DALMP Documentation

Release 3

Nicolas Embriz

June 06, 2016

Contents

Database Abstraction Layer for MySQL using PHP

0% fat and extremely easy to use. Only connect to database when needed.

Clone the repository:

```
$ git clone git://github.com/nbari/DALMP.git dalmp
```

See also:

Install

CHAPTER 1

Details

- Dependecy Injector (DI) support, load once, trigger when required.
- APC, Disk, Memcache, Redis.io cache support.
- Group caching cache by groups and flush by groups or individual keys.
- Prepared statements ready, support dynamic building queries, auto detect types (i,d,s,b).
- Secure connections with SSL.
- SQLite3 Encryption.
- Save sessions in database (mysql/sqlite) or a cache like redis/memcache/apc.
- Easy to use/install/adapt.
- Nested Transactions (SAVEPOINT / ROLLBACK TO SAVEPOINT).
- Support connections via unix_sockets.
- SQL queues.
- Export to CSV.
- Trace/measure everything enabling the debugger.
- Works out of the box with Cloud databases like Amazon RDS or Google cloud.
- Lazy database connection. Connect only when needed.
- PSR-0 compliance.

CHAPTER 2

Requirements

- PHP >= 5.4
- A MySQL server to connect via host or unix sockets.

To use the cache features you need either the redis, memcache or APC extensions compiled, otherwise disk cache will be used.

- Redis extension - http://github.com/nicolasff/phpredis
- Memcache PECL extencsion - http://pecl.php.net/package/memcache
- APC PECL extension - http://pecl.php.net/package/APC

If you want to store session encrypted then you need SQLite3 Encryption (http://sqlcipher.net).

DALMP does not use PDO, so do not worry if your PHP does not have the pdo extension.

On FreeBSD you can install **DALMP** from ports: /usr/ports/databases/dalmp

CHAPTER 3

Table of Contents

3.1 DALMP

Database Abstraction Layer for MySQL using PHP

3.1.1 What & Why DALMP

PHP and MySQL is one of the most popular combinations used in web applications, sometimes this "combo" join forces with tools like: redis, memcache, APC, etc, always trying to achieve the best possible performance.

Setting all this together becomes tricky, especially when your goals are "speed & security" and you have a short deadline.

DALMP makes all this integration without hassle, offering several methods that can help the developer to focus more in optimizing his 'SQL statements' rather than worry about how to properly configure cache instances or about duplicating current connections to the database.

See also:

Prepared statements

One of the main goals of **DALMP** is to avoid complexity at all cost without losing flexibility and performance. The main class uses the PHP mysqli extension, therefore there is not need to have the PDO extension (older version of PHP didn't include PDO by default).

To take advantage of the cache class and methods it is suggested to install the following extensions:

- redis: http://github.com/nicolasff/phpredis
- memcache: http://pecl.php.net/package/memcache

See also:

Dalmp\Cache

If you have a site on the cloud or in a load balanced enviroment, you could take advantege of how **DALMP** handle sessions by storing them in a database or in a cache engine.

See also:

Dalmp\Sessions

On FreeBSD you can install **DALMP** from ports: /usr/ports/databases/dalmp.

3.2 Download

Zip File: https://github.com/nbari/dalmp/zipball/master

Tar Ball: https://github.com/nbari/dalmp/tarball/master

View on GitHub: https://github.com/nbari/dalmp

Clone the repository:

```
$ git clone git://github.com/nbari/DALMP.git dalmp
```

3.3 Install

Clone the repository:

```
$ git clone git://github.com/nbari/DALMP.git dalmp
```

3.3.1 Composer

A basic composer.json file:

```
{
  "require": {
    "DALMP/dalmp": "*"
  },
  "minimum-stability": "dev"
}
```

```
$ composer.phar install
```

See also:

composer

3.4 Quick Start

Connecting and doing a query:

```
<?php

require_once 'dalmp.php';

$db = new DALMP\database('utf8://root@localhost');

$rs = $db->FetchMode('ASSOC')->GetAssoc('SHOW VARIABLES LIKE "char%"');
```

Note: **DALMP** is the name of the namespace

Will output something like:

```
Array
(
    [character_set_client] => utf8
    [character_set_connection] => utf8
    [character_set_database] => latin1
    [character_set_filesystem] => binary
    [character_set_results] => utf8
    [character_set_server] => latin1
    [character_set_system] => utf8
    [character_sets_dir] => /usr/local/mysql-5.6.10-osx10.7-x86/share/charsets/
)
```

`DALMP\Database` takes the parameters from a DNS (database source name) so before you can start using it you need to define this values.

3.4.1 DSN format

```
charset://username:password@host:port/database
```

When using Unix domain sockets:

```
charset://username:password@unix_socket=\path\of\the.socket/database
```

- Notice that the path of the socket is using backslashes.

```
\path\of\the.socket
```

Will be translated to:

```
/path/of/the.socket
```

See also:

Unix sockets vs Internet sockets

3.4.2 DSN Cache format

```
charset://username:password@host:port/database?(type:host:port:compression)
```

type Memcache, Redis, Disk.

host The host of the Memcache, Redis server.

port The port of the Memcache, Redis server.

compression To use or not compression, only available for memcache.

See also:

DALMP\Database Cache method.

3.4.3 Common methods

The next table contains, 5 common methods for retrieving data:

Name	Normal	Prepared Statements	Cache Normal	Cache Prepared Statements
all	GetAll	PGetAll	CacheGetAll	CachePGetAll
assoc	GetAssoc	PGetAssoc	CacheGetAssoc	CachePGetAssoc
col	GetCol	PGetCol	CacheGetCol	CachePGetCol
one	GetOne	PGetOne	CacheGetOne	CachePGetOne
row	GetRow	PGetRow	CacheGetRow	CachePGetRow

For Inserting or Updating, you can use the Execure or PExecute methods.

See also:

Prepared statements.

3.4.4 DALMP Classes

For better code maintainability, **DALMP** is formed by different classes, the main class and the one that does the abstraction layer is DALMP\Database.

mysql	DALMP\Database
cache	DALMP\Cache
queue	DALMP\Queue
sessions	DALMP\Sessions
DI	DALMP\DI

3.4.5 Examples

DSN values

charset utf8
user $user
password $password
host 127.0.0.1
database test

```php
<?php

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

require_once 'dalmp.php';

$DSN = "utf8://$user:$password@127.0.0.1/test";

$db = new DALMP\Database($DSN);

try {
    $rs = $db->getOne('SELECT now()');
} catch (\Exception $e) {
    print_r($e->getMessage());
}

/**
 * 1 log to single file
 * 2 log to multiple files (creates a log per request)
```

```
 * 'off' to stop debuging
 */
$db->debug(1);

echo $db, PHP_EOL; // print connection details
```

If you wan to use the system default charset the DSN would be:

```
$DSN = "mysql://$user:$password@127.0.0.1/test";
```

- notice the **mysql://** instead of the **utf8://**

3.4.6 SSL

If you want to use SSL, an array containing the SSL parameters must be passed as the second argument to the database method example:

```
$ssl = array('key' => null, 'cert' => null, 'ca' => 'mysql-ssl.ca-cert.pem', 'capath' => null, 'ciphe

$DSN = 'latin1://root:secret@127.0.0.1/test';

$db = new DALMP\Database($DSN, $ssl);
```

The **$ssl** array argument, must follow this format:

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.

Note: When using SSL, PHP OpenSSL support must be enable for this to work.

To check that your connection has SSL you can test with this:

```
<?php

require_once 'dalmp.php';

$ssl = array('key' => null, 'cert' => null, 'ca' => 'mysql-ssl.ca-cert.pem', 'capath' => null, 'ciphe

$DSN = 'utf8://root:secret@127.0.0.1/test';

$db = new DALMP\Database($DSN, $ssl);

try {
  $db->getOne('SELECT NOW()');
  print_r($db->FetchMode('ASSOC')->GetRow("show variables like 'have_ssl'"));
} catch (\Exception $e) {
  print_r($e->getMessage());
}

try {
  print_r($db->GetRow("show status like 'ssl_cipher'"));
```

```
} catch (\Exception $e) {
  print_r($e->getMessage());
}
```

If you have SSL you will get something like:

```
Array
(
  [Variable_name] => have_ssl
  [Value] => YES
)

Array
(
  [Variable_name] => Ssl_cipher
  [Value] => DHE-RSA-AES256-SHA
)
```

Otherwise:

```
Array
(
  [Variable_name] => have_ssl
  [Value] => DISABLED
)

Array
(
  [Variable_name] => Ssl_cipher
  [Value] =>
)
```

3.4.7 Example using a socket

```
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@unix_socket=\tmp\mysql.sock/test';

$db = new DALMP\Database($DSN);

$db->debug(1);

try {
  echo PHP_EOL, 'example using unix_socket: ', $db->getOne('SELECT NOW()'), PHP_EOL;
} catch (\Exception $e) {
  print_r($e->getMessage());
}

echo $db;
# will print: DALMP :: connected to: db, Character set: utf8, Localhost via UNIX socket,...
```

3.4.8 Example using cache (memcache)

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@localhost/test';

$db = new DALMP\Database($DSN);

$cache = new DALMP\Cache(new DALMP\Cache\Memcache());

$db->useCache($cache);

$rs = $db->CacheGetOne('SELECT now()');

echo $rs, PHP_EOL;
```

3.4.9 Example using DSN cache (redis)

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@localhost/dalmp?redis:127.0.0.1:6379';

$db = new DALMP\Database($DSN);

$db->FetchMode('ASSOC');

$rs = $db->CacheGetAll('SELECT * FROM City');

echo $rs, PHP_EOL;
```

See also:

DALMP Examples

3.5 DALMP\Database

The `DALMP\Database` class contains a set of methods that allow to query a database in a more easy and secure way.

