

Returns an escaped string.

Examples

Example #50 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
```

```
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$mysqli->query("CREATE TEMPORARY TABLE myCity LIKE City");

$city = "'s Hertogenbosch";

/* this query will fail, cause we didn't escape $city */
if (!$mysqli->query("INSERT into myCity (Name) VALUES ('$city')")) {
   printf("Error: %s\n", $mysqli->sqlstate);
}

$city = $mysqli->real_escape_string($city);

/* this query with escaped $city will work */
if ($mysqli->query("INSERT into myCity (Name) VALUES ('$city')")) {
   printf("%d Row inserted.\n", $mysqli->affected_rows);
}

$mysqli->close();
?>
```

Example #51 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

mysqli_query($link, "CREATE TEMPORARY TABLE myCity LIKE City");

$city = "'s Hertogenbosch";

/* this query will fail, cause we didn't escape $city */
if (!mysqli_query($link, "INSERT into myCity (Name) VALUES ('$city')")) {
   printf("Error: %s\n", mysqli_sqlstate($link));
}

$city = mysqli_real_escape_string($link, $city);

/* this query with escaped $city will work */
if (mysqli_query($link, "INSERT into myCity (Name) VALUES ('$city')")) {
   printf("%d Row inserted.\n", mysqli_affected_rows($link));
}

mysqli_close($link);
?>
```

The above example will output:

```
Error: 42000
```

```
1 Row inserted.
```

See Also

- mysqli_character_set_name()

mysqli::real_query

mysqli_real_query

mysqli::real_query -- mysqli_real_query -- Execute an SQL query

Description

Object oriented style (method):

bool **real_query** (string `$query`)

Procedural style

bool **mysqli_real_query** (mysqli `$link`, string `$query`)

Executes a single query against the database whose result can then be retrieved or stored using the mysqli_store_result() or mysqli_use_result() functions.

In order to determine if a given query should return a result set or not, see mysqli_field_count() .

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

`query`
The query, as a string.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- mysqli_query()
- mysqli_store_result()
- mysqli_use_result()

mysqli::rollback

mysqli_rollback

mysqli::rollback -- mysqli_rollback -- Rolls back current transaction

Description

Object oriented style (method):

bool **mysqli::rollback** (void)

Procedural style:

bool **mysqli_rollback** (mysqli `$link`)

Rollbacks the current transaction for the database.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #52 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* disable autocommit */
$mysqli->autocommit(FALSE);

$mysqli->query("CREATE TABLE myCity LIKE City");
$mysqli->query("ALTER TABLE myCity Type=InnoDB");
$mysqli->query("INSERT INTO myCity SELECT * FROM City LIMIT 50");
```

```
/* commit insert */
$mysqli->commit();

/* delete all rows */
$mysqli->query("DELETE FROM myCity");

if ($result = $mysqli->query("SELECT COUNT(*) FROM myCity")) {
   $row = $result->fetch_row();
   printf("%d rows in table myCity.\n", $row[0]);
   /* Free result */
   $result->close();
}

/* Rollback */
$mysqli->rollback();

if ($result = $mysqli->query("SELECT COUNT(*) FROM myCity")) {
   $row = $result->fetch_row();
   printf("%d rows in table myCity (after rollback).\n", $row[0]);
   /* Free result */
   $result->close();
}

/* Drop table myCity */
$mysqli->query("DROP TABLE myCity");

$mysqli->close();
?>
```

Example #53 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* disable autocommit */
mysqli_autocommit($link, FALSE);

mysqli_query($link, "CREATE TABLE myCity LIKE City");
mysqli_query($link, "ALTER TABLE myCity Type=InnoDB");
mysqli_query($link, "INSERT INTO myCity SELECT * FROM City LIMIT 50");

/* commit insert */
mysqli_commit($link);

/* delete all rows */
mysqli_query($link, "DELETE FROM myCity");

if ($result = mysqli_query($link, "SELECT COUNT(*) FROM myCity")) {
   $row = mysqli_fetch_row($result);
   printf("%d rows in table myCity.\n", $row[0]);
   /* Free result */
   mysqli_free_result($result);
```

```
}

/* Rollback */
mysqli_rollback($link);

if ($result = mysqli_query($link, "SELECT COUNT(*) FROM myCity")) {
   $row = mysqli_fetch_row($result);
   printf("%d rows in table myCity (after rollback).\n", $row[0]);
   /* Free result */
   mysqli_free_result($result);
}

/* Drop table myCity */
mysqli_query($link, "DROP TABLE myCity");

mysqli_close($link);
?>
```

The above example will output:

```
0 rows in table myCity.
50 rows in table myCity (after rollback).
```

See Also

- mysqli_commit()
- mysqli_autocommit()

mysqli::select_db

mysqli_select_db

mysqli::select_db -- mysqli_select_db -- Selects the default database for database queries

Description

Object oriented style (method):

bool **mysqli::select_db** (string `$dbname`)

Procedural style:

bool **mysqli_select_db** (mysqli `$link`, string `$dbname`)

Selects the default database to be used when performing queries against the database connection.

Note
This function should only be used to change the default database for the connection. You can select the default database with 4th parameter in mysqli_connect().

Parameters

link
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

dbname
The database name.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #54 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "test");
```

```
/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* return name of current default database */
if ($result = $mysqli->query("SELECT DATABASE()")) {
   $row = $result->fetch_row();
   printf("Default database is %s.\n", $row[0]);
   $result->close();
}

/* change db to world db */
$mysqli->select_db("world");

/* return name of current default database */
if ($result = $mysqli->query("SELECT DATABASE()")) {
   $row = $result->fetch_row();
   printf("Default database is %s.\n", $row[0]);
   $result->close();
}

$mysqli->close();
?>
```

Example #55 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "test");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* return name of current default database */
if ($result = mysqli_query($link, "SELECT DATABASE()")) {
   $row = mysqli_fetch_row($result);
   printf("Default database is %s.\n", $row[0]);
   mysqli_free_result($result);
}

/* change db to world db */
mysqli_select_db($link, "world");

/* return name of current default database */
if ($result = mysqli_query($link, "SELECT DATABASE()")) {
   $row = mysqli_fetch_row($result);
   printf("Default database is %s.\n", $row[0]);
   mysqli_free_result($result);
}

mysqli_close($link);
?>
```

The above example will output:

```
Default database is test.
Default database is world.
```

See Also

- mysqli_connect()
- mysqli_real_connect()

mysqli::set_charset

mysqli_set_charset

mysqli::set_charset -- mysqli_set_charset -- Sets the default client character set

Description

Object oriented style (method):

bool **mysqli::set_charset** (string `$charset`)

Procedural style:

bool **mysqli_set_charset** (mysqli `$link`, string `$charset`)

Sets the default character set to be used when sending data from and to the database server.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

`charset`
The charset to be set as default.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
To use this function on a Windows platform you need MySQL client library version 4.1.11 or above (for MySQL 5.0 you need 5.0.6 or above).

Examples

Example #56 - Object oriented style
`<?php`

```
$mysqli = new mysqli("localhost", "my_user", "my_password", "test");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* change character set to utf8 */
if (!$mysqli->set_charset("utf8")) {
   printf("Error loading character set utf8: %s\n", $mysqli->error);
} else {
   printf("Current character set: %s\n", $mysqli->character_set_name());
}

$mysqli->close();
?>
```

Example #57 - Procedural style

```
<?php
$link = mysqli_connect('localhost', 'my_user', 'my_password', 'test');

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* change character set to utf8 */
if (!mysqli_set_charset($link, "utf8")) {
   printf("Error loading character set utf8: %s\n", mysqli_error($link));
} else {
   printf("Current character set: %s\n", mysqli_character_set_name($link));
}

mysqli_close($link);
?>
```

The above example will output:

```
Current character set: utf8
```

See Also

- mysqli_character_set_name()
- mysqli_real_escape_string()
- » List of character sets that MySQL supports

mysqli::set_local_infile_default

mysqli_set_local_infile_default

mysqli::set_local_infile_default -- mysqli_set_local_infile_default -- Unsets user defined handler for load local infile command

Description

void **mysqli_set_local_infile_default** (mysqli `$link`)

Deactivates a *LOAD DATA INFILE LOCAL* handler previously set with mysqli_set_local_infile_handler().

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

No value is returned.

Examples

See mysqli_set_local_infile_handler() examples

See Also

- mysqli_set_local_infile_handler()

mysqli::set_local_infile_handler

mysqli_set_local_infile_handler

mysqli::set_local_infile_handler -- mysqli_set_local_infile_handler -- Set callback function for LOAD DATA LOCAL INFILE command

Description

bool **mysqli_set_local_infile_handler** (mysqli `$link`, callback `$read_func`)

Object oriented style (method)

mysqli

bool **set_local_infile_handler** (mysqli `$link`, callback `$read_func`)

Set callback function for LOAD DATA LOCAL INFILE command

The callbacks task is to read input from the file specified in the *LOAD DATA LOCAL INFILE* and to reformat it into the format understood by *LOAD DATA INFILE*.

The returned data needs to match the format speficied in the *LOAD DATA*

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

`read_func`
A callback function or object method taking the following parameters:

`stream`
A PHP stream associated with the SQL commands INFILE

`&buffer`
A string buffer to store the rewritten input into

`buflen`
The maximum number of characters to be stored in the buffer

`&errormsg`
If an error occures you can store an error message in here

The callback function should return the number of characters stored in the *buffer* or a negative value if an error occured.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #58 - Object oriented style

```
<?php
 $db = mysqli_init();
 $db->real_connect("localhost","root","","test");

 function callme($stream, &$buffer, $buflen, &$errmsg)
 {
   $buffer = fgets($stream);

   echo $buffer;

   // convert to upper case and replace "," delimiter with [TAB]
   $buffer = strtoupper(str_replace(",", "\t", $buffer));

   return strlen($buffer);
 }


 echo "Input:\n";

 $db->set_local_infile_handler("callme");
 $db->query("LOAD DATA LOCAL INFILE 'input.txt' INTO TABLE t1");
 $db->set_local_infile_default();

 $res = $db->query("SELECT * FROM t1");

 echo "\nResult:\n";
 while ($row = $res->fetch_assoc()) {
   echo join(",", $row)."\n";
 }
?>
```

Example #59 - Procedural style

```
<?php
 $db = mysqli_init();
 mysqli_real_connect($db, "localhost","root","","test");

 function callme($stream, &$buffer, $buflen, &$errmsg)
 {
   $buffer = fgets($stream);

   echo $buffer;
```

```
    // convert to upper case and replace "," delimiter with [TAB]
    $buffer = strtoupper(str_replace(",", "\t", $buffer));

    return strlen($buffer);
  }


  echo "Input:\n";

  mysqli_set_local_infile_handler($db, "callme");
  mysqli_query($db, "LOAD DATA LOCAL INFILE 'input.txt' INTO TABLE t1");
  mysqli_set_local_infile_default($db);

  $res = mysqli_query($db, "SELECT * FROM t1");


  echo "\nResult:\n";
  while ($row = mysqli_fetch_assoc($res)) {
    echo join(",", $row)."\n";
  }
 ?>
```

The above example will output:

```
Input:
23,foo
42,bar

Output:
23,FOO
42,BAR
```

See Also

- mysqli_set_local_infile_default()

mysqli->sqlstate

mysqli_sqlstate

mysqli->sqlstate -- mysqli_sqlstate -- Returns the SQLSTATE error from previous MySQL operation

Description

Object oriented style (property):

mysqli

string *sqlstate*;

Procedural style:

string **mysqli_sqlstate** (mysqli `$link`)

Returns a string containing the SQLSTATE error code for the last error. The error code consists of five characters. *'00000'* means no error. The values are specified by ANSI SQL and ODBC. For a list of possible values, see » http://dev.mysql.com/doc/mysql/en/error-handling.html.

Note
Note that not all MySQL errors are yet mapped to SQLSTATE's. The value *HY000* (general error) is used for unmapped errors.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

Returns a string containing the SQLSTATE error code for the last error. The error code consists of five characters. *'00000'* means no error.