The next table contains, 5 common methods for retrieving data:

Name	Normal	Prepared Statements	Cache Normal	Cache Prepared Statements
all	GetAll	PGetAll	CacheGetAll	CachePGetAll
assoc	GetAssoc	PGetAssoc	CacheGetAssoc	CachePGetAssoc
col	GetCol	PGetCol	CacheGetCol	CachePGetCol
one	GetOne	PGetOne	CacheGetOne	CachePGetOne
row	GetRow	PGetRow	CacheGetRow	CachePGetRow

See also:

DALMP\Cache

Any query or either for Inserting or Updating:

Name	Normal	Prepared statements
Execute	Execute	PExecute

See also:

Prepared statements

The available methods are:

3.5.1 AutoExecute

Automatically prepares and runs an INSERT or UPDATE query based on variables you supply.

This is very usefull when you simple want to save post data from a huge web form, AutoExecute will genereate a mysql prepared statement from the array used and INSERT or UPDATE

Parameters

```
AutoExecute($table, $fields, $mode = 'INSERT', $where = null)
```

$table The name of the table you want to INSERT or UPDATE

$fields An assoc array (key => value), keys are fields names, values are values of these fields.

$mode INSERT or UPDATE

$where A string to be used in the WHERE clause. This is only used when $mode = UPDATE.

Examples

Insert all $_POST data example:

```
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

// the key values of $_POST must be equal to the column names of the mysql table
$db->AutoExecute('mytable', $_POST);
```

Update example:

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

$date = array('username' => 'nbari',
              'status' => 1);

$db->AutoExecute('mytable', $data, 'UPDATE', 'status=0 AND uid=14');
```

3.5.2 Cache

The method `Cache` returns or instantiate a DALMP\Cache instance.

To use the cache features you must specify the type of cache either via DSN or by passing a DALMP\Cache instance as an argument to the method useCache.

DSN Cache format

```
charset://username:password@host:port/database?(type:host:port:compression)
```

type Memcache, Redis, Disk.

host The host of the Memcache, Redis server.

port The port of the Memcache, Redis server.

compression To use or not compression, only available for memcache.

Note: If no Cache is specified, defaults to disk cache type.

The Cache methods

The idea of using a 'cache' is to dispatch faster the results of a previous query with out need to connect again to the database and fetch the results.

There are five methods you can use within the Cache method which are:

method	Normal	Prepared statements
all	CacheGetAll	CachePGetAll
assoc	CacheGetASSOC	CachePGetASSOC
col	CacheGetCol	CachePGetCol
one	CacheGetOne	CachePGetOne
row	CacheGetRow	CachePGetRow

Note: Notice that when using "Cache" the methods are prefixed with **Cache**.

Constants

```
define('DALMP_CACHE_DIR', '/tmp/dalmp/cache/');
```

Defines where to store the cache when using dir cache type.

How to use

Whenever you want to use the cache, just just need to prepend the word **Cache** to the method you are using.

Parameters

You can have finer control over your cached queries, for this you have the following options:

```
Cache[P]method(TTL, <query>, key or group:X)
```

Cache[P]method A normal or prepared statements method: 'all, assoc, col, one, row'

TTL The time to live (timeout) in seconds for your query, default 3600 seconds / 1 hour if not set.

query A normal or prepared statements query.

key or group A unique custom key for storing the query result or the name of a caching group:**X** Where X is the desired name of the group.

TTL example

Cache the results for 300 seconds, 5 minutes:

```
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@localhost/dalmp?redis:127.0.0.1:6379';

$db = new DALMP\Database($DSN);

$db->FetchMode('ASSOC');

$rs = $db->CacheGetAll(300, 'SELECT * FROM City');

echo $rs, PHP_EOL;
```

Custom key example

If you specify a custom key, the query result will be stored on the cache.

On the cache engine, the (key, value) is translated to:

key Your custom key.

value The output of your query.

This is useful when you only want to flush certain parts of the cache, example:

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@localhost/dalmp?redis:127.0.0.1:6379';

$db = new DALMP\Database($DSN);

$db->FetchMode('ASSOC');

$rs = $db->CacheGetAll(300, 'SELECT * FROM City', 'my_custom_key');

// To flush the query
$db->CacheFlush('SELECT * FROM City', 'my_custom_key');
```

Group caching, group:X

Helps to group your queries in groups, so that later you can only flush group without affecting the rest of your cache.

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@localhost/dalmp?redis:127.0.0.1:6379';

$db = new DALMP\Database($DSN);

$db->FetchMode('ASSOC');

$rs = $db->CacheGetAll(300, 'SELECT * FROM City', 'group:B');

// To flush the group
$db->CacheFlush('group:B');
```

Note: When creating a cache group for your queries all of them must start with **group:**, so if you want a group called 'my_group' it should be: **group:my_group**.

See also:

CacheFlush & Cache Examples.

3.5.3 CacheFlush

Flush the cache, if no parameter specified it will flush all the cache.

Parameters

```
CacheFlush(sql or group, $key=null)
```

sql or group The query to flush or the group name.

$key The custom key assigned to the query.

To flush / empty all the cache just call the CacheFlush with no parameteres, example:

```
$db->CacheFlush();
```

Examples

Flush a query:

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@localhost/dalmp?redis:127.0.0.1:6379';

$db = new DALMP\Database($DSN);

$db->FetchMode('ASSOC');

$rs = $db->CacheGetAll(300, 'SELECT * FROM City');

// To flush the query
$db->CacheFlush('SELECT * FROM City');
```

Flush a query with a custom key:

```php
<?php
...
$rs = $db->CacheGetAll(300, 'SELECT * FROM City', 'my_custom_key');

$db->CacheFlush('SELECT * FROM City', 'my_custom_key');
```

Flushing a chached group:

```php
<?php
...
$rs = $db->CachePGetAll('SELECT * FROM Country WHERE Population <= ?', 100000, 'group:B');
$rs = $db->CachePGetAll(86400, 'SELECT * FROM Country WHERE Continent = ?', 'Europe', 'group:B');

// To flush all the group
$db->CacheFlush('group:B');
```

3.5.4 Close

Closes a previously opened database connection, you normally not need to call this method, since DALMP when finishes automatically close all opening connections.

3.5.5 CompleteTrans

Complete the transaction, this must be used in conjunction with method StartTrans.

If success returns **true**, otherwise **false**.

Example

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

$db->Execute('CREATE TABLE IF NOT EXISTS t_test (id INT NOT NULL PRIMARY KEY) ENGINE=InnoDB');
$db->Execute('TRUNCATE TABLE t_test');
$db->FetchMode('ASSOC');

$db->StartTrans();
$db->Execute('INSERT INTO t_test VALUES(1)');
    $db->StartTrans();
    $db->Execute('INSERT INTO t_test VALUES(2)');
    print_r($db->GetAll('SELECT * FROM t_test'));
        $db->StartTrans();
        $db->Execute('INSERT INTO t_test VALUES(3)');
        print_r($db->GetAll('SELECT * FROM t_test'));
            $db->StartTrans();
            $db->Execute('INSERT INTO t_test VALUES(7)');
            print_r($db->GetALL('SELECT * FROM t_test'));
        $db->RollBackTrans();
        print_r($db->GetALL('SELECT * FROM t_test'));
        $db->CompleteTrans();
    $db->CompleteTrans();
$db->CompleteTrans();

if ($db->CompleteTrans()) {
 // your code
}
```

See also:

StartTrans

3.5.6 __construct

`DALMP\Database` takes the parameters from a DNS (database source name) so before you can start using it you need to define this values.

DSN format

```
charset://username:password@host:port/database
```

When using Unix domain sockets:

```
charset://username:password@unix_socket=\path\of\the.socket/database
```

- Notice that the path of the socket is using backslashes.