Examples

Example #60 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* Table City already exists, so we should get an error */
if (!$mysqli->query("CREATE TABLE City (ID INT, Name VARCHAR(30))")) {
   printf("Error - SQLSTATE %s.\n", $mysqli->sqlstate);
}

$mysqli->close();
?>
```

Example #61 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* Table City already exists, so we should get an error */
if (!mysqli_query($link, "CREATE TABLE City (ID INT, Name VARCHAR(30))")) {
   printf("Error - SQLSTATE %s.\n", mysqli_sqlstate($link));
}

mysqli_close($link);
?>
```

The above example will output:

```
Error - SQLSTATE 42S01.
```

See Also

- mysqli_errno()
- mysqli_error()

mysqli::ssl_set

mysqli_ssl_set

mysqli::ssl_set -- mysqli_ssl_set -- Used for establishing secure connections using SSL

Description

Object oriented style (method):

bool **mysqli::ssl_set** (string `$key`, string `$cert`, string `$ca`, string `$capath`, string `$cipher`)

Procedural style:

bool **mysqli_ssl_set** (mysqli `$link`, string `$key`, string `$cert`, string `$ca`, string `$capath`, string `$cipher`)

Used for establishing secure connections using SSL. It must be called before mysqli_real_connect(). This function does nothing unless OpenSSL support is enabled.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

`key`
The path name to the key file.

`cert`
The path name to the certificate file.

`ca`
The path name to the certificate authority file.

`capath`
The pathname to a directory that contains trusted SSL CA certificates in PEM format.

`cipher`
A list of allowable ciphers to use for SSL encryption.

Any unused SSL parameters may be given as **NULL**

Return Values

This function always returns **TRUE** value. If SSL setup is incorrect mysqli_real_connect() will return an error when you attempt to connect.

See Also

- mysqli_options()
- mysqli_real_connect()

mysqli::stat

mysqli_stat

mysqli::stat -- mysqli_stat -- Gets the current system status

Description

Object oriented style (method):

string **mysqli::stat** (void)

Procedural style:

string **mysqli_stat** (mysqli `$link`)

mysqli_stat() returns a string containing information similar to that provided by the 'mysqladmin status' command. This includes uptime in seconds and the number of running threads, questions, reloads, and open tables.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

A string describing the server status. **FALSE** if an error occurred.

Examples

Example #62 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

printf ("System status: %s\n", $mysqli->stat());

$mysqli->close();
?>
```

Example #63 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

printf("System status: %s\n", mysqli_stat($link));

mysqli_close($link);
?>
```

The above example will output:

```
System status: Uptime: 272  Threads: 1  Questions: 5340  Slow queries: 0
Opens: 13  Flush tables: 1  Open tables: 0  Queries per second avg: 19.632
Memory in use: 8496K  Max memory used: 8560K
```

See Also

- mysqli_get_server_info()

mysqli::stmt_init

mysqli_stmt_init

mysqli::stmt_init -- mysqli_stmt_init -- Initializes a statement and returns an object for use with mysqli_stmt_prepare

Description

Object oriented style (property):

mysqli

mysqli_stmt **stmt_init** (void)

Procedural style :

mysqli_stmt **mysqli_stmt_init** (mysqli `$link`)

Allocates and initializes a statement object suitable for mysqli_stmt_prepare().

Note
Any subsequent calls to any mysqli_stmt function will fail until mysqli_stmt_prepare() was called.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

Returns an object.

See Also

- mysqli_stmt_prepare()

mysqli::store_result

mysqli_store_result

mysqli::store_result -- mysqli_store_result -- Transfers a result set from the last query

Description

Object oriented style (method):

mysqli_result **store_result** (void)

Procedural style:

mysqli_result **mysqli_store_result** (mysqli `$link`)

Transfers the result set from the last query on the database connection represented by the *`link`* parameter to be used with the mysqli_data_seek() function.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

Returns a buffered result object or **FALSE** if an error occurred.

Note
mysqli_store_result() returns **FALSE** in case the query didn't return a result set (if the query was, for example an INSERT statement). This function also returns **FALSE** if the reading of the result set failed. You can check if you have got an error by checking if mysqli_error() doesn't return an empty string, if mysqli_errno() returns a non zero value, or if mysqli_field_count() returns a non zero value. Also possible reason for this function returning **FALSE** after successfull call to mysqli_query() can be too large result set (memory for it cannot be allocated). If mysqli_field_count() returns a non-zero value, the statement should have produced a non-empty result set.

Notes

Note
Although it is always good practice to free the memory used by the result of a query using the mysqli_free_result() function, when transfering large result sets using the mysqli_store_result() this becomes particularly important.

Examples

See mysqli_multi_query().

See Also

- mysqli_real_query()
- mysqli_use_result()

mysqli::thread_id

mysqli_thread_id

mysqli::thread_id -- mysqli_thread_id -- Returns the thread ID for the current connection

Description

Object oriented style (property):

mysqli

int *thread_id*;

Procedural style:

int **mysqli_thread_id** (mysqli `$link`)

The mysqli_thread_id() function returns the thread ID for the current connection which can then be killed using the mysqli_kill() function. If the connection is lost and you reconnect with mysqli_ping(), the thread ID will be other. Therefore you should get the thread ID only when you need it.

Note
The thread ID is assigned on a connection-by-connection basis. Hence, if the connection is broken and then re-established a new thread ID will be assigned. To kill a running query you can use the SQL command *KILL QUERY processid*.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

Returns the Thread ID for the current connection.

Examples

Example #64 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* determine our thread id */
$thread_id = $mysqli->thread_id;

/* Kill connection */
$mysqli->kill($thread_id);

/* This should produce an error */
if (!$mysqli->query("CREATE TABLE myCity LIKE City")) {
   printf("Error: %s\n", $mysqli->error);
   exit;
}

/* close connection */
$mysqli->close();
?>
```

Example #65 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* determine our thread id */
$thread_id = mysqli_thread_id($link);

/* Kill connection */
mysqli_kill($link, $thread_id);

/* This should produce an error */
if (!mysqli_query($link, "CREATE TABLE myCity LIKE City")) {
   printf("Error: %s\n", mysqli_error($link));
   exit;
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Error: MySQL server has gone away
```

See Also

- mysqli_kill()

mysqli::thread_safe

mysqli_thread_safe

mysqli::thread_safe -- mysqli_thread_safe -- Returns whether thread safety is given or not

Description

Procedural style:

bool **mysqli_thread_safe** (void)

Tells whether the client library is compiled as thread-safe.

Return Values

TRUE if the client library is thread-safe, otherwise **FALSE**.

mysqli::use_result

mysqli_use_result

mysqli::use_result -- mysqli_use_result -- Initiate a result set retrieval

Description

Object oriented style (method):

mysqli_result **use_result** (void)

Procedural style:

mysqli_result **mysqli_use_result** (mysqli `$link`)

Used to initiate the retrieval of a result set from the last query executed using the mysqli_real_query() function on the database connection.

Either this or the mysqli_store_result() function must be called before the results of a query can be retrieved, and one or the other must be called to prevent the next query on that database connection from failing.

Note
The mysqli_use_result() function does not transfer the entire result set from the database and hence cannot be used functions such as mysqli_data_seek() to move to a particular row within the set. To use this functionality, the result set must be stored using mysqli_store_result(). One should not use mysqli_use_result() if a lot of processing on the client side is performed, since this will tie up the server and prevent other threads from updating any tables from which the data is being fetched.

Return Values

Returns an unbuffered result object or **FALSE** if an error occurred.

Examples

Example #66 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
```

```
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query  = "SELECT CURRENT_USER();";
$query .= "SELECT Name FROM City ORDER BY ID LIMIT 20, 5";

/* execute multi query */
if ($mysqli->multi_query($query)) {
   do {
       /* store first result set */
       if ($result = $mysqli->use_result()) {
           while ($row = $result->fetch_row()) {
               printf("%s\n", $row[0]);
           }
           $result->close();
       }
       /* print divider */
       if ($mysqli->more_results()) {
           printf("-----------------\n");
       }
   } while ($mysqli->next_result());
}

/* close connection */
$mysqli->close();
?>
```

Example #67 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query  = "SELECT CURRENT_USER();";
$query .= "SELECT Name FROM City ORDER BY ID LIMIT 20, 5";

/* execute multi query */
if (mysqli_multi_query($link, $query)) {
   do {
       /* store first result set */
       if ($result = mysqli_use_result($link)) {
           while ($row = mysqli_fetch_row($result)) {
               printf("%s\n", $row[0]);
           }
           mysqli_free_result($result);
       }
       /* print divider */
       if (mysqli_more_results($link)) {
           printf("-----------------\n");
       }
   } while (mysqli_next_result($link));
}
```

```
/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
my_user@localhost
-----------------
Amersfoort
Maastricht
Dordrecht
Leiden
Haarlemmermeer
```

See Also

- mysqli_real_query()
- mysqli_store_result()

mysqli::warning_count

mysqli_warning_count

mysqli::warning_count -- mysqli_warning_count -- Returns the number of warnings from the last query for the given link

Description

Object oriented style (property):

mysqli

int *warning_count*;

Procedural style:

int **mysqli_warning_count** (mysqli `$link`)

Returns the number of warnings from the last query in the connection.

Note
For retrieving warning messages you can use the SQL command *SHOW WARNINGS [limit row_count]*.

Parameters

`link`
Procedural style only: A link identifier returned by mysqli_connect() or mysqli_init()

Return Values

Number of warnings or zero if there are no warnings.

Examples

Example #68 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$mysqli->query("CREATE TABLE myCity LIKE City");

/* a remarkable city in Wales */
$query = "INSERT INTO myCity (CountryCode, Name) VALUES('GBR',
       'Llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch')";

$mysqli->query($query);

if ($mysqli->warning_count) {
   if ($result = $mysqli->query("SHOW WARNINGS")) {
       $row = $result->fetch_row();
       printf("%s (%d): %s\n", $row[0], $row[1], $row[2]);
       $result->close();
   }
}

/* close connection */
$mysqli->close();
?>
```

Example #69 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

mysqli_query($link, "CREATE TABLE myCity LIKE City");

/* a remarkable long city name in Wales */
$query = "INSERT INTO myCity (CountryCode, Name) VALUES('GBR',
       'Llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch')";

mysqli_query($link, $query);

if (mysqli_warning_count($link)) {
   if ($result = mysqli_query($link, "SHOW WARNINGS")) {
       $row = mysqli_fetch_row($result);
       printf("%s (%d): %s\n", $row[0], $row[1], $row[2]);
       mysqli_free_result($result);
   }
```

```
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Warning (1264): Data truncated for column 'Name' at row 1
```

See Also

- mysqli_errno()
- mysqli_error()
- mysqli_sqlstate()

The MySQLi_STMT class

Introduction

Represents a prepared statement.

Class synopsis

MySQLi_STMT

MySQLi_STMT {

/* Properties */

int *affected_rows*;

int *errno*;

string *error*;

int *field_count*;

int *insert_id*;

int *num_rows*;

int *param_count*;

string *sqlstate*;

/* Methods */

int **mysqli_stmt_affected_rows** (mysqli_stmt `$stmt`)

int **mysqli_stmt_attr_get** (mysqli_stmt `$stmt`, int `$attr`)

bool **mysqli_stmt_attr_set** (mysqli_stmt `$stmt`, int `$attr`, int `$mode`)

bool **mysqli_stmt::bind_param** (string `$types`, mixed `&$var1` [, mixed `&$...`])

bool **mysqli_stmt::bind_result** (mixed `&$var1` [, mixed `&$...`])

bool **mysqli_stmt::close** (void)

void **mysqli_stmt::data_seek** (int `$offset`)

int **mysqli_stmt_errno** (mysqli_stmt `$stmt`)

string **mysqli_stmt_error** (mysqli_stmt `$stmt`)

bool **mysqli_stmt::execute** (void)

bool **mysqli_stmt::fetch** (void)

int **mysqli_stmt_field_count** (mysqli_stmt `$stmt`)

void **mysqli_stmt::free_result** (void)

object **mysqli_stmt::get_warnings** (mysqli_stmt `$stmt`)

mixed **mysqli_stmt_insert_id** (mysqli_stmt `$stmt`)

int **mysqli_stmt_num_rows** (mysqli_stmt `$stmt`)

int **mysqli_stmt_param_count** (mysqli_stmt `$stmt`)

mixed **mysqli_stmt::prepare** (string `$query`)

bool **mysqli_stmt::reset** (void)

mysqli_result **mysqli_stmt::result_metadata** (void)

bool **mysqli_stmt::send_long_data** (int `$param_nr`, string `$data`)

string **mysqli_stmt_sqlstate** (mysqli_stmt `$stmt`)

bool **mysqli_stmt::store_result** (void)
}

mysqli_stmt->affected_rows

mysqli_stmt_affected_rows

mysqli_stmt->affected_rows -- mysqli_stmt_affected_rows -- Returns the total number of rows changed, deleted, or inserted by the last executed statement

Description

Object oriented style (property):

mysqli_stmt

int *affected_rows*;

Procedural style :

int **mysqli_stmt_affected_rows** (mysqli_stmt `$stmt`)

Returns the number of rows affected by *INSERT*, *UPDATE*, or *DELETE* query.