```
\path\of\the.socket
```

Will be translated to:

```
/path/of/the.socket
```

See also:

Unix sockets vs Internet sockets

Examples

DSN values

charset utf8
user $user
password $password
host 127.0.0.1
database test

```php
<?php

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

require_once 'dalmp.php';

$DSN = "utf8://$user:$password@127.0.0.1/test";

$db = new DALMP\Database($DSN);

try {
    $rs = $db->getOne('SELECT now()');
} catch (\Exception $e) {
    print_r($e->getMessage());
}

/**
 * 1 log to single file
 * 2 log to multiple files (creates a log per request)
 * 'off' to stop debuging
 */
$db->debug(1);

echo $db, PHP_EOL; // print connection details
```

If you wan to use the system default charset the DSN would be:

```
$DSN = "mysql://$user:$password@127.0.0.1/test";
```

- notice the **mysql://** instead of the **utf8://**

See also:

Quickstart.

3.5.7 csv

This method exports your results in CSV (Comma Separated Values).

Parameters

```
csv($sql)
```

$sql Your normal sql or either a prepared statement query.

returns CVS.

Example

Serve a CSV file:

```
<?php

header("Content-type: application/csv");
header("Content-Disposition: attachment; filename=$filename.csv");
header("Pragma: no-cache");
header("Expires: 0");

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

$db->csv('SELECT row1, row2, row3, row4 FROM table WHERE uid=? AND cat=?', 3, 'oi');
```

3.5.8 debug

This method will enable debugging, so that you can trace your full queries.

Parameters

```
debug($log2file = false, $debugFile = false)
```

$log2file When set to **1**, log is written to a single file, if **2** it creates multiple log files per request so that you can do a more intense debugging, **off** stop debuging.

$debugFile Path of the file to write logs, defaults to `/tmp/dalmp.log`

Constants

```
define('DALMP_DEBUG_FILE', '/home/tmp/debug.log');
```

Defines where the debug file will be write to.

Example

```
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

/**
 * 1 log to single file
 * 2 log to multiple files (creates a log per request)
 * off stop debug
 */
 $db->debug(1);

 try {
   $rs = $db->getOne('SELECT now()');
 } catch (Exception $e) {
   print_r($e->getMessage());
 }

 echo $db,PHP_EOL; // print connection details
```

3.5.9 ErrorMsg

Returns a string description of the last error.

Example

```
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

try {
  $db->Execute('SET a=1');
} catch (Exception $d) {
```

```
  // Errormessage: Unknown system variable 'a'
  printf("Errormessage: %s\n", $db->ErrorMsg());
}
```

3.5.10 ErrorNum

Returns the error code for the most recent function call.

Example

```
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

try {
  $db->Execute('SET a=1');
} catch (Exception $d) {
  // Errormessage: 1193
  printf("Errormessage: %s\n", $db->ErrorNum());
}
```

3.5.11 Execute

Execute an SQL statement, returns true on success, false if there was an error in executing the sql.

Parameters

```
Execute($sql)
```

$sql The MySQL query to perform on the database.

In most cases you only use this method when Inserting or Updating data, for retrieving data the 5 common methods are:

Name	Normal	Prepared Statements	Cache Normal	Cache Prepared Statements
all	GetAll	PGetAll	CacheGetAll	CachePGetAll
assoc	GetAssoc	PGetAssoc	CacheGetAssoc	CachePGetAssoc
col	GetCol	PGetCol	CacheGetCol	CachePGetCol
one	GetOne	PGetOne	CacheGetOne	CachePGetOne
row	GetRow	PGetRow	CacheGetRow	CachePGetRow

See also:

Prepared statements & PExecute.

Examples

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

$rs = $db->Execute("INSERT INTO table (name,email,age) VALUES('name', 'email', 70)");
```

You can also catch exception and continue execution:

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

try {
    $db->Execute('CREATE TABLE myCity LIKE City');
} catch (Exception $e) {
    echo "Table already exists.",$db->isCli(1);
}

$db->Execute("INSERT INTO myCity VALUES (NULL, 'Toluca', 'MEX', 'México', 467713)");
```

3.5.12 FetchMode

This chainable method indicates what type of array should be returned.

Parameters

```
FetchMode($mode = null)
```

$mode can be NUM (MYSQLI_NUM), ASSOC (MYSQLI_ASSOC) or if not set, it will use both (MYSQLI_BOTH).

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 e need to access the result with numeric indices by using **FetchMode('NUM')** or add alias names.

Examples

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/dalmp';

$db = new DALMP\Database($DSN);

$db->FetchMode('NUM');

$rs = $db->PGetAll('SELECT * FROM Country WHERE Region = ?', 'Caribbean');
```

Chainable example

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/dalmp';

$db = new DALMP\Database($DSN);

$db->FetchMode('NUM')->PGetAll('SELECT * FROM Country WHERE Region = ?', 'Caribbean');
```

When using NUM the keys of the result array are numeric. Example of the output:

```
// output of print($rs);
Array
(
    [0] => Array
        (
            [0] => ABW
            [1] => Aruba
            [2] => North America
            [3] => Caribbean
            [4] => 193
            [5] =>
            [6] => 103000
            [7] => 78.400001525879
            [8] => 828
            [9] => 793
            [10] => Aruba
            [11] => Nonmetropolitan Territory of The Netherlands
            [12] => Beatrix
            [13] => 129
            [14] => AW
        )

    [1] => Array
        (
            [0] => AIA
```

```
            [1] => Anguilla
            [2] => North America
            [3] => Caribbean
            [4] => 96
            [5] =>
            [6] => 8000
            [7] => 76.099998474121
            [8] => 63.200000762939
            [9] =>
            [10] => Anguilla
            [11] => Dependent Territory of the UK
            [12] => Elisabeth II
            [13] => 62
            [14] => AI
        )
    ...
```

ASSOC mode, example

```php
<?php
...
$rs = $db->FetchMode('ASSOC')->PGetAll('SELECT * FROM Country WHERE Region = ?', 'Caribbean');
```

The output would be something like:

```
// output of print($rs);
Array
(
    [0] => Array
        (
            [Code] => ABW
            [Name] => Aruba
            [Continent] => North America
            [Region] => Caribbean
            [SurfaceArea] => 193
            [IndepYear] =>
            [Population] => 103000
            [LifeExpectancy] => 78.400001525879
            [GNP] => 828
            [GNPOld] => 793
            [LocalName] => Aruba
            [GovernmentForm] => Nonmetropolitan Territory of The Netherlands
            [HeadOfState] => Beatrix
            [Capital] => 129
            [Code2] => AW
        )

    [1] => Array
        (
            [Code] => AIA
            [Name] => Anguilla
            [Continent] => North America
            [Region] => Caribbean
            [SurfaceArea] => 96
            [IndepYear] =>
            [Population] => 8000
            [LifeExpectancy] => 76.099998474121
```

```
            [GNP] => 63.200000762939
            [GNPOld] =>
            [LocalName] => Anguilla
            [GovernmentForm] => Dependent Territory of the UK
            [HeadOfState] => Elisabeth II
            [Capital] => 62
            [Code2] => AI
        )
...
```

No mode

When No mode is defined, the default is to use 'both' (MYSQLI_BOTH). example:

```php
<?php
...
$rs = $db->PGetAll('SELECT * FROM Country WHERE Region = ?', 'Caribbean');
```

In this case the output is like:

```
// output of print($rs);
Array
(
    [0] => Array
        (
            [0] => ABW
            [Code] => ABW
            [1] => Aruba
            [Name] => Aruba
            [2] => North America
            [Continent] => North America
            [3] => Caribbean
            [Region] => Caribbean
            [4] => 193
            [SurfaceArea] => 193
            [5] =>
            [IndepYear] =>
            [6] => 103000
            [Population] => 103000
            [7] => 78.400001525879
            [LifeExpectancy] => 78.400001525879
            [8] => 828
            [GNP] => 828
            [9] => 793
            [GNPOld] => 793
            [10] => Aruba
            [LocalName] => Aruba
            [11] => Nonmetropolitan Territory of The Netherlands
            [GovernmentForm] => Nonmetropolitan Territory of The Netherlands
            [12] => Beatrix
            [HeadOfState] => Beatrix
            [13] => 129
            [Capital] => 129
            [14] => AW
            [Code2] => AW
        )

```

```
    [1] => Array
        (
            [0] => AIA
            [Code] => AIA
            [1] => Anguilla
            [Name] => Anguilla
            [2] => North America
            [Continent] => North America
            [3] => Caribbean
            [Region] => Caribbean
            [4] => 96
            [SurfaceArea] => 96
            [5] =>
            [IndepYear] =>
            [6] => 8000
            [Population] => 8000
            [7] => 76.099998474121
            [LifeExpectancy] => 76.099998474121
            [8] => 63.200000762939
            [GNP] => 63.200000762939
            [9] =>
            [GNPOld] =>
            [10] => Anguilla
            [LocalName] => Anguilla
            [11] => Dependent Territory of the UK
            [GovernmentForm] => Dependent Territory of the UK
            [12] => Elisabeth II
            [HeadOfState] => Elisabeth II
            [13] => 62
            [Capital] => 62
            [14] => AI
            [Code2] => AI
        )
...
```

See also:

PHP mysqli_fetch_array & mysqli_stmt_fetch

3.5.13 forceTruncate

Force truncate of a table either if are InnoDB.

Parameters

```
forceTable($table)
```

$table Name of the table to truncate.

Example

```php
<?php

require_once 'dalmp.php';

```

```php
$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

$db->forceTrucate('mytable');
```

3.5.14 getAll / PgetAll

Executes the SQL and returns the all the rows as a 2-dimensional array. If an error occurs, false is returned.

Parameters

```php
getAll($sql)
```

$sql The MySQL query to perfom on the database.

Prepared statements Parameters

```php
PgetAll($sql, $varN)
```

$sql The MySQL query to perfom on the database

$varN The variable(s) that will be placed instead of the **?** placeholder separated by a ',' or it can be the method Prepare.

Example

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

/**
 * GetAll
 */
rs = $db->FetchMode('ASSOC')->GetAll('SELECT name, continent FROM Country WHERE Region ="Caribbean"')

/**
 * Prepared statements
 */
$rs = $db->FetchMode('ASSOC')->PGetAll('SELECT name, continent FROM Country WHERE Region = ?', 'Carib
```

Output of `print_r($rs)`:

```
Array
(
    [0] => Array
        (
            [name] => Aruba
            [continent] => North America
        )

    [1] => Array
        (
            [name] => Anguilla
            [continent] => North America
        )

    [2] => Array
        (
            [name] => Netherlands Antilles
            [continent] => North America
        )

    [3] => Array
        (
            [name] => Antigua and Barbuda
            [continent] => North America
        )

    [4] => Array
        (
            [name] => Bahamas
            [continent] => North America
        )

    [5] => Array
        (
            [name] => Barbados
            [continent] => North America
        )

    [6] => Array
        (
            [name] => Cuba
            [continent] => North America
        )

    [7] => Array
        (
            [name] => Cayman Islands
            [continent] => North America
        )

    [8] => Array
        (
            [name] => Dominica
            [continent] => North America
        )

    [9] => Array
        (
```

```
            [name] => Dominican Republic
            [continent] => North America
        )

    [10] => Array
        (
            [name] => Guadeloupe
            [continent] => North America
        )

    [11] => Array
        (
            [name] => Grenada
            [continent] => North America
        )

    [12] => Array
        (
            [name] => Haiti
            [continent] => North America
        )

    [13] => Array
        (
            [name] => Jamaica
            [continent] => North America
        )

    [14] => Array
        (
            [name] => Saint Kitts and Nevis
            [continent] => North America
        )

    [15] => Array
        (
            [name] => Saint Lucia
            [continent] => North America
        )

    [16] => Array
        (
            [name] => Montserrat
            [continent] => North America
        )

    [17] => Array
        (
            [name] => Martinique
            [continent] => North America
        )

    [18] => Array
        (
            [name] => Puerto Rico
            [continent] => North America
        )

```

```
    [19] => Array
        (
            [name] => Turks and Caicos Islands
            [continent] => North America
        )

    [20] => Array
        (
            [name] => Trinidad and Tobago
            [continent] => North America
        )

    [21] => Array
        (
            [name] => Saint Vincent and the Grenadines
            [continent] => North America
        )

    [22] => Array
        (
            [name] => Virgin Islands, British
            [continent] => North America
        )

    [23] => Array
        (
            [name] => Virgin Islands, U.S.
            [continent] => North America
        )
)
```

3.5.15 getASSOC / PgetASSOC

Executes the SQL and returns an associative array for the given query.

If the number of columns returned is greater to two, a 2-dimensional array is returned, with the first column of the recordset becomes the keys to the rest of the rows. If the columns is equal to two, a 1-dimensional array is created, where the the keys directly map to the values.

If an error occurs, false is returned.

Parameters

```
getASSOC($sql)
```

$sql The MySQL query to perfom on the database.

Prepared statements Parameters

```
PgetASSOC($sql, $varN)
```

$sql The MySQL query to perfom on the database

$varN The variable(s) that will be placed instead of the **?** placeholder separated by a ',' or it can be the method Prepare.

Example

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

$rs = $db->PGetASSOC('SELECT name, continent FROM Country WHERE Region = ?', 'Caribbean');
```

Output of `print_r($rs)`:

```
Array
(
    [Aruba] => North America
    [Anguilla] => North America
    [Netherlands Antilles] => North America
    [Antigua and Barbuda] => North America
    [Bahamas] => North America
    [Barbados] => North America
    [Cuba] => North America
    [Cayman Islands] => North America
    [Dominica] => North America
    [Dominican Republic] => North America
    [Guadeloupe] => North America
    [Grenada] => North America
    [Haiti] => North America
    [Jamaica] => North America
    [Saint Kitts and Nevis] => North America
    [Saint Lucia] => North America
    [Montserrat] => North America
    [Martinique] => North America
    [Puerto Rico] => North America
    [Turks and Caicos Islands] => North America
    [Trinidad and Tobago] => North America
    [Saint Vincent and the Grenadines] => North America
    [Virgin Islands, British] => North America
    [Virgin Islands, U.S.] => North America
)
```

3.5.16 getCol / PgetCol

Executes the SQL and returns all elements of the first column as a 1-dimensional array.

If an error occurs, false is returned.

Parameters

```
getCol($sql)
```

$sql The MySQL query to perfom on the database.

Prepared statements Parameters

```
PgetCol($sql, $varN)
```

$sql The MySQL query to perfom on the database

$varN The variable(s) that will be placed instead of the **?** placeholder separated by a ',' or it can be the method Prepare.

Example

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

$rs = $db->PGetCol('SELECT name FROM Country WHERE Region = ?', 'Caribbean');
```

Output of `print_r($rs)`:

```
Array
(
    [0] => Aruba
    [1] => Anguilla
    [2] => Netherlands Antilles
    [3] => Antigua and Barbuda
    [4] => Bahamas
    [5] => Barbados
    [6] => Cuba
    [7] => Cayman Islands
    [8] => Dominica
    [9] => Dominican Republic
    [10] => Guadeloupe
    [11] => Grenada
    [12] => Haiti
    [13] => Jamaica
    [14] => Saint Kitts and Nevis
    [15] => Saint Lucia
    [16] => Montserrat
    [17] => Martinique
    [18] => Puerto Rico
    [19] => Turks and Caicos Islands
    [20] => Trinidad and Tobago
    [21] => Saint Vincent and the Grenadines
    [22] => Virgin Islands, British
    [23] => Virgin Islands, U.S.
)
```

3.5.17 getOne / PGetOne

Executes the SQL and returns the first field of the first row. If an error occurs, false is returned.

Parameters

```
getOne($sql)
```

$sql The MySQL query to perfom on the database.

Prepared statements Parameters

```
PgetOne($sql, $varN)
```

$sql The MySQL query to perfom on the database

$varN The variable(s) that will be placed instead of the **?** placeholder separated by a ',' or it can be the method Prepare.

Example

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

$rs = $db->PGetOne('SELECT * FROM Country WHERE Region = ? LIMIT 1', 'Caribbean');
```

Output of `echo $rs`:

```
Aruba
```

3.5.18 getRow / PgetRow

Executes the SQL and returns the first field of the first row. If an error occurs, false is returned.

Parameters

```
getRow($sql)
```

$sql The MySQL query to perfom on the database.

Prepared statements Parameters

```
PgetRow($sql, $varN)
```

$sql The MySQL query to perfom on the database

$varN The variable(s) that will be placed instead of the **?** placeholder separated by a ',' or it can be the method Prepare.

Example

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

$rs = $db->PGetRow('SELECT * FROM Country WHERE Region = ? LIMIT 1', 'Caribbean');
```

Output of `print_r($rs)`:

```
Array
(
    [0] => ABW
    [Code] => ABW
    [1] => Aruba
    [Name] => Aruba
    [2] => North America
    [Continent] => North America
    [3] => Caribbean
    [Region] => Caribbean
    [4] => 193
    [SurfaceArea] => 193
    [5] =>
    [IndepYear] =>
    [6] => 103000
    [Population] => 103000
    [7] => 78.400001525879
    [LifeExpectancy] => 78.400001525879
    [8] => 828
    [GNP] => 828
    [9] => 793
    [GNPOld] => 793
    [10] => Aruba
    [LocalName] => Aruba
    [11] => Nonmetropolitan Territory of The Netherlands
    [GovernmentForm] => Nonmetropolitan Territory of The Netherlands
    [12] => Beatrix
    [HeadOfState] => Beatrix
    [13] => 129
    [Capital] => 129
    [14] => AW
    [Code2] => AW
)
```

Same query but using FetchMode('ASSOC')

```php
<?php
...
$rs = $db->FetchMode('ASSOC')->PGetRow('SELECT * FROM Country WHERE Region = ? LIMIT 1', 'Caribbean')
```

Output of `print_r($rs)`:

```
Array
(
    [Code] => ABW
    [Name] => Aruba
    [Continent] => North America
    [Region] => Caribbean
    [SurfaceArea] => 193
    [IndepYear] =>
    [Population] => 103000
    [LifeExpectancy] => 78.400001525879
    [GNP] => 828
    [GNPOld] => 793
    [LocalName] => Aruba
    [GovernmentForm] => Nonmetropolitan Territory of The Netherlands
    [HeadOfState] => Beatrix
    [Capital] => 129
    [Code2] => AW
)
```

3.5.19 getClientVersion

Returns client version number as an integer.