This function only works with queries which update a table. In order to get the number of rows from a SELECT query, use mysqli_stmt_num_rows() instead.

Parameters

`stmt`
Procedural style only: A statement identifier returned by mysqli_stmt_init().

Return Values

An integer greater than zero indicates the number of rows affected or retrieved. Zero indicates that no records where updated for an UPDATE/DELETE statement, no rows matched the WHERE clause in the query or that no query has yet been executed. -1 indicates that the query has returned an error.

Note
If the number of affected rows is greater than maximal PHP int value, the number of affected rows will be returned as a string value.

Examples

Example #70 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* create temp table */
$mysqli->query("CREATE TEMPORARY TABLE myCountry LIKE Country");

$query = "INSERT INTO myCountry SELECT * FROM Country WHERE Code LIKE ?";

/* prepare statement */
if ($stmt = $mysqli->prepare($query)) {

   /* Bind variable for placeholder */
   $code = 'A%';
   $stmt->bind_param("s", $code);

   /* execute statement */
   $stmt->execute();

   printf("rows inserted: %d\n", $stmt->affected_rows);

   /* close statement */
   $stmt->close();
}

/* close connection */
$mysqli->close();
?>
```

Example #71 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* create temp table */
mysqli_query($link, "CREATE TEMPORARY TABLE myCountry LIKE Country");

$query = "INSERT INTO myCountry SELECT * FROM Country WHERE Code LIKE ?";

/* prepare statement */
if ($stmt = mysqli_prepare($link, $query)) {
```

```
    /* Bind variable for placeholder */
    $code = 'A%';
    mysqli_stmt_bind_param($stmt, "s", $code);

    /* execute statement */
    mysqli_stmt_execute($stmt);

    printf("rows inserted: %d\n", mysqli_stmt_affected_rows($stmt));

    /* close statement */
    mysqli_stmt_close($stmt);
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
rows inserted: 17
```

See Also

- mysqli_stmt_num_rows()
- mysqli_prepare()

mysqli_stmt::attr_get

mysqli_stmt_attr_get

mysqli_stmt::attr_get -- mysqli_stmt_attr_get --

Description

int **mysqli_stmt_attr_get** (mysqli_stmt `$stmt`, int `$attr`)

Warning
This function is currently not documented; only its argument list is available.

mysqli_stmt::attr_set

mysqli_stmt_attr_set

mysqli_stmt::attr_set -- mysqli_stmt_attr_set --

Description

bool **mysqli_stmt_attr_set** (mysqli_stmt `$stmt`, int `$attr`, int `$mode`)

Warning
This function is currently not documented; only its argument list is available.

mysqli_stmt::bind_param

mysqli_stmt_bind_param

mysqli_stmt::bind_param -- mysqli_stmt_bind_param -- Binds variables to a prepared statement as parameters

Description

Object oriented style (method):

bool **mysqli_stmt::bind_param** (string `$types`, mixed `&$var1` [, mixed `&$...`])

Procedural style:

bool **mysqli_stmt_bind_param** (mysqli_stmt `$stmt`, string `$types`, mixed `&$var1` [, mixed `&$...`])

Bind variables for the parameter markers in the SQL statement that was passed to mysqli_prepare().

Note
If data size of a variable exceeds max. allowed packet size (max_allowed_packet), you have to specify *b* in `types` and use mysqli_stmt_send_long_data() to send the data in packets.

Parameters

`stmt`
Procedural style only: A statement identifier returned by mysqli_stmt_init().

`types`
A string that contains one or more characters which specify the types for the corresponding bind variables:

Type specification chars

Character	Description
i	corresponding variable has type integer
d	corresponding variable has type double
s	corresponding variable has type string

b	corresponding variable is a blob and will be sent in packets

var1
The number of variables and length of string *types* must match the parameters in the statement.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #72 - Object oriented style

```
<?php
$mysqli = new mysqli('localhost', 'my_user', 'my_password', 'world');

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$stmt = $mysqli->prepare("INSERT INTO CountryLanguage VALUES (?, ?, ?, ?)");
$stmt->bind_param('sssd', $code, $language, $official, $percent);

$code = 'DEU';
$language = 'Bavarian';
$official = "F";
$percent = 11.2;

/* execute prepared statement */
$stmt->execute();

printf("%d Row inserted.\n", $stmt->affected_rows);

/* close statement and connection */
$stmt->close();

/* Clean up table CountryLanguage */
$mysqli->query("DELETE FROM CountryLanguage WHERE Language='Bavarian'");
printf("%d Row deleted.\n", $mysqli->affected_rows);

/* close connection */
$mysqli->close();
?>
```

Example #73 - Procedural style

```
<?php
$link = mysqli_connect('localhost', 'my_user', 'my_password', 'world');

/* check connection */
if (!$link) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$stmt = mysqli_prepare($link, "INSERT INTO CountryLanguage VALUES (?, ?, ?,
?)");
mysqli_stmt_bind_param($stmt, 'sssd', $code, $language, $official, $percent);

$code = 'DEU';
$language = 'Bavarian';
$official = "F";
$percent = 11.2;

/* execute prepared statement */
mysqli_stmt_execute($stmt);

printf("%d Row inserted.\n", mysqli_stmt_affected_rows($stmt));

/* close statement and connection */
mysqli_stmt_close($stmt);

/* Clean up table CountryLanguage */
mysqli_query($link, "DELETE FROM CountryLanguage WHERE Language='Bavarian'");
printf("%d Row deleted.\n", mysqli_affected_rows($link));

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
1 Row inserted.
1 Row deleted.
```

See Also

- mysqli_stmt_bind_result()
- mysqli_stmt_execute()
- mysqli_stmt_fetch()
- mysqli_prepare()
- mysqli_stmt_send_long_data()
- mysqli_stmt_errno()
- mysqli_stmt_error()

mysqli_stmt::bind_result

mysqli_stmt_bind_result

mysqli_stmt::bind_result -- mysqli_stmt_bind_result -- Binds variables to a prepared statement for result storage

Description

Object oriented style (method):

bool **mysqli_stmt::bind_result** (mixed `&$var1` [, mixed `&$...`])

Procedural style:

bool **mysqli_stmt_bind_result** (mysqli_stmt `$stmt`, mixed `&$var1` [, mixed `&$...`])

Binds columns in the result set to variables.

When mysqli_stmt_fetch() is called to fetch data, the MySQL client/server protocol places the data for the bound columns into the specified variables *`var1, ...`*.

Note
Note that all columns must be bound after mysqli_stmt_execute() and prior to calling mysqli_stmt_fetch(). Depending on column types bound variables can silently change to the corresponding PHP type. A column can be bound or rebound at any time, even after a result set has been partially retrieved. The new binding takes effect the next time mysqli_stmt_fetch() is called.

Parameters

`stmt`
Procedural style only: A statement identifier returned by mysqli_stmt_init().

`var1`
The variable to be bound.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #74 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* prepare statement */
if ($stmt = $mysqli->prepare("SELECT Code, Name FROM Country ORDER BY Name LIMIT
5")) {
   $stmt->execute();

   /* bind variables to prepared statement */
   $stmt->bind_result($col1, $col2);

   /* fetch values */
   while ($stmt->fetch()) {
       printf("%s %s\n", $col1, $col2);
   }

   /* close statement */
   $stmt->close();
}
/* close connection */
$mysqli->close();

?>
```

Example #75 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (!$link) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* prepare statement */
if ($stmt = mysqli_prepare($link, "SELECT Code, Name FROM Country ORDER BY Name
LIMIT 5")) {
   mysqli_stmt_execute($stmt);

   /* bind variables to prepared statement */
   mysqli_stmt_bind_result($stmt, $col1, $col2);

   /* fetch values */
   while (mysqli_stmt_fetch($stmt)) {
       printf("%s %s\n", $col1, $col2);
   }
```

```
    /* close statement */
    mysqli_stmt_close($stmt);
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
AFG Afghanistan
ALB Albania
DZA Algeria
ASM American Samoa
AND Andorra
```

See Also

- mysqli_stmt_bind_param()
- mysqli_stmt_execute()
- mysqli_stmt_fetch()
- mysqli_prepare()
- mysqli_stmt_prepare()
- mysqli_stmt_init()
- mysqli_stmt_errno()
- mysqli_stmt_error()

mysqli_stmt::close

mysqli_stmt_close

mysqli_stmt::close -- mysqli_stmt_close -- Closes a prepared statement

Description

Object oriented style (method):

bool **mysqli_stmt::close** (void)

Procedural style:

bool **mysqli_stmt_close** (mysqli_stmt `$stmt`)

Closes a prepared statement. mysqli_stmt_close() also deallocates the statement handle. If the current statement has pending or unread results, this function cancels them so that the next query can be executed.

Parameters

`stmt`
Procedural style only: A statement identifier returned by mysqli_stmt_init().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- mysqli_prepare()

mysqli_stmt::data_seek

mysqli_stmt_data_seek

mysqli_stmt::data_seek -- mysqli_stmt_data_seek -- Seeks to an arbitray row in statement result set

Description

Object oriented style (method):

void **mysqli_stmt::data_seek** (int `$offset`)

Procedural style:

void **mysqli_stmt_data_seek** (mysqli_stmt `$stmt`, int `$offset`)

Seeks to an arbitrary result pointer in the statement result set.

Parameters

`stmt`
Procedural style only: A statement identifier returned by mysqli_stmt_init().

`offset`
Must be between zero and the total number of rows minus one (0.. mysqli_stmt_num_rows() - 1).

Return Values

No value is returned.

Examples

Example #76 - Object oriented style

```
<?php
/* Open a connection */
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}
```

```
$query = "SELECT Name, CountryCode FROM City ORDER BY Name";
if ($stmt = $mysqli->prepare($query)) {

   /* execute query */
   $stmt->execute();

   /* bind result variables */
   $stmt->bind_result($name, $code);

   /* store result */
   $stmt->store_result();

   /* seek to row no. 400 */
   $stmt->data_seek(399);

   /* fetch values */
   $stmt->fetch();

   printf ("City: %s  Countrycode: %s\n", $name, $code);

   /* close statement */
   $stmt->close();
}

/* close connection */
$mysqli->close();
?>
```

Example #77 - Procedural style

```
<?php
/* Open a connection */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER BY Name";
if ($stmt = mysqli_prepare($link, $query)) {

   /* execute query */
   mysqli_stmt_execute($stmt);

   /* bind result variables */
   mysqli_stmt_bind_result($stmt, $name, $code);

   /* store result */
   mysqli_stmt_store_result($stmt);

   /* seek to row no. 400 */
   mysqli_stmt_data_seek($stmt, 399);

   /* fetch values */
   mysqli_stmt_fetch($stmt);
```

```
    printf ("City: %s  Countrycode: %s\n", $name, $code);

    /* close statement */
    mysqli_stmt_close($stmt);
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
City: Benin City  Countrycode: NGA
```

See Also

- mysqli_prepare()

mysqli_stmt->errno

mysqli_stmt_errno

mysqli_stmt->errno -- mysqli_stmt_errno -- Returns the error code for the most recent statement call

Description

Object oriented style (property):

mysqli_stmt

int *errno*;

Procedural style :

int **mysqli_stmt_errno** (mysqli_stmt `$stmt`)

Returns the error code for the most recently invoked statement function that can succeed or fail.

Client error message numbers are listed in the MySQL *errmsg.h* header file, server error message numbers are listed in *mysqld_error.h*. In the MySQL source distribution you can find a complete list of error messages and error numbers in the file *Docs/mysqld_error.txt*.

Parameters

`stmt`
Procedural style only: A statement identifier returned by mysqli_stmt_init().

Return Values

An error code value. Zero means no error occurred.