Example

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

echo $db->getClientVersion();
```

3.5.20 getColumnNames

Return the name of the tables.

Parameters

```
getColumnNames($tablename)
```

$tablename Name of the table to get the column names.

Example

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

$rs = $db->getColumnNames();

// output of print_r($rs);
Array
(
    [0] => Code
    [1] => Name
    [2] => Continent
    [3] => Region
    [4] => SurfaceArea
    [5] => IndepYear
    [6] => Population
    [7] => LifeExpectancy
    [8] => GNP
    [9] => GNPOld
    [10] => LocalName
    [11] => GovernmentForm
    [12] => HeadOfState
    [13] => Capital
    [14] => Code2
)
```

3.5.21 getNumOfFields

Returns the number of columns for the most recent query.

Example

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

$rs = $db->FetchMode('ASSOC')->PGetRow('SELECT * FROM Country WHERE Region = ? LIMIT 1', 'Caribbean')

// output of print_r($rs);
```

```
Array
(
    [Code] => ABW
    [Name] => Aruba
    [Continent] => North America
    [Region] => Caribbean
    [SurfaceArea] => 193
    [IndepYear] =>
    [Population] => 103000
    [LifeExpectancy] => 78.400001525879
    [GNP] => 828
    [GNPOld] => 793
    [LocalName] => Aruba
    [GovernmentForm] => Nonmetropolitan Territory of The Netherlands
    [HeadOfState] => Beatrix
    [Capital] => 129
    [Code2] => AW
)

echo $db->getNumOfFields(); // return 15
```

3.5.22 getNumOfRows

Returns the number of rows for the most recent query.

Example

```
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

$rs = $db->FetchMode('ASSOC')->PGetRow('SELECT * FROM Country WHERE Region = ? LIMIT 1', 'Caribbean')

// output of print_r($rs);
Array
(
    [Code] => ABW
    [Name] => Aruba
    [Continent] => North America
    [Region] => Caribbean
    [SurfaceArea] => 193
    [IndepYear] =>
    [Population] => 103000
    [LifeExpectancy] => 78.400001525879
    [GNP] => 828
    [GNPOld] => 793
    [LocalName] => Aruba
    [GovernmentForm] => Nonmetropolitan Territory of The Netherlands
```

```
    [HeadOfState] => Beatrix
    [Capital] => 129
    [Code2] => AW
)

echo $db->getNumOfRows(); // return 1
```

3.5.23 getNumOfRowsAffected

Returns the total number of rows changed, deleted, or inserted by the last executed statement.

3.5.24 getServerVersion

Returns server version number as an integer.

Example

```
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

$rs = $db->FetchMode('ASSOC')->PGetRow('SELECT * FROM Country WHERE Region = ? LIMIT 1', 'Caribbean');

/**
 * After a query made you can get the server version
 */
echo $db->getServerVersion();
```

3.5.25 Insert_Id

Returns the auto generated id used in the last query.

Example

```
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);
```

```php

try {
    $db->Execute('CREATE TABLE myCity LIKE City');
} catch (Exception $e) {
   echo "Table already exists.",$db->isCli(1);
}

$db->Execute("INSERT INTO myCity VALUES (NULL, 'Toluca', 'MEX', 'México', 467713)");
printf ("New Record has id %d.\n", $db->Insert_id());
```

3.5.26 isConnected

Returns boolean (true/false) depending on if it is connected or not to the database.

3.5.27 map

Maps the result to an object.

Parameters

```
map($sql, $class_name=null, $params=array())
```

$sql The MySQL query to perfom on the database.

$class_name The name of the class to instantiate, set the properties of and return. If not specified, a stdClass object is returned.

$params An optional array of parameters to pass to the constructor for **$class_name** objects.

Example

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';
$host = getenv('MYSQL_HOST') ?: '127.0.0.1';
$port = getenv('MYSQL_PORT') ?: '3306';

$db = new DALMP\Database("utf8://$user:$password@$host:$port/dalmp");

$db->FetchMode('ASSOC');
$ors = $db->map('SELECT * FROM City WHERE Name="Toluca"');

echo sprintf('ID: %d CountryCode: %s', $ors->ID, $ors->CountryCode);

print_r($ors);
```

Output:

```
ID: 2534 CountryCode: MEX

stdClass Object
```

```
(
    [ID] => 2534
    [Name] => Toluca
    [CountryCode] => MEX
    [District] => México
    [Population] => 665617
)
```

See also:

mysqli_fetch_object.

3.5.28 multipleInsert

Performs one query to insert multiple records.

Parameters

```
multipleInsert($table, array $col_name, array $values)
```

$table Name of the table to insert the data.

$col_name Array containing the name of the columns.

$values Multidimensional Array containing the values.

Example

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

$values = array(
    array(1,2,3),
    array(1,3),
    array('date','select', 3),
    array('niño','coraçao', 'Ú'),
    array(null,5,7)
);

$rs = $db->multipleInsert('tests', array('col1', 'col2', 'col3'), $values);
```

Note: The `multipleInsert` method uses Prepared statements PExecute to Insert the data.

3.5.29 PClose

Closes a previously opened database connection, you normally not need to call this method, since DALMP when finishes automatically close all opening connections.

3.5.30 PExecute

Execute an SQL statement using Prepared Statements.

Parameters

```
PExecute($sql, $varN)
```

$sql The MySQL query to perform on the database

$varN The variable(s) that will be placed instead of the **?** placeholder separated by a ',' or it can be the method Prepare.

See also:

SQL syntax prepared statements.

Like the Execute Method, in most cases you probably only use this method when **Inserting** or **Updating** data for retrieving data you can use:

method	Description
PgetAll	Executes the SQL and returns the all the rows as a 2-dimensional array.
PgetRow	Executes the SQL and returns the first row as an array.
PgetCol	Executes the SQL and returns all elements of the first column as a 1-dimensional array.
PgetOne	Executes the SQL and returns the first field of the first row.
PgetASSOC	Executes the SQL and returns an associative array for the given query. If the number of columns returned is greater to two, a 2-dimensional array is returnedwith the first column of the recordset becomes the keys to the rest of the rows. If the columns is equal to two, a 1-dimensional array is created, where the the keys directly map to the values.

Note: Notice that when using "Prepared statements" the methods are prefixed with a **P**.

Examples

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password@127.0.0.1/test";

$db = new DALMP\Database($DSN);

$db->PExecute('SET time_zone=?', 'UTC');
```

An Insert example:

```php
<?php

$db->PExecute('INSERT INTO mytable (colA, colB) VALUES(?, ?)', rand(), rand());
```

An Update example:

```php
<?php

$db->PExecute('UPDATE Country SET code=? WHERE Code=?', 'PRT', 'PRT');
```

Warning: When updating the return value **0**, Zero indicates that no records where updated.

3.5.31 Pquery

Prepared Statements query.

Parameters

```
Pquery($out = array())
```

$out An empty array that will contain the output.

This method is useful, in cases where you need to process each row of a query without consuming too much memory.

Example

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

$rs = $db->PExecute('SELECT * FROM Country WHERE Continent = ?', 'Europe');

$out = array();
while ($rows = $db->Pquery($out)) {
    print_r($out);
}
```

3.5.32 Prepare

Prepare arguments for the Prepared Statements PExecute method.

Parameters

```
Prepare($arg= null)
```

$arg Argument to be used in the PExecute method, if no input it will return the array with the prepared statements.

The prepare method automatically detect the input type, you can also override this, using something like:

```
Prepare('s','1e1')
```

Example

Building dynamic queries that require prepared statements:

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$x = 3;
$id = 1;

$db->Prepare($id)

$sql = 'SELECT * FROM test WHERE id=?';

if ($x == 3) {
    $db->Prepare($x);
    $sql .= 'AND id !=?';
}

$db->Prepare('s', 'colb');

$sql .= 'AND colB=?';

/**
 * this will produce a query like:
 * SELECT * FROM test WHERE id=? AND id !=? AND colB=?
 * with params = ["iis",1,3,"colb"]
 */
 $rs = $db->PgetAll($sql, $db->Prepare());
```

See also:

Prepared statements

3.5.33 qstr

Quotes a string, used when not using prepared statements and want to safetly insert/update data, it uses real_escape_string.

Parameters

```
qstr($string)
```

$string The var to quote.