Examples

Example #78 - Object oriented style

```
<?php
/* Open a connection */
```

```
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$mysqli->query("CREATE TABLE myCountry LIKE Country");
$mysqli->query("INSERT INTO myCountry SELECT * FROM Country");


$query = "SELECT Name, Code FROM myCountry ORDER BY Name";
if ($stmt = $mysqli->prepare($query)) {

   /* drop table */
   $mysqli->query("DROP TABLE myCountry");

   /* execute query */
   $stmt->execute();

   printf("Error: %d.\n", $stmt->errno);

   /* close statement */
   $stmt->close();
}

/* close connection */
$mysqli->close();
?>
```

Example #79 - Procedural style

```
<?php
/* Open a connection */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

mysqli_query($link, "CREATE TABLE myCountry LIKE Country");
mysqli_query($link, "INSERT INTO myCountry SELECT * FROM Country");


$query = "SELECT Name, Code FROM myCountry ORDER BY Name";
if ($stmt = mysqli_prepare($link, $query)) {

   /* drop table */
   mysqli_query($link, "DROP TABLE myCountry");

   /* execute query */
   mysqli_stmt_execute($stmt);

   printf("Error: %d.\n", mysqli_stmt_errno($stmt));
```

```
   /* close statement */
   mysqli_stmt_close($stmt);
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Error: 1146.
```

See Also

- mysqli_stmt_error()
- mysqli_stmt_sqlstate()

mysqli_stmt->error

mysqli_stmt_error

mysqli_stmt->error -- mysqli_stmt_error -- Returns a string description for last statement error

Description

Object oriented style (property):

mysqli_stmt

string *error*;

Procedural style:

string **mysqli_stmt_error** (mysqli_stmt `$stmt`)

Returns a containing the error message for the most recently invoked statement function that can succeed or fail.

Parameters

`stmt`
Procedural style only: A statement identifier returned by mysqli_stmt_init().

Return Values

A string that describes the error. An empty string if no error occurred.

Examples

Example #80 - Object oriented style

```
<?php
/* Open a connection */
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
```

```
   exit();
}

$mysqli->query("CREATE TABLE myCountry LIKE Country");
$mysqli->query("INSERT INTO myCountry SELECT * FROM Country");


$query = "SELECT Name, Code FROM myCountry ORDER BY Name";
if ($stmt = $mysqli->prepare($query)) {

   /* drop table */
   $mysqli->query("DROP TABLE myCountry");

   /* execute query */
   $stmt->execute();

   printf("Error: %s.\n", $stmt->error);

   /* close statement */
   $stmt->close();
}

/* close connection */
$mysqli->close();
?>
```

Example #81 - Procedural style

```
<?php
/* Open a connection */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

mysqli_query($link, "CREATE TABLE myCountry LIKE Country");
mysqli_query($link, "INSERT INTO myCountry SELECT * FROM Country");


$query = "SELECT Name, Code FROM myCountry ORDER BY Name";
if ($stmt = mysqli_prepare($link, $query)) {

   /* drop table */
   mysqli_query($link, "DROP TABLE myCountry");

   /* execute query */
   mysqli_stmt_execute($stmt);

   printf("Error: %s.\n", mysqli_stmt_error($stmt));

   /* close statement */
   mysqli_stmt_close($stmt);
}

/* close connection */
```

```
mysqli_close($link);
?>
```

The above example will output:

```
Error: Table 'world.myCountry' doesn't exist.
```

See Also

- mysqli_stmt_errno()
- mysqli_stmt_sqlstate()

mysqli_stmt->execute

mysqli_stmt_execute

mysqli_stmt->execute -- mysqli_stmt_execute -- Executes a prepared Query

Description

Object oriented style (method):

bool **mysqli_stmt::execute** (void)

Procedural style:

bool **mysqli_stmt_execute** (mysqli_stmt `$stmt`)

Executes a query that has been previously prepared using the mysqli_prepare() function. When executed any parameter markers which exist will automatically be replaced with the appropiate data.

If the statement is *UPDATE*, *DELETE*, or *INSERT*, the total number of affected rows can be determined by using the mysqli_stmt_affected_rows() function. Likewise, if the query yields a result set the mysqli_stmt_fetch() function is used.

Note
When using mysqli_stmt_execute(), the mysqli_stmt_fetch() function must be used to fetch the data prior to performing any additional queries.

Parameters

`stmt`
: Procedural style only: A statement identifier returned by mysqli_stmt_init().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #82 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$mysqli->query("CREATE TABLE myCity LIKE City");

/* Prepare an insert statement */
$query = "INSERT INTO myCity (Name, CountryCode, District) VALUES (?,?,?)";
$stmt = $mysqli->prepare($query);

$stmt->bind_param("sss", $val1, $val2, $val3);

$val1 = 'Stuttgart';
$val2 = 'DEU';
$val3 = 'Baden-Wuerttemberg';

/* Execute the statement */
$stmt->execute();

$val1 = 'Bordeaux';
$val2 = 'FRA';
$val3 = 'Aquitaine';

/* Execute the statement */
$stmt->execute();

/* close statement */
$stmt->close();

/* retrieve all rows from myCity */
$query = "SELECT Name, CountryCode, District FROM myCity";
if ($result = $mysqli->query($query)) {
   while ($row = $result->fetch_row()) {
       printf("%s (%s,%s)\n", $row[0], $row[1], $row[2]);
   }
   /* free result set */
   $result->close();
}

/* remove table */
$mysqli->query("DROP TABLE myCity");

/* close connection */
$mysqli->close();
?>
```

Example #83 - Procedural style

```
<?php
```

```
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

mysqli_query($link, "CREATE TABLE myCity LIKE City");

/* Prepare an insert statement */
$query = "INSERT INTO myCity (Name, CountryCode, District) VALUES (?,?,?)";
$stmt = mysqli_prepare($link, $query);

mysqli_stmt_bind_param($stmt, "sss", $val1, $val2, $val3);

$val1 = 'Stuttgart';
$val2 = 'DEU';
$val3 = 'Baden-Wuerttemberg';

/* Execute the statement */
mysqli_stmt_execute($stmt);

$val1 = 'Bordeaux';
$val2 = 'FRA';
$val3 = 'Aquitaine';

/* Execute the statement */
mysqli_stmt_execute($stmt);

/* close statement */
mysqli_stmt_close($stmt);

/* retrieve all rows from myCity */
$query = "SELECT Name, CountryCode, District FROM myCity";
if ($result = mysqli_query($link, $query)) {
   while ($row = mysqli_fetch_row($result)) {
       printf("%s (%s,%s)\n", $row[0], $row[1], $row[2]);
   }
   /* free result set */
   mysqli_free_result($result);
}

/* remove table */
mysqli_query($link, "DROP TABLE myCity");

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Stuttgart (DEU,Baden-Wuerttemberg)
Bordeaux (FRA,Aquitaine)
```

See Also

- mysqli_prepare()
- mysqli_stmt_bind_param()

mysqli_stmt::fetch

mysqli_stmt_fetch

mysqli_stmt::fetch -- mysqli_stmt_fetch -- Fetch results from a prepared statement into the bound variables

Description

Object oriented style (method):

bool **mysqli_stmt::fetch** (void)

Procedural style:

bool **mysqli_stmt_fetch** (mysqli_stmt `$stmt`)

Fetch the result from a prepared statement into the variables bound by mysqli_stmt_bind_result().

Note
Note that all columns must be bound by the application before calling mysqli_stmt_fetch().

Parameters

`stmt`
Procedural style only: A statement identifier returned by mysqli_stmt_init().

Return Values

Return Values

Value	Description
TRUE	Success. Data has been fetched
FALSE	Error occured
NULL	No more rows/data exists or data truncation occurred

Examples

Example #84 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 150,5";

if ($stmt = $mysqli->prepare($query)) {

   /* execute statement */
   $stmt->execute();

   /* bind result variables */
   $stmt->bind_result($name, $code);

   /* fetch values */
   while ($stmt->fetch()) {
       printf ("%s (%s)\n", $name, $code);
   }

   /* close statement */
   $stmt->close();
}

/* close connection */
$mysqli->close();
?>
```

Example #85 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 150,5";

if ($stmt = mysqli_prepare($link, $query)) {

   /* execute statement */
   mysqli_stmt_execute($stmt);

   /* bind result variables */
   mysqli_stmt_bind_result($stmt, $name, $code);
```

```
    /* fetch values */
    while (mysqli_stmt_fetch($stmt)) {
        printf ("%s (%s)\n", $name, $code);
    }

    /* close statement */
    mysqli_stmt_close($stmt);
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Rockford (USA)
Tallahassee (USA)
Salinas (USA)
Santa Clarita (USA)
Springfield (USA)
```

See Also

- mysqli_prepare()
- mysqli_stmt_errno()
- mysqli_stmt_error()
- mysqli_stmt_bind_result()

mysqli_stmt->field_count

mysqli_stmt_field_count

mysqli_stmt->field_count -- mysqli_stmt_field_count -- Returns the number of field in the given statement

Description

mysqli_stmt

int *field_count*;

int **mysqli_stmt_field_count** (mysqli_stmt `$stmt`)

Warning
This function is currently not documented; only its argument list is available.

stmt::free_result

mysqli_stmt_free_result

stmt::free_result -- mysqli_stmt_free_result -- Frees stored result memory for the given statement handle

Description

Object oriented style (method):

void **mysqli_stmt::free_result** (void)

Procedural style:

void **mysqli_stmt_free_result** (mysqli_stmt `$stmt`)

Frees the result memory associated with the statement, which was allocated by mysqli_stmt_store_result().

Parameters

`stmt`
: Procedural style only: A statement identifier returned by mysqli_stmt_init().

Return Values

No value is returned.

See Also

- mysqli_stmt_store_result()

mysqli_stmt::get_warnings

mysqli_stmt_get_warnings

mysqli_stmt::get_warnings -- mysqli_stmt_get_warnings --

Description

object **mysqli_stmt::get_warnings** (mysqli_stmt `$stmt`)

object **mysqli_stmt_get_warnings** (mysqli_stmt `$stmt`)

Warning
This function is currently not documented; only its argument list is available.

mysqli_stmt->insert_id

mysqli_stmt_insert_id

mysqli_stmt->insert_id -- mysqli_stmt_insert_id -- Get the ID generated from the previous INSERT operation

Description

mysqli_stmt

int *insert_id*;

mixed **mysqli_stmt_insert_id** (mysqli_stmt `$stmt`)

Warning
This function is currently not documented; only its argument list is available.

mysqli_stmt::num_rows

mysqli_stmt_num_rows

mysqli_stmt::num_rows -- mysqli_stmt_num_rows -- Return the number of rows in statements result set

Description

Object oriented style (property):

mysqli_stmt

int *num_rows*;

Procedural style :

int **mysqli_stmt_num_rows** (mysqli_stmt `$stmt`)

Returns the number of rows in the result set. The use of mysqli_stmt_num_rows() depends on whether or not you used mysqli_stmt_store_result() to buffer the entire result set in the statement handle.

If you use mysqli_stmt_store_result(), mysqli_stmt_num_rows() may be called immediately.

Parameters

`stmt`
Procedural style only: A statement identifier returned by mysqli_stmt_init().

Return Values

An integer representing the number of rows in result set.

Examples

Example #86 - Object oriented style

```
<?php
/* Open a connection */
```

```
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER BY Name LIMIT 20";
if ($stmt = $mysqli->prepare($query)) {

   /* execute query */
   $stmt->execute();

   /* store result */
   $stmt->store_result();

   printf("Number of rows: %d.\n", $stmt->num_rows);

   /* close statement */
   $stmt->close();
}

/* close connection */
$mysqli->close();
?>
```

Example #87 - Procedural style

```
<?php
/* Open a connection */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER BY Name LIMIT 20";
if ($stmt = mysqli_prepare($link, $query)) {

   /* execute query */
   mysqli_stmt_execute($stmt);

   /* store result */
   mysqli_stmt_store_result($stmt);

   printf("Number of rows: %d.\n", mysqli_stmt_num_rows($stmt));

   /* close statement */
   mysqli_stmt_close($stmt);
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Number of rows: 20.
```

See Also

- mysqli_stmt_affected_rows()
- mysqli_prepare()
- mysqli_stmt_store_result()

mysqli_stmt->param_count

mysqli_stmt_param_count

mysqli_stmt->param_count -- mysqli_stmt_param_count -- Returns the number of parameter for the given statement

Description

Object oriented style (property):

mysqli_stmt

int *param_count*;

Procedural style:

int **mysqli_stmt_param_count** (mysqli_stmt `$stmt`)

Returns the number of parameter markers present in the prepared statement.

Parameters

`stmt`
Procedural style only: A statement identifier returned by mysqli_stmt_init().

Return Values

Returns an integer representing the number of parameters.

Examples

Example #88 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
```

```
}

if ($stmt = $mysqli->prepare("SELECT Name FROM Country WHERE Name=? OR
Code=?")) {

   $marker = $stmt->param_count;
   printf("Statement has %d markers.\n", $marker);

   /* close statement */
   $stmt->close();
}

/* close connection */
$mysqli->close();
?>
```