Example

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

$data = "nbari' OR admin";
$query = $db->qstr($data);

$db->GetRow("SELECT * FROM users WHERE name=$query");
```

Warning: This method will query the database every time is called, so in cases where you are using cache it is not very usefull, since it will need to connect to the database before doing the query.

3.5.34 Query

Fetch a result row as an associative array, a numeric array, or both.

See also:

FetchMode

Example

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

$rs = $db->Execute('SELECT * FROM City');

if ($rs) {
    while (($rows = $db->query()) != false){
        list($r1,$r2,$r3) = $rows;
        echo "w1: $r1, w2: $r2, w3: $r3",$db->isCli(1);
```

```
    }
}
```

Note: Use Pquery when using Prepared statements.

3.5.35 renumber

Some times you lost continuity on tables with auto increment fields, for example instead of having a sequence like : 1 2 3 4 yo have something like: 1 5 18 30; in this cases, the method `renumber('table')` renumbers the table.

Parameters

```
renumber($table, $col='id')
```

$table name of the table to renumber.

$col name of the column with the **auto-increment** attribute.

Example

```
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

$db->renumber('table');
```

Example where uid is the auto-increment column:

```
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

$db->renumber('table', 'uid');
```

See also:

MySQL AUTO_INCREMENT.

3.5.36 RollBackTrans

Rollback the transaction, this must be used in conjunction with method StartTrans.

returns false if there was an error executing the ROLLBACK.

Example

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

$db->Execute('CREATE TABLE IF NOT EXISTS t_test (id INT NOT NULL PRIMARY KEY) ENGINE=InnoDB');
$db->Execute('TRUNCATE TABLE t_test');
$db->FetchMode('ASSOC');

$db->StartTrans();
$db->Execute('INSERT INTO t_test VALUES(1)');
    $db->StartTrans();
    $db->Execute('INSERT INTO t_test VALUES(2)');
    print_r($db->GetAll('SELECT * FROM t_test'));
        $db->StartTrans();
        $db->Execute('INSERT INTO t_test VALUES(3)');
        print_r($db->GetAll('SELECT * FROM t_test'));
            $db->StartTrans();
            $db->Execute('INSERT INTO t_test VALUES(7)');
            print_r($db->GetALL('SELECT * FROM t_test'));
        $db->RollBackTrans();
        print_r($db->GetALL('SELECT * FROM t_test'));
        $db->CompleteTrans();
    $db->CompleteTrans();
$db->CompleteTrans();

if ($db->CompleteTrans()) {
 // your code
}
```

See also:

StartTrans

3.5.37 StartTrans

Start the transaction, for this the mysql table must be of type InnoDB.

Example

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = new DALMP\Database($DSN);

$db->Execute('CREATE TABLE IF NOT EXISTS t_test (id INT NOT NULL PRIMARY KEY) ENGINE=InnoDB');
$db->Execute('TRUNCATE TABLE t_test');
$db->FetchMode('ASSOC');

$db->StartTrans();
$db->Execute('INSERT INTO t_test VALUES(1)');
    $db->StartTrans();
    $db->Execute('INSERT INTO t_test VALUES(2)');
    print_r($db->GetAll('SELECT * FROM t_test'));
        $db->StartTrans();
        $db->Execute('INSERT INTO t_test VALUES(3)');
        print_r($db->GetAll('SELECT * FROM t_test'));
            $db->StartTrans();
            $db->Execute('INSERT INTO t_test VALUES(7)');
            print_r($db->GetALL('SELECT * FROM t_test'));
        $db->RollBackTrans();
        print_r($db->GetALL('SELECT * FROM t_test'));
        $db->CompleteTrans();
    $db->CompleteTrans();
$db->CompleteTrans();

if ($db->CompleteTrans()) {
 // your code
}
```

See also:

CompleteTrans, MySQL START TRANSACTION, COMMIT and ROLLBACK Syntax.

3.5.38 useCache

Set a DALMP\Cache instance to use.

Parameters

```
useCache($cache)
```

$cache A DALMP\Cache instance.

Example

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@localhost/test';

$db = new DALMP\Database($DSN);

$cache = new DALMP\Cache(new DALMP\Cache\Memcache());

$db->useCache($cache);

$rs = $db->CacheGetOne('SELECT now()');

echo $rs, PHP_EOL;
```

See also:

Cache method

3.5.39 UUID

Generates an Universally Unique Identifier (UUID) v4.

Parameters

```
UUID($b=null)
```

$b If true will return the UUID in binary, removing the dashes so that you can store it on a DB using column data type binary(16).

Examples

```php
<?php
...

echo $db->UUID();

echo $db->UUID(true);
```

Example storing UUID as binary(16):

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';
```

```
9
10 $db = new DALMP\Database($DSN);
11
12 $uuid = $db->UUID();
13
14 $db->PExecute("INSERT INTO table (post, uuid) VALUES(?, UNHEX(REPLACE(?, '-', '')))", json_encode($_
```

Example converting from binary(16) to original UUID format chat(36):

```
1  <?php
2
3  require_once 'dalmp.php';
4
5  $user = getenv('MYSQL_USER') ?: 'root';
6  $password = getenv('MYSQL_PASS') ?: '';
7
8  $DSN = "utf8://$user:$password".'@127.0.0.1/test';
9
10 $db = new DALMP\Database($DSN);
11
12 $sql = "SELECT LOWER(CONCAT_WS('-',LEFT(HEX(uuid),8),SUBSTR(HEX(uuid),9,4),SUBSTR(HEX(uuid),13,4),SUB
13
14 $uuids = $db->FetchMode('ASSOC')->getCol($sql);
```

See also:

MySQL string functions.

3.5.40 X

Returns the mysqli object.

Example

```
1  <?php
2
3  require_once 'dalmp.php';
4
5  $user = getenv('MYSQL_USER') ?: 'root';
6  $password = getenv('MYSQL_PASS') ?: '';
7
8  $DSN = "utf8://$user:$password".'@127.0.0.1/test';
9
10 $db = new DALMP\Database($DSN);
11
12 $db->X()->ping();
13 $db->X()->stat();
14
15 $rs = $db->GetOne('SELECT DATABASE()');
16 echo $rs, PHP_EOL;
17
18 $db->X()->select_db('mysql');
19
20 $rs = $db->GetOne('SELECT DATABASE()');
21 echo $rs, PHP_EOL;
```

See also:

class.mysqli.php, ping, stat.

3.6 DALMP\Cache

The `DALMP\Cache` class works as a dispatcher for the current Cache classes, following a common interface in order to maintain compatibility with other **DALMP** classes.

Object interfaces allow you to create code which specifies which methods a class must implement, without having to define how these methods are handled.

See also:

PHP Object Interfaces.

Parameters

```
DALMP\Cache(object)
```

object An CacheInterface instance.

Example

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@localhost/test';

$db = new DALMP\Database($DSN);

$cache = new DALMP\Cache(new DALMP\Cache\Memcache());

$db->useCache($cache);

$rs = $db->CacheGetOne('SELECT now()');

echo $rs, PHP_EOL;
```

See also

3.6.1 CacheInterface

`CacheInterface` is **DALMP** interface to be use with the DALMP\Cache class.

The common methods are:

Method	Description
Delete($key)	Delete item from the server.
Flush()	Flush all existing items at the server.
Get($key)	Retrieve item from the server.
Set($key, $value, $ttl = 0)	Store data at the server.
Stats()	Get statistics of the server.
X()	Return the cache object.

All the cache backends must implement this interface in order to properly work with **DALMP**.

__construct

The construct for each cache backend maybe be different and it is used for defining specific options like the host, port, path, etc.

See also:

PHP Object Interfaces.

3.6.2 APC

Implements the `CacheInteface` using as APC as the cache backend.

Requires APC PECL extension.

3.6.3 Disk

Implements the `CacheInteface` using as the hard disk as the cache backend.

__construct

```
__construct($path)
```

$path Directory to store the cache files - default `/tmp/dalmp_cache`.

Constants

```
define('DALMP_CACHE_DIR', '/tmp/dalmp/cache/');
```

Defines where to store the cache when using 'dir' cache type.

This means that if no $path is passed as an argument to the **__construct** before using the default value will try to get a path from the **DALMP_CACHE_DIR** constant if set.

Example

```
<?php

 require_once 'dalmp.php';

 $cache = new DALMP\Cache\Disk('/tmp/my_cache_path')

 $cache->set('mykey', 'xpto', 300);

 $cache->get('mykey');

 $cache->stats();
```

See also:

Cache Examples.

3.6.4 Memcache

Implements the `CacheInteface` using *memcached <http://memcached.org>* as the cache backend.

Requires Memcache PECL extension.

__construct

```
__construct($host, $port, $timeout, $compress)
```

$host Point to the host where memcache is listening for connections. This parameter may also specify other transports like unix:///path/to/memcache.sock to use UNIX domain sockets - default 127.0.0.1.

$port Point to the port where memcache is listening for connections - default 11211.

$timeout Value in seconds which will be used for connecting to the daemon. Think twice before changing the default value of 1 second - you can lose all the advantages of caching if your connection is too slow.

$compress Enables or disables (true/false) payload compression, Use MEMCACHE_COMPRESSED to store the item compressed (uses zlib) - default false.

Example