Example #89 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

if ($stmt = mysqli_prepare($link, "SELECT Name FROM Country WHERE Name=? OR
Code=?")) {

   $marker = mysqli_stmt_param_count($stmt);
   printf("Statement has %d markers.\n", $marker);

   /* close statement */
   mysqli_stmt_close($stmt);
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Statement has 2 markers.
```

See Also

- mysqli_prepare()

mysqli_stmt::prepare

mysqli_stmt_prepare

mysqli_stmt::prepare -- mysqli_stmt_prepare -- Prepare a SQL statement for execution

Description

Object oriented style (method)

mixed **mysqli_stmt::prepare** (string `$query`)

Procedure style:

bool **mysqli_stmt_prepare** (mysqli_stmt `$stmt`, string `$query`)

Prepares the SQL query pointed to by the null-terminated string query.

The parameter markers must be bound to application variables using mysqli_stmt_bind_param() and/or mysqli_stmt_bind_result() before executing the statement or fetching rows.

Parameters

`stmt`
Procedural style only: A statement identifier returned by mysqli_stmt_init().

`query`
The query, as a string. It must consist of a single SQL statement. You can include one or more parameter markers in the SQL statement by embedding question mark (*?*) characters at the appropriate positions.

Note
You should not add a terminating semicolon or \g to the statement.

Note
The markers are legal only in certain places in SQL statements. For example, they are allowed in the VALUES() list of an INSERT statement (to specify column values for a row), or in a comparison with a column in a WHERE clause to specify a comparison value. However, they are not allowed for identifiers (such as table or column names), in the select list that names the columns to be returned by a SELECT statement), or

to specify both operands of a binary operator such as the = equal sign. The latter restriction is necessary because it would be impossible to determine the parameter type. In general, parameters are legal only in Data Manipulation Languange (DML) statements, and not in Data Definition Language (DDL) statements.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #90 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$city = "Amersfoort";

/* create a prepared statement */
$stmt =  $mysqli->stmt_init();
if ($stmt->prepare("SELECT District FROM City WHERE Name=?")) {

   /* bind parameters for markers */
   $stmt->bind_param("s", $city);

   /* execute query */
   $stmt->execute();

   /* bind result variables */
   $stmt->bind_result($district);

   /* fetch value */
   $stmt->fetch();

   printf("%s is in district %s\n", $city, $district);

   /* close statement */
   $stmt->close();
}

/* close connection */
$mysqli->close();
?>
```

Example #91 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$city = "Amersfoort";

/* create a prepared statement */
$stmt = mysqli_stmt_init($link);
if (mysqli_stmt_prepare($stmt, 'SELECT District FROM City WHERE Name=?')) {

   /* bind parameters for markers */
   mysqli_stmt_bind_param($stmt, "s", $city);

   /* execute query */
   mysqli_stmt_execute($stmt);

   /* bind result variables */
   mysqli_stmt_bind_result($stmt, $district);

   /* fetch value */
   mysqli_stmt_fetch($stmt);

   printf("%s is in district %s\n", $city, $district);

   /* close statement */
   mysqli_stmt_close($stmt);
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Amersfoort is in district Utrecht
```

See Also

mysqli_stmt_init(), mysqli_stmt_execute(), mysqli_stmt_fetch(), mysqli_stmt_bind_param() , mysqli_stmt_bind_result() mysqli_stmt_close().

mysqli_stmt::reset

mysqli_stmt_reset

mysqli_stmt::reset -- mysqli_stmt_reset -- Resets a prepared statement

Description

Object oriented style (method):

bool **mysqli_stmt::reset** (void)

Procedural style:

bool **mysqli_stmt_reset** (mysqli_stmt `$stmt`)

Resets a prepared statement on client and server to state after prepare.

For now this is mainly used to reset data sent with mysqli_stmt_send_long_data().

To prepare a statement with another query use function mysqli_stmt_prepare().

Parameters

`stmt`
Procedural style only: A statement identifier returned by mysqli_stmt_init().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- mysqli_prepare()

mysqli_stmt::result_metadata

mysqli_stmt_result_metadata

mysqli_stmt::result_metadata -- mysqli_stmt_result_metadata -- Returns result set metadata from a prepared statement

Description

Object oriented style (method):

mysqli_result **mysqli_stmt::result_metadata** (void)

Procedural style:

mysqli_result **mysqli_stmt_result_metadata** (mysqli_stmt `$stmt`)

If a statement passed to mysqli_prepare() is one that produces a result set, mysqli_stmt_result_metadata() returns the result object that can be used to process the meta information such as total number of fields and individual field information.

Note

This result set pointer can be passed as an argument to any of the field-based functions that process result set metadata, such as:

- mysqli_num_fields()
- mysqli_fetch_field()
- mysqli_fetch_field_direct()
- mysqli_fetch_fields()
- mysqli_field_count()
- mysqli_field_seek()
- mysqli_field_tell()
- mysqli_free_result()

The result set structure should be freed when you are done with it, which you can do by passing it to mysqli_free_result()

Note

The result set returned by mysqli_stmt_result_metadata() contains only metadata. It does not contain any row results. The rows are obtained by using the statement handle with mysqli_stmt_fetch().

Parameters

stmt
: Procedural style only: A statement identifier returned by mysqli_stmt_init().

Return Values

Returns a result object or **FALSE** if an error occured.

Examples

Example #92 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "test");

$mysqli->query("DROP TABLE IF EXISTS friends");
$mysqli->query("CREATE TABLE friends (id int, name varchar(20))");

$mysqli->query("INSERT INTO friends VALUES (1,'Hartmut'), (2, 'Ulf')");

$stmt = $mysqli->prepare("SELECT id, name FROM friends");
$stmt->execute();

/* get resultset for metadata */
$result = $stmt->result_metadata();

/* retrieve field information from metadata result set */
$field = $result->fetch_field();

printf("Fieldname: %s\n", $field->name);

/* close resultset */
$result->close();

/* close connection */
$mysqli->close();
?>
```

Example #93 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "test");

mysqli_query($link, "DROP TABLE IF EXISTS friends");
mysqli_query($link, "CREATE TABLE friends (id int, name varchar(20))");

mysqli_query($link, "INSERT INTO friends VALUES (1,'Hartmut'), (2, 'Ulf')");

$stmt = mysqli_prepare($link, "SELECT id, name FROM friends");
mysqli_stmt_execute($stmt);

/* get resultset for metadata */
$result = mysqli_stmt_result_metadata($stmt);

/* retrieve field information from metadata result set */
$field = mysqli_fetch_field($result);

printf("Fieldname: %s\n", $field->name);

/* close resultset */
mysqli_free_result($result);

/* close connection */
mysqli_close($link);
?>
```

See Also

- mysqli_prepare()
- mysqli_free_result()

mysqli_stmt::send_long_data

mysqli_stmt_send_long_data

mysqli_stmt::send_long_data -- mysqli_stmt_send_long_data -- Send data in blocks

Description

Object oriented style (method)

bool **mysqli_stmt::send_long_data** (int `$param_nr`, string `$data`)

Procedural style:

bool **mysqli_stmt_send_long_data** (mysqli_stmt `$stmt`, int `$param_nr`, string `$data`)

Allows to send parameter data to the server in pieces (or chunks), e.g. if the size of a blob exceeds the size of *max_allowed_packet*. This function can be called multiple times to send the parts of a character or binary data value for a column, which must be one of the TEXT or BLOB datatypes.

Parameters

`stmt`
Procedural style only: A statement identifier returned by mysqli_stmt_init().

`param_nr`
Indicates which parameter to associate the data with. Parameters are numbered beginning with 0.

`data`
A string containing data to be sent.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #94 - Object oriented style

```
<?php
$stmt = $mysqli->prepare("INSERT INTO messages (message) VALUES (?)");
$null = NULL;
$stmt->bind_param("b", $null);
```

```
$fp = fopen("messages.txt", "r");
while (!feof($fp)) {
   $stmt->send_long_data(0, fread($fp, 8192));
}
fclose($fp);
$stmt->execute();
?>
```

See Also

- mysqli_prepare()
- mysqli_stmt_bind_param()

mysqli_stmt::sqlstate

mysqli_stmt_sqlstate

mysqli_stmt::sqlstate -- mysqli_stmt_sqlstate -- Returns SQLSTATE error from previous statement operation

Description

Object oriented style (property):

mysqli_stmt

string *sqlstate*;

Procedural style:

string **mysqli_stmt_sqlstate** (mysqli_stmt `$stmt`)

Returns a string containing the SQLSTATE error code for the most recently invoked prepared statement function that can succeed or fail. The error code consists of five characters. *'00000'* means no error. The values are specified by ANSI SQL and ODBC. For a list of possible values, see » http://dev.mysql.com/doc/mysql/en/error-handling.html.

Parameters

`stmt`
Procedural style only: A statement identifier returned by mysqli_stmt_init().

Return Values

Returns a string containing the SQLSTATE error code for the last error. The error code consists of five characters. *'00000'* means no error.

Notes

Note
Note that not all MySQL errors are yet mapped to SQLSTATE's. The value *HY000* (general error) is used for unmapped errors.

Examples

Example #95 - Object oriented style

```
<?php
/* Open a connection */
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$mysqli->query("CREATE TABLE myCountry LIKE Country");
$mysqli->query("INSERT INTO myCountry SELECT * FROM Country");


$query = "SELECT Name, Code FROM myCountry ORDER BY Name";
if ($stmt = $mysqli->prepare($query)) {

   /* drop table */
   $mysqli->query("DROP TABLE myCountry");

   /* execute query */
   $stmt->execute();

   printf("Error: %s.\n", $stmt->sqlstate);

   /* close statement */
   $stmt->close();
}

/* close connection */
$mysqli->close();
?>
```

Example #96 - Procedural style

```
<?php
/* Open a connection */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

mysqli_query($link, "CREATE TABLE myCountry LIKE Country");
mysqli_query($link, "INSERT INTO myCountry SELECT * FROM Country");


$query = "SELECT Name, Code FROM myCountry ORDER BY Name";
if ($stmt = mysqli_prepare($link, $query)) {
```

```
    /* drop table */
    mysqli_query($link, "DROP TABLE myCountry");

    /* execute query */
    mysqli_stmt_execute($stmt);

    printf("Error: %s.\n", mysqli_stmt_sqlstate($stmt));

    /* close statement */
    mysqli_stmt_close($stmt);
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Error: 42S02.
```

See Also

- mysqli_stmt_errno()
- mysqli_stmt_error()

mysqli_stmt::store_result

mysqli_stmt_store_result

mysqli_stmt::store_result -- mysqli_stmt_store_result -- Transfers a result set from a prepared statement

Description

Object oriented style (method):

bool **mysqli_stmt::store_result** (void)

Procedural style:

bool **mysqli_stmt_store_result** (mysqli_stmt `$stmt`)

You must call mysqli_stmt_store_result() for every query that successfully produces a result set (*SELECT, SHOW, DESCRIBE, EXPLAIN*), and only if you want to buffer the complete result set by the client, so that the subsequent mysqli_stmt_fetch() call returns buffered data.

Note
It is unnecessary to call mysqli_stmt_store_result() for other queries, but if you do, it will not harm or cause any notable performance in all cases. You can detect whether the query produced a result set by checking if mysqli_stmt_result_metadata() returns NULL.

Parameters

`stmt`
Procedural style only: A statement identifier returned by mysqli_stmt_init().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #97 - Object oriented style

```
<?php
/* Open a connection */
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER BY Name LIMIT 20";
if ($stmt = $mysqli->prepare($query)) {

   /* execute query */
   $stmt->execute();

   /* store result */
   $stmt->store_result();

   printf("Number of rows: %d.\n", $stmt->num_rows);

   /* free result */
   $stmt->free_result();

   /* close statement */
   $stmt->close();
}

/* close connection */
$mysqli->close();
?>
```

Example #98 - Procedural style

```
<?php
/* Open a connection */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER BY Name LIMIT 20";
if ($stmt = mysqli_prepare($link, $query)) {

   /* execute query */
   mysqli_stmt_execute($stmt);

   /* store result */
   mysqli_stmt_store_result($stmt);

   printf("Number of rows: %d.\n", mysqli_stmt_num_rows($stmt));
```

```
    /* free result */
    mysqli_stmt_free_result($stmt);