```
<?php

 require_once 'dalmp.php';

 $cache = new DALMP\Cache\Memcache('127.0.0.1', 11211, 1, 1);

 $cache->set('mykey', 'xpto', 300);

 $cache->get('mykey');

 $cache->X()->replace('mykey', 'otpx', false, 300);
```

See also:

Cache Examples.

3.6.5 Redis

Implements the `CacheInteface` using as *redis.io <http://www.redis.io* as the cache backend.

Requires REDIS extension.

__construct

```
__construct($host, $port, $timeout)
```

$host Point to the host where redis is listening for connections. This parameter may also specify other transports like unix:///path/to/redis.sock to use UNIX domain sockets - default 127.0.0.1.

$port Point to the port where redis is listening for connections - default 6379.

$timeout Value in seconds which will be used for connecting to the daemon. Think twice before changing the default value of 1 second - you can lose all the advantages of caching if your connection is too slow.

Example

```php
<?php

 require_once 'dalmp.php';

 $cache = new DALMP\Cache\Redis('10.10.10.13', 6379);

 $cache->set('mykey', 'xpto', 300);

 $cache->get('mykey');

 $cache->X()->HSET('myhash', 'field1', 'hello'):

 $cache->X()->HGET('myhash', 'field1');

 $cache->X()->HGETALL('myhash');
```

See also:

Cache Examples.

Note: The **Dalmp\Cache** has no dependency with the DALMP\Database class, this means that you can use only the Database or the Cache classes with out need to depend on eitherone.

3.7 DALMP\Queue

The `DALMP\Queue` class works as a dispatcher for the current Queue classes, following a common interface in order to maintain compatibility with other **DALMP** classes.

Object interfaces allow you to create code which specifies which methods a class must implement, without having to define how these methods are handled.

See also:

PHP Object Interfaces.

Parameters

```
DALMP\Queue(object)
```

object An QueueInterface instance.

Why?

There are times where database go down or you can't Insert/Update data into a table because of the 'too many connections mysql'. In this cases a queue always is useful so that you can later process the queries and not lose important data.

Example

```php
<?php

require_once 'dalmp.php';

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

$DSN = "utf8://$user:$password".'@localhost/test';

$db = new DALMP\Database($DSN);

$queue = new DALMP\Queue(new DALMP\Queue\SQLite('/tmp/dalmp_queue.db'));

/**
 * In case something goes wrong, the database is unavailable, fields missing,  etc,
 * you can save the 'sql query' and later process it again.
 */

 $sql = 'INSERT INTO testX SET colA=(NOW())';
 try {
     $rs = $db->Execute($sql);}
 } catch(Exception $e) {
    $queue->enqueue($sql);
 }
```

See also

3.7.1 QueueInterface

`QueueInterface` is **DALMP** interface to be use with the DALMP\Queue class.

The common methods are:

Method	Description
enqueue($key)	Adds an element to the queue.
dequeue($limit = false)	Dequeues an element from the queue.
delete($key)	Delete an element from the queue.
X()	Return the queue object.

All the queue backends must implement this interface in order to properly work with **DALMP**.

__construct

The construct for each queue backend maybe be different and it is used for defining specific options like the host, port, path etc,

See also:

PHP Object Interfaces.

3.7.2 SQLite

Implements the `QueueInteface` using as SQLite as the queue backend.

Requires PHP SQLite3 support

__construct

```
__construct($filename, $queue_name, $enc_key)
```

$filename Path to the SQLite database, or :memory: to use in-memory database.

$queue_name Name of the queue, defaults to 'default'.

$enc_key The encryption key, default not set.

See also:

For using sqlite3 databases encrypted you need to install sqlcipher: sqlcipher.net.

Example

```
<?php

 require_once 'dalmp.php';

 $queue = new DALMP\Queue(new DALMP\Queue\SQLite('/tmp/dalmp_queue.db'));

 echo 'enqueue status: ', var_dump($queue->enqueue('this is a teste')), PHP_EOL;

 echo 'dequeue all: ', print_r($queue->dequeue(), true), PHP_EOL;

 echo 'dequeue only 3: ', print_r($queue->dequeue(3), true), PHP_EOL;

 echo 'delete from queue: ', var_dump($queue->delete(63)), PHP_EOL;
```

See also:

Queue Examples.

Note: The **Dalmp\Queue** has no dependency with the DALMP\Database class, this means that you can use only the Database or the Queue classes with out need to depend on eitherone.

3.8 DALMP\Sessions

DALMP can store PHP sessions in a mysql/sqlite database or in a cache engine like redis or memcache.

One of the advantage of storing the session on mysql or cache engine is the ability to make your application more scalable, without hassle.

Besides the normal use of sessions, **DALMP** allows the creation of references attached to a session, this means that in some cases you can take advantage of the storage engine that keep the sessions for storing valued information.

The methods you can use to handle references stored on the sessions are:

Method	Description
getSessionsRefs	Return array of sessions containing any reference.
getSessionRef	Return array of sessions containing a specific reference.
delSessionRef	Delete sessions containing a specific reference.

Warning: The Files backend, does NOT support reference handling.

For example, you can store in the reference, the current user id 'UID' and configure your site to only accept users to loggin once avoiding with this duplicate entries/access using the same user/password.

`DALMP\Sessions` implements the SessionHandlerInterface class.

The current available backends are:

Backend	Description
Files	Use file system to store the sessions.
Memcache	Use memcache DALMP\Cache.
MySQL	Use MySQL database DALMP\Database.
Redis	Use redis DALMP\Cache.
SQLite	Use SQLite.

See Also:

3.8.1 __construct

In order to use the `DALMP\Sessions` you need to create an instance of it, while creating the instance you define the backend that will store the sessions and the hash algorithm used to create them.

Parameters

```
__construct($handler = false, $algo = 'sha256')
```

$handler If false uses SQLite, otherwise argument must be an instance of SessionHandlerInterface.

$algo Allows you to specify the hash algorithm used to generate the session IDs - default **sha256**.

The current backends are:

- Files (does NOT support reference handling).
- Memcache.
- MySQL.
- Redis.
- SQLite.

Note: The construct for each cache backend maybe be different and it is used for defining specific options like the host, port, path, etc.

Constants

```
define('DALMP_SESSIONS_MAXLIFETIME', 900);
```

If set, the value is used as an argument for the session.gc_maxlifetime with specifies the number of seconds after which data will be seen as 'garbage' and potentially cleaned up.

```
define('DALMP_SESSIONS_REF', 'UID');
```

The global reference value that will be checked/used when handling sessions, every session will contain this value.

```
define('DALMP_SESSIONS_KEY', '4d37a965ef035a7def3cd9c1baf82924c3cc792a');
```

A unique key that will be used to create the store the session on Memcache/Redis backends, this is useful when working in shared hosted enviroments, basically to avoid collisions.

3.8.2 Files

Handler for storing sessions in local hard disk, implements SessionHandlerInterface.

__construct

```
__construct($sessions_dir = false)
```

$sessions_dir Path to store the sessions, default **/tmp/dalmp_sessions**.

Constants

```
define('DALMP_SESSIONS_DIR', '/tmp/my_sessions');
```

If set and no `$session_dir` defined while initializing the class, it will use this value.

Warning: The **Files** backend, does NOT support reference handling.

3.8.3 Memcache

Handler for storing sessions in memcache, implements SessionHandlerInterface.

__construct

```
__construct(\DALMP\Cache\Memcache $cache, $sessions_ref = 'UID')
```

$cache An instance of DALMP\Cache\Memcache.

$sessions_ref Name of the global reference, defaults to **UID**.

Constants

```
define('DALMP_SESSIONS_REF', 'UID');
```

The global reference value that will be checked/used when handling sessions, every session will contain this value.

```
define('DALMP_SESSIONS_KEY', '4d37a965ef035a7def3cd9c1baf82924c3cc792a');
```

A unique key that will be used to create the store the session on Memcache/Redis backends, this is useful when working in shared hosted enviroments, basically to avoid collisions.

3.8.4 MySQL

Handler for storing sessions in MySQL, implements SessionHandlerInterface.

__construct

```
__construct(\DALMP\Database $DB, $sessions_ref = 'UID')
```

$DB An instance of DALMP\Database.

$sessions_ref Name of the global reference, defaults to **UID**.

Constants

```
define('DALMP_SESSIONS_REF', 'UID');
```

The global reference value that will be checked/used when handling sessions, every session will contain this value.