    /* close statement */
    mysqli_stmt_close($stmt);
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Number of rows: 20.
```

See Also

- mysqli_prepare()
- mysqli_stmt_result_metadata()
- mysqli_stmt_fetch()

The MySQLi_Result class

Introduction

Represents the result set obtained from a query against the database.

Class synopsis

MySQLi_Result

MySQLi_Result {

/* Properties */

int *current_field*;

int *field_count*;

array *lengths*;

int *num_rows*;

/* Methods */

int **mysqli_field_tell** (mysqli_result `$result`)

bool **mysqli_result::data_seek** (int `$offset`)

mixed **mysqli_result::fetch_array** ([int `$resulttype`])

array **mysqli_result::fetch_assoc** (void)

object **mysqli_result::fetch_field_direct** (int `$fieldnr`)

object **mysqli_result::fetch_field** (void)

array **mysqli_result::fetch_fields** (void)

object **mysqli_result::fetch_object** ([string `$class_name` [, array `$params`]])

mixed **mysqli_result::fetch_row** (void)

int **mysqli_num_fields** (mysqli_result `$result`)

```
    bool mysqli_result::field_seek ( int $fieldnr )

    void mysqli_result::free ( void )

    array mysqli_fetch_lengths ( mysqli_result $result )

    int mysqli_num_rows ( mysqli_result $result )
}
```

mysqli_result->current_field

mysqli_field_tell

mysqli_result->current_field -- mysqli_field_tell -- Get current field offset of a result pointer

Description

Object oriented style (property):

mysqli_result

int *current_field*;

Procedural style:

int **mysqli_field_tell** (mysqli_result `$result`)

Returns the position of the field cursor used for the last mysqli_fetch_field() call. This value can be used as an argument to mysqli_field_seek().

Parameters

result
Procedural style only: A result set identifier returned by mysqli_query(), mysqli_store_result() or mysqli_use_result().

Return Values

Returns current offset of field cursor.

Examples

Example #99 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
```

```
   exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";

if ($result = $mysqli->query($query)) {

   /* Get field information for all columns */
   while ($finfo = $result->fetch_field()) {

       /* get fieldpointer offset */
       $currentfield = $result->current_field;

       printf("Column %d:\n", $currentfield);
       printf("Name:     %s\n", $finfo->name);
       printf("Table:    %s\n", $finfo->table);
       printf("max. Len: %d\n", $finfo->max_length);
       printf("Flags:    %d\n", $finfo->flags);
       printf("Type:     %d\n\n", $finfo->type);
   }
   $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Example #100 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";

if ($result = mysqli_query($link, $query)) {

   /* Get field information for all fields */
   while ($finfo = mysqli_fetch_field($result)) {

       /* get fieldpointer offset */
       $currentfield = mysqli_field_tell($result);

       printf("Column %d:\n", $currentfield);
       printf("Name:     %s\n", $finfo->name);
       printf("Table:    %s\n", $finfo->table);
       printf("max. Len: %d\n", $finfo->max_length);
       printf("Flags:    %d\n", $finfo->flags);
       printf("Type:     %d\n\n", $finfo->type);
   }
   mysqli_free_result($result);
}
```

```
/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Column 1:
Name:     Name
Table:    Country
max. Len: 11
Flags:    1
Type:     254

Column 2:
Name:     SurfaceArea
Table:    Country
max. Len: 10
Flags:    32769
Type:     4
```

See Also

- mysqli_fetch_field()
- mysqli_field_seek()

mysqli_result::data_seek

mysqli_data_seek

mysqli_result::data_seek -- mysqli_data_seek -- Adjusts the result pointer to an arbitary row in the result

Description

Object oriented style (method):

bool **mysqli_result::data_seek** (int `$offset`)

Procedural style:

bool **mysqli_data_seek** (mysqli_result `$result`, int `$offset`)

The mysqli_data_seek() function seeks to an arbitrary result pointer specified by the *`offset`* in the result set.

Parameters

`result`
Procedural style only: A result set identifier returned by mysqli_query(), mysqli_store_result() or mysqli_use_result().

`offset`
The field offset. Must be between zero and the total number of rows minus one (0.. mysqli_num_rows() - 1).

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
This function can only be used with buffered results attained from the use of the mysqli_store_result() or mysqli_query() functions.

Examples

Example #101 - Object oriented style

```
<?php
/* Open a connection */
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER BY Name";
if ($result = $mysqli->query( $query)) {

   /* seek to row no. 400 */
   $result->data_seek(399);

   /* fetch row */
   $row = $result->fetch_row();

   printf ("City: %s  Countrycode: %s\n", $row[0], $row[1]);

   /* free result set*/
   $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Example #102 - Procedural style

```
<?php
/* Open a connection */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (!$link) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER BY Name";

if ($result = mysqli_query($link, $query)) {

   /* seek to row no. 400 */
   mysqli_data_seek($result, 399);

   /* fetch row */
   $row = mysqli_fetch_row($result);

   printf ("City: %s  Countrycode: %s\n", $row[0], $row[1]);

   /* free result set*/
```

```
    mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
City: Benin City  Countrycode: NGA
```

See Also

- mysqli_store_result()
- mysqli_fetch_row()
- mysqli_fetch_array()
- mysqli_fetch_assoc()
- mysqli_fetch_object()
- mysqli_query()
- mysqli_num_rows()

mysqli_result::fetch_array

mysqli_fetch_array

mysqli_result::fetch_array -- mysqli_fetch_array -- Fetch a result row as an associative, a numeric array, or both

Description

Object oriented style (method):

mixed **mysqli_result::fetch_array** ([int `$resulttype`])

Procedural style:

mixed **mysqli_fetch_array** (mysqli_result `$result` [, int `$resulttype`])

Returns an array that corresponds to the fetched row or **NULL** if there are no more rows for the resultset represented by the *`result`* parameter.

mysqli_fetch_array() is an extended version of the mysqli_fetch_row() function. In addition to storing the data in the numeric indices of the result array, the mysqli_fetch_array() function can also store the data in associative indices, using the field names of the result set as keys.

Note
Field names returned by this function are *case-sensitive*.

Note
This function sets NULL fields to the PHP **NULL** value.

If two or more columns of the result have the same field names, the last column will take precedence and overwrite the earlier data. In order to access multiple columns with the same name, the numerically indexed version of the row must be used.

Parameters

`result`
: Procedural style only: A result set identifier returned by mysqli_query(), mysqli_store_result() or mysqli_use_result().

resulttype
: This optional parameter is a constant indicating what type of array should be produced from the current row data. The possible values for this parameter are the constants **MYSQLI_ASSOC**, **MYSQLI_NUM**, or **MYSQLI_BOTH**. Defaults to **MYSQLI_BOTH**. By using the **MYSQLI_ASSOC** constant this function will behave identically to the mysqli_fetch_assoc(), while **MYSQLI_NUM** will behave identically to the mysqli_fetch_row() function. The final option **MYSQLI_BOTH** will create a single array with the attributes of both.

Return Values

Returns an array of strings that corresponds to the fetched row or **NULL** if there are no more rows in resultset.

Examples

Example #103 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID LIMIT 3";
$result = $mysqli->query($query);

/* numeric array */
$row = $result->fetch_array(MYSQLI_NUM);
printf ("%s (%s)\n", $row[0], $row[1]);

/* associative array */
$row = $result->fetch_array(MYSQLI_ASSOC);
printf ("%s (%s)\n", $row["Name"], $row["CountryCode"]);

/* associative and numeric array */
$row = $result->fetch_array(MYSQLI_BOTH);
printf ("%s (%s)\n", $row[0], $row["CountryCode"]);

/* free result set */
$result->close();

/* close connection */
$mysqli->close();
?>
```

Example #104 - Procedural style

```
<?php
```

```
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID LIMIT 3";
$result = mysqli_query($link, $query);

/* numeric array */
$row = mysqli_fetch_array($result, MYSQLI_NUM);
printf ("%s (%s)\n", $row[0], $row[1]);

/* associative array */
$row = mysqli_fetch_array($result, MYSQLI_ASSOC);
printf ("%s (%s)\n", $row["Name"], $row["CountryCode"]);

/* associative and numeric array */
$row = mysqli_fetch_array($result, MYSQLI_BOTH);
printf ("%s (%s)\n", $row[0], $row["CountryCode"]);

/* free result set */
mysqli_free_result($result);

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Kabul (AFG)
Qandahar (AFG)
Herat (AFG)
```

See Also

- mysqli_fetch_assoc()
- mysqli_fetch_row()
- mysqli_fetch_object()
- mysqli_query()
- mysqli_data_seek()

mysqli_result::fetch_assoc

mysqli_fetch_assoc

mysqli_result::fetch_assoc -- mysqli_fetch_assoc -- Fetch a result row as an associative array

Description

Object oriented style (method):

array **mysqli_result::fetch_assoc** (void)

Procedural style:

array **mysqli_fetch_assoc** (mysqli_result `$result`)

Returns an associative array that corresponds to the fetched row or **NULL** if there are no more rows.

Note
Field names returned by this function are *case-sensitive*.

Note
This function sets NULL fields to the PHP **NULL** value.

Parameters

`result`
Procedural style only: A result set identifier returned by mysqli_query(), mysqli_store_result() or mysqli_use_result().

Return Values

Returns an associative array of strings representing the fetched row in the result set, where each key in the array represents the name of one of the result set's columns or **NULL** if there are no more rows in resultset.

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you either need to access

the result with numeric indices by using mysqli_fetch_row() or add alias names.

Examples

Example #105 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 50,5";

if ($result = $mysqli->query($query)) {

   /* fetch associative array */
   while ($row = $result->fetch_assoc()) {
       printf ("%s (%s)\n", $row["Name"], $row["CountryCode"]);
   }

   /* free result set */
   $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Example #106 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 50,5";

if ($result = mysqli_query($link, $query)) {

   /* fetch associative array */
   while ($row = mysqli_fetch_assoc($result)) {
       printf ("%s (%s)\n", $row["Name"], $row["CountryCode"]);
   }

   /* free result set */
   mysqli_free_result($result);
}
```

```
/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Pueblo (USA)
Arvada (USA)
Cape Coral (USA)
Green Bay (USA)
Santa Clara (USA)
```

See Also

- mysqli_fetch_array()
- mysqli_fetch_row()
- mysqli_fetch_object()
- mysqli_query()
- mysqli_data_seek()

mysqli_result::fetch_field_direct

mysqli_fetch_field_direct

mysqli_result::fetch_field_direct -- mysqli_fetch_field_direct -- Fetch meta-data for a single field

Description

Object oriented style (method):

object **mysqli_result::fetch_field_direct** (int `$fieldnr`)

Procedural style:

object **mysqli_fetch_field_direct** (mysqli_result `$result`, int `$fieldnr`)

Returns an object which contains field definition informations from specified resultset.

Parameters

`result`
: Procedural style only: A result set identifier returned by mysqli_query(), mysqli_store_result() or mysqli_use_result().

`fieldnr`
: The field number. This value must be in the range from *0* to *number of fields - 1*.

Return Values

Returns an object which contains field definition information or **FALSE** if no field information for specified *fieldnr* is available.

Object attributes

Attribute	Description
name	The name of the column
orgname	Original column name if an alias was specified
table	The name of the table this field belongs to (if not calculated)

orgtable	Original table name if an alias was specified
def	The default value for this field, represented as a string
max_length	The maximum width of the field for the result set.
length	The width of the field, as specified in the tabl definition.
charsetnr	The character set number for the field.
flags	An integer representing the bit-flags for the field.
type	The data type used for this field
decimals	The number of decimals used (for integer fields)

Examples

Example #107 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Name LIMIT 5";

if ($result = $mysqli->query($query)) {

   /* Get field information for column 'SurfaceArea' */
   $finfo = $result->fetch_field_direct(1);

   printf("Name:     %s\n", $finfo->name);
   printf("Table:    %s\n", $finfo->table);
   printf("max. Len: %d\n", $finfo->max_length);
   printf("Flags:    %d\n", $finfo->flags);
   printf("Type:     %d\n", $finfo->type);

   $result->close();
}

/* close connection */
$mysqli->close();
```

```
?>
```

Example #108 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Name LIMIT 5";

if ($result = mysqli_query($link, $query)) {

   /* Get field information for column 'SurfaceArea' */
   $finfo = mysqli_fetch_field_direct($result, 1);

   printf("Name:     %s\n", $finfo->name);
   printf("Table:    %s\n", $finfo->table);
   printf("max. Len: %d\n", $finfo->max_length);
   printf("Flags:    %d\n", $finfo->flags);
   printf("Type:     %d\n", $finfo->type);

   mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Name:     SurfaceArea
Table:    Country
max. Len: 10
Flags:    32769
Type:     4
```

See Also

- mysqli_num_fields()
- mysqli_fetch_field()
- mysqli_fetch_fields()

mysqli_result::fetch_field

mysqli_fetch_field

mysqli_result::fetch_field -- mysqli_fetch_field -- Returns the next field in the result set

Description

Object oriented style (method):

object **mysqli_result::fetch_field** (void)

Procedural style:

object **mysqli_fetch_field** (mysqli_result `$result`)

Returns the definition of one column of a result set as an object. Call this function repeatedly to retrieve information about all columns in the result set.

Parameters

`result`
Procedural style only: A result set identifier returned by mysqli_query(), mysqli_store_result() or mysqli_use_result().

Return Values

Returns an object which contains field definition information or **FALSE** if no field information is available.

Object properties

Property	Description
name	The name of the column
orgname	Original column name if an alias was specified
table	The name of the table this field belongs to (if not calculated)
orgtable	Original table name if an alias was specified

def	The default value for this field, represented as a string
max_length	The maximum width of the field for the result set.
length	The width of the field, as specified in the table definition.
charsetnr	The character set number for the field.
flags	An integer representing the bit-flags for the field.
type	The data type used for this field
decimals	The number of decimals used (for integer fields)

Examples

Example #109 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";

if ($result = $mysqli->query($query)) {

   /* Get field information for all columns */
   while ($finfo = $result->fetch_field()) {

       printf("Name:     %s\n", $finfo->name);
       printf("Table:    %s\n", $finfo->table);
       printf("max. Len: %d\n", $finfo->max_length);
       printf("Flags:    %d\n", $finfo->flags);
       printf("Type:     %d\n\n", $finfo->type);
   }
   $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Example #110 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";

if ($result = mysqli_query($link, $query)) {

   /* Get field information for all fields */
   while ($finfo = mysqli_fetch_field($result)) {

       printf("Name:     %s\n", $finfo->name);
       printf("Table:    %s\n", $finfo->table);
       printf("max. Len: %d\n", $finfo->max_length);
       printf("Flags:    %d\n", $finfo->flags);
       printf("Type:     %d\n\n", $finfo->type);
   }
   mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Name:     Name
Table:    Country
max. Len: 11
Flags:    1
Type:     254

Name:     SurfaceArea
Table:    Country
max. Len: 10
Flags:    32769
Type:     4
```

See Also

- mysqli_num_fields()
- mysqli_fetch_field_direct()
- mysqli_fetch_fields()
- mysqli_field_seek()

mysqli_result::fetch_fields

mysqli_fetch_fields

mysqli_result::fetch_fields -- mysqli_fetch_fields -- Returns an array of objects representing the fields in a result set

Description

Object oriented style (method):

array **mysqli_result::fetch_fields** (void)

Procedural Style:

array **mysqli_fetch_fields** (mysqli_result `$result`)

This function serves an identical purpose to the mysqli_fetch_field() function with the single difference that, instead of returning one object at a time for each field, the columns are returned as an array of objects.

Parameters

`result`
Procedural style only: A result set identifier returned by mysqli_query(), mysqli_store_result() or mysqli_use_result().

Return Values

Returns an array of objects which contains field definition information or **FALSE** if no field information is available.

Object properties

Property	Description
name	The name of the column
orgname	Original column name if an alias was specified
table	The name of the table this field belongs to (if not calculated)

orgtable	Original table name if an alias was specified
def	The default value for this field, represented as a string
max_length	The maximum width of the field for the result set.
length	The width of the field, as specified in the table definition.
charsetnr	The character set number for the field.
flags	An integer representing the bit-flags for the field.
type	The data type used for this field
decimals	The number of decimals used (for integer fields)

Examples

Example #111 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";

if ($result = $mysqli->query($query)) {

   /* Get field information for all columns */
   $finfo = $result->fetch_fields();

   foreach ($finfo as $val) {
       printf("Name:     %s\n", $val->name);
       printf("Table:    %s\n", $val->table);
       printf("max. Len: %d\n", $val->max_length);
       printf("Flags:    %d\n", $val->flags);
       printf("Type:     %d\n\n", $val->type);
   }
   $result->close();
}

/* close connection */
```

```
$mysqli->close();
?>
```

Example #112 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";

if ($result = mysqli_query($link, $query)) {

   /* Get field information for all columns */
   $finfo = mysqli_fetch_fields($result);

   foreach ($finfo as $val) {
       printf("Name:     %s\n", $val->name);
       printf("Table:    %s\n", $val->table);
       printf("max. Len: %d\n", $val->max_length);
       printf("Flags:    %d\n", $val->flags);
       printf("Type:     %d\n\n", $val->type);
   }
   mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Name:     Name
Table:    Country
max. Len: 11
Flags:    1
Type:     254

Name:     SurfaceArea
Table:    Country
max. Len: 10
Flags:    32769
Type:     4
```

See Also

- mysqli_num_fields()
- mysqli_fetch_field_direct()

- mysqli_fetch_field()

mysqli_result::fetch_object

mysqli_fetch_object

mysqli_result::fetch_object -- mysqli_fetch_object -- Returns the current row of a result set as an object

Description

Object oriented style (method):

object **mysqli_result::fetch_object** ([string `$class_name` [, array `$params`]])

Procedural style:

object **mysqli_fetch_object** (mysqli_result `$result` [, string `$class_name` [, array `$params`]])

The mysqli_fetch_object() will return the current row result set as an object where the attributes of the object represent the names of the fields found within the result set.

Parameters

`result`
Procedural style only: A result set identifier returned by mysqli_query(), mysqli_store_result() or mysqli_use_result().

`class_name`

`params`

Return Values

Returns an object with string properties that corresponds to the fetched row or **NULL** if there are no more rows in resultset.

Note
Field names returned by this function are *case-sensitive*.

Note

This function sets NULL fields to the PHP **NULL** value.

ChangeLog

Version	Description
5.0.0	Added the ability to return as a different object.

Examples

Example #113 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 50,5";

if ($result = $mysqli->query($query)) {

   /* fetch object array */
   while ($obj = $result->fetch_object()) {
       printf ("%s (%s)\n", $obj->Name, $obj->CountryCode);
   }

   /* free result set */
   $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Example #114 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");
```

```
/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 50,5";

if ($result = mysqli_query($link, $query)) {

   /* fetch associative array */
   while ($obj = mysqli_fetch_object($result)) {
       printf ("%s (%s)\n", $obj->Name, $obj->CountryCode);
   }

   /* free result set */
   mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Pueblo (USA)
Arvada (USA)
Cape Coral (USA)
Green Bay (USA)
Santa Clara (USA)
```

See Also

- mysqli_fetch_array()
- mysqli_fetch_assoc()
- mysqli_fetch_row()
- mysqli_query()
- mysqli_data_seek()

mysqli_result::fetch_row

mysqli_fetch_row

mysqli_result::fetch_row -- mysqli_fetch_row -- Get a result row as an enumerated array

Description

Object oriented style (method):

mixed **mysqli_result::fetch_row** (void)

Procedural style:

mixed **mysqli_fetch_row** (mysqli_result `$result`)

Fetches one row of data from the result set and returns it as an enumerated array, where each column is stored in an array offset starting from 0 (zero). Each subsequent call to this function will return the next row within the result set, or **NULL** if there are no more rows.

Parameters

`result`
Procedural style only: A result set identifier returned by mysqli_query(), mysqli_store_result() or mysqli_use_result().

Return Values

mysqli_fetch_row() returns an array of strings that corresponds to the fetched row or **NULL** if there are no more rows in result set.

Note
This function sets NULL fields to the PHP **NULL** value.

Examples

Example #115 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
```

```
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 50,5";

if ($result = $mysqli->query($query)) {

   /* fetch object array */
   while ($row = $result->fetch_row()) {
       printf ("%s (%s)\n", $row[0], $row[1]);
   }

   /* free result set */
   $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Example #116 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 50,5";

if ($result = mysqli_query($link, $query)) {

   /* fetch associative array */
   while ($row = mysqli_fetch_row($result)) {
       printf ("%s (%s)\n", $row[0], $row[1]);
   }

   /* free result set */
   mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Pueblo (USA)
Arvada (USA)
Cape Coral (USA)
Green Bay (USA)
```

```
Santa Clara (USA)
```

See Also

- mysqli_fetch_array()
- mysqli_fetch_assoc()
- mysqli_fetch_object()
- mysqli_query()
- mysqli_data_seek()

mysqli_result->field_count

mysqli_num_fields

mysqli_result->field_count -- mysqli_num_fields -- Get the number of fields in a result

Description

Object oriented style (property):

mysqli_result

int *field_count*;

Procedural style:

int **mysqli_num_fields** (mysqli_result `$result`)

Returns the number of fields from specified result set.

Parameters

`result`
Procedural style only: A result set identifier returned by mysqli_query(), mysqli_store_result() or mysqli_use_result().

Return Values

The number of fields from a result set.

Examples

Example #117 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
```

```
}

if ($result = $mysqli->query("SELECT * FROM City ORDER BY ID LIMIT 1")) {

   /* determine number of fields in result set */
   $field_cnt = $result->field_count;

   printf("Result set has %d fields.\n", $field_cnt);

   /* close result set */
   $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Example #118 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

if ($result = mysqli_query($link, "SELECT * FROM City ORDER BY ID LIMIT 1"))
{

   /* determine number of fields in result set */
   $field_cnt = mysqli_num_fields($result);

   printf("Result set has %d fields.\n", $field_cnt);

   /* close result set */
   mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Result set has 5 fields.
```

See Also

- mysqli_fetch_field()

mysqli_result::field_seek

mysqli_field_seek

mysqli_result::field_seek -- mysqli_field_seek -- Set result pointer to a specified field offset

Description

Object oriented style (method):

bool **mysqli_result::field_seek** (int `$fieldnr`)

Procedural style:

bool **mysqli_field_seek** (mysqli_result `$result`, int `$fieldnr`)

Sets the field cursor to the given offset. The next call to mysqli_fetch_field() will retrieve the field definition of the column associated with that offset.

Note
To seek to the beginning of a row, pass an offset value of zero.

Parameters

`result`
: Procedural style only: A result set identifier returned by mysqli_query(), mysqli_store_result() or mysqli_use_result().

`fieldnr`
: The field number. This value must be in the range from *0* to *number of fields - 1*.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #119 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");
```

```
/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";

if ($result = $mysqli->query($query)) {

   /* Get field information for 2nd column */
   $result->field_seek(1);
   $finfo = $result->fetch_field();

   printf("Name:     %s\n", $finfo->name);
   printf("Table:    %s\n", $finfo->table);
   printf("max. Len: %d\n", $finfo->max_length);
   printf("Flags:    %d\n", $finfo->flags);
   printf("Type:     %d\n\n", $finfo->type);

   $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Example #120 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";

if ($result = mysqli_query($link, $query)) {

   /* Get field information for 2nd column */
   mysqli_field_seek($result, 1);
   $finfo = mysqli_fetch_field($result);

   printf("Name:     %s\n", $finfo->name);
   printf("Table:    %s\n", $finfo->table);
   printf("max. Len: %d\n", $finfo->max_length);
   printf("Flags:    %d\n", $finfo->flags);
   printf("Type:     %d\n\n", $finfo->type);

   mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Name:     SurfaceArea
Table:    Country
max. Len: 10
Flags:    32769
Type:     4
```

See Also

- mysqli_fetch_field()

mysqli_result::free

mysqli_free_result

mysqli_result::free -- mysqli_free_result -- Frees the memory associated with a result

Description

Object oriented style (all methods are equivalent):

void **mysqli_result::free** (void)

void **mysqli_result::close** (void)

void **mysqli_result::free_result** (void)

Procedural style:

void **mysqli_free_result** (mysqli_result `$result`)

Frees the memory associated with the result.

Note
You should always free your result with mysqli_free_result(), when your result object is not needed anymore.

Parameters

`result`
Procedural style only: A result set identifier returned by mysqli_query(), mysqli_store_result() or mysqli_use_result().

Return Values

No value is returned.

See Also

- mysqli_query()
- mysqli_stmt_store_result()

- mysqli_store_result()
- mysqli_use_result()

mysqli_result->lengths

mysqli_fetch_lengths

mysqli_result->lengths -- mysqli_fetch_lengths -- Returns the lengths of the columns of the current row in the result set

Description

Object oriented style (property):

mysqli_result

array *lengths*;

Procedural style:

array **mysqli_fetch_lengths** (mysqli_result `$result`)

The mysqli_fetch_lengths() function returns an array containing the lengths of every column of the current row within the result set.

Parameters

`result`
: Procedural style only: A result set identifier returned by mysqli_query(), mysqli_store_result() or mysqli_use_result().

Return Values

An array of integers representing the size of each column (not including any terminating null characters). **FALSE** if an error occurred.

mysqli_fetch_lengths() is valid only for the current row of the result set. It returns **FALSE** if you call it before calling mysqli_fetch_row/array/object or after retrieving all rows in the result.

Examples

Example #121 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT * from Country ORDER BY Code LIMIT 1";

if ($result = $mysqli->query($query)) {

   $row = $result->fetch_row();

   /* display column lengths */
   foreach ($result->lengths as $i => $val) {
       printf("Field %2d has Length %2d\n", $i+1, $val);
   }
   $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Example #122 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

$query = "SELECT * from Country ORDER BY Code LIMIT 1";

if ($result = mysqli_query($link, $query)) {

   $row = mysqli_fetch_row($result);

   /* display column lengths */
   foreach (mysqli_fetch_lengths($result) as $i => $val) {
       printf("Field %2d has Length %2d\n", $i+1, $val);
   }
   mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Field  1 has Length  3
Field  2 has Length  5
Field  3 has Length 13
Field  4 has Length  9
Field  5 has Length  6
Field  6 has Length  1
Field  7 has Length  6
Field  8 has Length  4
Field  9 has Length  6
Field 10 has Length  6
Field 11 has Length  5
Field 12 has Length 44
Field 13 has Length  7
Field 14 has Length  3
Field 15 has Length  2
```

mysqli_result->num_rows

mysqli_num_rows

mysqli_result->num_rows -- mysqli_num_rows -- Gets the number of rows in a result

Description

Object oriented style (property):

mysqli_result

int *num_rows*;

Procedural style:

int **mysqli_num_rows** (mysqli_result `$result`)

Returns the number of rows in the result set.

The use of mysqli_num_rows() depends on whether you use buffered or unbuffered result sets. In case you use unbuffered resultsets mysqli_num_rows() will not correct the correct number of rows until all the rows in the result have been retrieved.

Parameters

`result`
Procedural style only: A result set identifier returned by mysqli_query(), mysqli_store_result() or mysqli_use_result().

Return Values

Returns number of rows in the result set.

Note
If the number of rows is greater than maximal int value, the number will be returned as a string.

Examples

Example #123 - Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

if ($result = $mysqli->query("SELECT Code, Name FROM Country ORDER BY
Name")) {

   /* determine number of rows result set */
   $row_cnt = $result->num_rows;

   printf("Result set has %d rows.\n", $row_cnt);

   /* close result set */
   $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Example #124 - Procedural style

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

if ($result = mysqli_query($link, "SELECT Code, Name FROM Country ORDER BY
Name")) {

   /* determine number of rows result set */
   $row_cnt = mysqli_num_rows($result);

   printf("Result set has %d rows.\n", $row_cnt);

   /* close result set */
   mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

The above example will output:

```
Result set has 239 rows.
```

See Also

- mysqli_affected_rows()
- mysqli_store_result()
- mysqli_use_result()
- mysqli_query()

The MySQLi_Driver class

Introduction

MySQLi Driver.

Class synopsis

MySQLi_Driver

```
MySQLi_Driver {

    /*  Properties */

    public readonly string client_info;

    public readonly string client_version;

    public readonly string driver_version;

    public readonly string embedded;

    public bool reconnect;

    public int report-mode;

    /*  Methods */

    void mysqli_driver::embedded_server_end ( void )

    bool mysqli_driver::embedded_server_start ( bool $start, array $arguments, array
    $groups )
}
```

Properties

client_info
: The Client API header version

client_version
: The Client version

driver_version
: The MySQLi Driver version

embedded
: Wether MySQLi Embedded support is enabled

reconnect
: Allow or prevent reconnect (see the mysqli.reconnect INI directive)

report_mode
: Set to **MYSQLI_REPORT_STRICT** to throw Exceptions for errors

mysqli_driver::embedded_server_end

mysqli_embedded_server_end

mysqli_driver::embedded_server_end -- mysqli_embedded_server_end -- Stop embedded server

Description

void **mysqli_driver::embedded_server_end** (void)

void **mysqli_embedded_server_end** (void)

Warning
This function is currently not documented; only its argument list is available.

mysqli_driver::embedded_server_start

mysqli_embedded_server_start

mysqli_driver::embedded_server_start -- mysqli_embedded_server_start -- Initialize and start embedded server

Description

bool **mysqli_driver::embedded_server_start** (bool `$start`, array `$arguments`, array `$groups`)

bool **mysqli_embedded_server_start** (bool `$start`, array `$arguments`, array `$groups`)

Warning
This function is currently not documented; only its argument list is available.

Aliases and deprecated Mysqli Functions

mysqli_bind_param

mysqli_bind_param -- Alias for mysqli_stmt_bind_param()

Description

This function is an alias of mysqli_stmt_bind_param().

Notes

Note
mysqli_bind_param() is deprecated and will be removed.

See Also

- mysqli_stmt_bind_param()

mysqli_bind_result

mysqli_bind_result -- Alias for mysqli_stmt_bind_result()

Description

This function is an alias of mysqli_stmt_bind_result().

Notes

Note
mysqli_bind_result() is deprecated and will be removed.

See Also

- mysqli_stmt_bind_result()

mysqli_client_encoding

mysqli_client_encoding -- Alias of mysqli_character_set_name()

Description

This function is an alias of mysqli_character_set_name().

See Also

- mysqli_real_escape_string()

mysqli_disable_reads_from_master

mysqli->disable_reads_from_master()

mysqli_disable_reads_from_master -- mysqli->disable_reads_from_master() -- Disable reads from master

Description

Procedural style:

bool **mysqli_disable_reads_from_master** (mysqli `$link`)

Object oriented style (method):

mysqli

void **disable_reads_from_master** (void)

Warning
This function is currently not documented; only its argument list is available.

Warning
This function has been *DEPRECATED* and *REMOVED* as of PHP 5.3.0.

mysqli_disable_rpl_parse

mysqli_disable_rpl_parse -- Disable RPL parse

Description

bool **mysqli_disable_rpl_parse** (mysqli `$link`)

Warning
This function is currently not documented; only its argument list is available.

Warning
This function has been *DEPRECATED* and *REMOVED* as of PHP 5.3.0.

mysqli_enable_reads_from_master

mysqli_enable_reads_from_master -- Enable reads from master

Description

bool **mysqli_enable_reads_from_master** (mysqli `$link`)

Warning
This function is currently not documented; only its argument list is available.

Warning
This function has been *DEPRECATED* and *REMOVED* as of PHP 5.3.0.

mysqli_enable_rpl_parse

mysqli_enable_rpl_parse -- Enable RPL parse

Description

bool **mysqli_enable_rpl_parse** (mysqli `$link`)

Warning
This function is currently not documented; only its argument list is available.

Warning
This function has been *DEPRECATED* and *REMOVED* as of PHP 5.3.0.

mysqli_escape_string

mysqli_escape_string -- Alias of mysqli_real_escape_string()

Description

This function is an alias of mysqli_real_escape_string().

See Also

- mysqli_real_escape_string()

mysqli_execute

mysqli_execute -- Alias for mysqli_stmt_execute()

Description

This function is an alias of mysqli_stmt_execute().

Notes

Note
mysqli_execute() is deprecated and will be removed.

See Also

- mysqli_stmt_execute()

mysqli_fetch

mysqli_fetch -- Alias for mysqli_stmt_fetch()

Description

This function is an alias of mysqli_stmt_fetch().

Notes

Note
mysqli_fetch() is deprecated and will be removed.

See Also

- mysqli_stmt_fetch()

mysqli_get_metadata

mysqli_get_metadata -- Alias for mysqli_stmt_result_metadata()

Description

This function is an alias of mysqli_stmt_result_metadata().

Notes

Note
mysqli_get_metadata() is deprecated and will be removed.

See Also

- mysqli_stmt_result_metadata()

mysqli_master_query

mysqli_master_query -- Enforce execution of a query on the master in a master/slave setup

Description

bool **mysqli_master_query** (mysqli `$link`, string `$query`)

Warning
This function is currently not documented; only its argument list is available.

Warning
This function has been *DEPRECATED* and *REMOVED* as of PHP 5.3.0.

mysqli_param_count

mysqli_param_count -- Alias for mysqli_stmt_param_count()

Description

This function is an alias of mysqli_stmt_param_count().

Notes

Note
mysqli_param_count() is deprecated and will be removed.

See Also

- mysqli_stmt_param_count()

mysqli_report

mysqli_report -- Enables or disables internal report functions

Description

bool **mysqli_report** (int `$flags`)

mysqli_report() is a powerful function to improve your queries and code during development and testing phase. Depending on the flags it reports errors from mysqli function calls or queries which don't use an index (or use a bad index).

Parameters

`flags`

Supported flags

Name	Description
MYSQLI_REPORT_OFF	Turns reporting off
MYSQLI_REPORT_ERROR	Report errors from mysqli function calls
MYSQLI_REPORT_STRICT	Report warnings from mysqli function calls
MYSQLI_REPORT_INDEX	Report if no index or bad index was used in a query
MYSQLI_REPORT_ALL	Set all options (report all)

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #125 - Object oriented style

```
<?php
/* activate reporting */
mysqli_report(MYSQLI_REPORT_ALL);
```

```
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
}

/* this query should report an error */
$result = $mysqli->query("SELECT Name FROM Nonexistingtable WHERE population >
50000");

/* this query should report a warning */
$result = $mysqli->query("SELECT Name FROM City WHERE population > 50000");
$result->close();

$mysqli->close();
?>
```

See Also

- mysqli_debug()
- mysqli_dump_debug_info()

mysqli_rpl_parse_enabled

mysqli_rpl_parse_enabled -- Check if RPL parse is enabled

Description

int **mysqli_rpl_parse_enabled** (mysqli `$link`)

Warning
This function is currently not documented; only its argument list is available.

Warning
This function has been *DEPRECATED* and *REMOVED* as of PHP 5.3.0.

mysqli_rpl_probe

mysqli_rpl_probe -- RPL probe

Description

bool **mysqli_rpl_probe** (mysqli `$link`)

Warning
This function is currently not documented; only its argument list is available.

Warning
This function has been *DEPRECATED* and *REMOVED* as of PHP 5.3.0.

mysqli_rpl_query_type

mysqli->rpl_query_type()

mysqli_rpl_query_type -- mysqli->rpl_query_type() -- Returns RPL query type

Description

Procedural style:

int **mysqli_rpl_query_type** (mysqli `$link`, string `$query`)

Object oriented style (method)

mysqli

int **rpl_query_type** (string `$query`)

Returns **MYSQLI_RPL_MASTER**, **MYSQLI_RPL_SLAVE** or **MYSQLI_RPL_ADMIN** depending on a query type. *INSERT*, *UPDATE* and similar are *master* queries, *SELECT* is *slave*, and *FLUSH*, *REPAIR* and similar are *admin*.

Warning
This function is currently not documented; only its argument list is available.

Warning
This function has been *DEPRECATED* and *REMOVED* as of PHP 5.3.0.

mysqli_send_long_data

mysqli_send_long_data -- Alias for mysqli_stmt_send_long_data()

Description

This function is an alias of mysqli_stmt_send_long_data().

Notes

Note
mysqli_send_long_data() is deprecated and will be removed.

See Also

- mysqli_stmt_send_long_data()

mysqli_send_query

mysqli->send_query()

mysqli_send_query -- mysqli->send_query() -- Send the query and return

Description

Procedural style:

bool **mysqli_send_query** (mysqli `$link`, string `$query`)

Object oriented style (method)

mysqli

bool **send_query** (string `$query`)

Warning
This function is currently not documented; only its argument list is available.

Warning
This function has been *DEPRECATED* and *REMOVED* as of PHP 5.3.0.

mysqli_set_opt

mysqli_set_opt -- Alias of mysqli_options()

Description

This function is an alias of mysqli_options().

mysqli_slave_query

mysqli_slave_query -- Force execution of a query on a slave in a master/slave setup

Description

bool **mysqli_slave_query** (mysqli `$link`, string `$query`)

Warning
This function is currently not documented; only its argument list is available.

Warning
This function has been *DEPRECATED* and *REMOVED* as of PHP 5.3.0.