```
define('DALMP_SESSIONS_TABLE', 'dalmp_sessions');
```

Name of the MySQL table where sessions will be stored, by default the table 'dalmp_sessions' will be used.

MySQL table schema

For storing PHP sessions on mysql you need to create a table with the following schema:

```
CREATE TABLE IF NOT EXISTS `dalmp_sessions` (
`sid` varchar(128) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '',
`expiry` int(11) unsigned NOT NULL DEFAULT '0',
`data` longtext CHARACTER SET utf8 COLLATE utf8_bin,
`ref` varchar(255) DEFAULT NULL,
`ts` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY (`sid`),
KEY `index` (`ref`,`sid`,`expiry`)
) DEFAULT CHARSET=utf8;
```

See also:

DALMP Quickstart.

3.8.5 Redis

Handler for storing sessions in redis, implements SessionHandlerInterface.

__construct

```
__construct(\DALMP\Cache\Redis $cache, $sessions_ref = 'UID')
```

$cache An instance of DALMP\Cache\Redis.

$sessions_ref Name of the global reference, defaults to **UID**.

Constants

```
define('DALMP_SESSIONS_REF', 'UID');
```

The global reference value that will be checked/used when handling sessions, every session will contain this value.

```
define('DALMP_SESSIONS_KEY', '4d37a965ef035a7def3cd9c1baf82924c3cc792a');
```

A unique key that will be used to create the store the session on Memcache/Redis backends, this is useful when working in shared hosted enviroments, basically to avoid collisions.

3.8.6 SQLite

Handler for storing sessions in SQLite, implements SessionHandlerInterface.

__construct

```
__construct($filename = false, $sessions_ref = 'UID', $enc_key = false)
```

$filename Path to the SQLite database, or :memory: to use in-memory database.

$sessions_ref Name of the global reference, defaults to **UID**.

$enc_key The encryption key, default not set.

See also:

For using sqlite3 databases encrypted you need to install sqlcipher: sqlcipher.net.

3.8.7 regenerate_id

The `regenerate_id` method, regenerate a sessions and create a fingerprint, helps to prevent HTTP session hijacking attacks.

Parameters

```
regenerate_id($use_IP = true)
```

$use_IP Include client IP address on the fingerprint.

Example

```
<?php

require_once 'dalmp.php';

$sessions = new DALMP\Sessions();

if ((mt_rand() % 10) == 0) {
    $sessions->regenerate_id(true);
}

$_SESSION['test'] = 1 + @$_SESSION['test'];

echo $_SESSION['test'];

echo session_id();
```

See also:

PHP session_regenerate_id.

3.8.8 getSessionsRefs

Return array of sessions containing any reference.

Example

```
<?php

require_once 'dalmp.php';

$cache= new DALMP\Cache\Memcache('127.0.0.1', 11211, 1, 1);

$handler = new DALMP\Sessions\Memcache($cache, 'ID');

$sessions = new DALMP\Sessions($handler, 'sha512');

$sessions->getSessionsRefs();
```

3.8.9 getSessionRef

Return array of session containing a specific reference.

Parameters

getSessionRef($ref)

$ref Value of the reference to search for.

Example

In this example all the sessions containing the value '3', will returned.

```
<?php

require_once 'dalmp.php';

$cache= new DALMP\Cache\Memcache('127.0.0.1', 11211, 1, 1);

$handler = new DALMP\Sessions\Memcache($cache, 'ID');

$sessions = new DALMP\Sessions($handler, 'sha512');

$sessions->getSessionRef('3');
```

3.8.10 delSessionRef

Delete sessions containing a specific reference.

Parameters

delSessionRef($ref)

$ref Value of the reference to search for.

Example

In this example all the sessions containing the value '3', will be deleted.

```php
<?php

require_once 'dalmp.php';

$cache= new DALMP\Cache\Memcache('127.0.0.1', 11211, 1, 1);

$handler = new DALMP\Sessions\Memcache($cache, 'ID');

$sessions = new DALMP\Sessions($handler, 'sha512');

$sessions->delSessionRef('3');
```

3.8.11 Example

In this example the backend is going to be redis , the global reference name will be **UID**, and the hash algorithim will be **sha512**.

For example, you can store in the reference **UID**, the current user id and configure your site to only accept users to loggin once, avoiding with this duplicate entries/access using the same user/password.

```php
<?php

require_once 'dalmp.php';

$cache= new DALMP\Cache\Redis('127.0.0.1', 6379);

$handler = new DALMP\Sessions\Redis($cache, 'UID');

$sessions = new DALMP\Sessions($handler, 'sha512');

/**
 * your login logic goes here, for example suppose a user logins and has user id=37
 * therefore you store the user id on the globals UID.
 */
$GLOBALS['UID'] = 37;

/**
 * To check if there is no current user logged in you could use:
 */
if ($sessions->getSessionRef($GLOBALS['UID'])) {
    // user is online
    exit('user already logged');
} else {
    $sessions->regenerate_id(true);
}

/**
```

```
 * You can use $_SESSIONS like always
 */
$_SESSIONS['foo'] = 'bar';
```

Warning: In order to properly use `DALMP\Sessions` you need (PHP 5 >= 5.4.0).

3.9 Prepared statements

Prepared statements help you in many cases to avoid avoid mysql injections and helps increasing security of your queries by separating the SQL logic from the data being supplied.

DALMP\Database by default tries to determine the type of the data supplied, so you can just focus on your query without needing to specify the type of data, If you preffer you can manually specify the type of the data. The following table, show the characters which specify the types for the corresponding bind variables:

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

See also:

Method prepare, & mysqli_stmt_bind_param.

To use "Prepared statements" on your SQL statements for retrieving data, the following methods can be used:

Name	Normal	Prepared statements	Cache Normal	Cache Prepared statements
all	GetAll	PGetAll	CacheGetAll	CachePGetAll
assoc	GetAssoc	PGetAssoc	CacheGetAssoc	CachePGetAssoc
col	GetCol	PGetCol	CacheGetCol	CachePGetCol
one	GetOne	PGetOne	PGetOne	CacheGetOne
row	GetRow	PGetRow	PGetRow	CacheGetRow

Any query or either for Inserting or Updating:

Name	Normal	Prepared statements
Execute	Execute	PExecute

Note: Notice that when using "Prepared statements" the methods are prefixed with a **P**.

See also:

Method Cache.

3.9.1 Examples

```
<?php

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';

require_once 'dalmp.php';

```

```
$DSN = "utf8://$user:$password@127.0.0.1/test";

$db = new DALMP\Database($DSN);

$db->PExecute('SET time_zone=?', 'UTC');
```

Example using the LIKE statement:

```
<?php

$sql = 'SELECT Name, Continent FROM Country WHERE Population > ? AND Code LIKE ?';

$rs = $db->FetchMode('ASSOC')->PGetAll($sql, 1000000, '%P%');
```

If you want to define the types, you must pass an array specifying each type. Example:

```
<?php

$sql = 'SELECT * FROM mytable WHERE name=? AND id=?';

$rs = $db->FetchMode('ASSOC')->PGetAll($sql, array('s' => '99.3', 7));
```

An Insert example:

```
<?php

$db->PExecute('INSERT INTO mytable (colA, colB) VALUES(?, ?)', rand(), rand());
```

See also:

Method PExecute

An Update example:

```
<?php

$db->PExecute('UPDATE Country SET code=? WHERE Code=?', 'PRT', 'PRT');
```

Warning: When updating the return value **0**, Zero indicates that no records where updated.

3.10 Dependency injection

DALMP offers a set of classes that allows you to work with MySQL, Cache backends and sessions, but when trying to use all this together, some perfomance issues may be raised, so to deal with this, a DI (dependecy injector) is high recomendable.

The idea of using a DI is to load once, and use any time with out need to reload the objects.

3.10.1 abstractDI

abstractDI is the name of an abstrac class that works as a base for building a dependecy injector.

Note: The abstracDI class can be used as a base for any project not only **DALMP**

3.10.2 DI

DI (Dependecy Injector) extends `abstractDI` and creates the DI for **DALMP**.

3.10.3 Example

Using mysql, cache (redis), sessions.

```php
<?php

require_once 'dalmp.php';

$di = new DALMP\DI();

$user = getenv('MYSQL_USER') ?: 'root';
$password = getenv('MYSQL_PASS') ?: '';
$host = getenv('MYSQL_HOST') ?: '127.0.0.1';
$port = getenv('MYSQL_PORT') ?: '3306';

$DSN = "utf8://$user:$password".'@127.0.0.1/test';

$db = $di->database($DSN);

$cache = $di->cache($redis_cache);

$sessions = $di->sessions($di->sessions_redis($redis_cache), 'sha512');
$sessions->regenerate_id(true);

$db->useCache($cache);

$now = $db->CachegetOne('SELECT NOW()');

echo $now, PHP_EOL;

echo session_id();
```

TODO Examples

3.11 Tests

For testing **DALMP** load the world.sql.gz located at the examples dir:

```
gzcat examples/world.sql.gz | mysql -uroot dalmp
```

You can try also with gzip, gunzip, zcat as alternative to gzcat

That will load all the world tables into the dalmp database and also create the dalmp_sessions table.

For testing purposes the same DSN (same database) is used when testing sessions and database, in practice you can have different DSN depending on your requirements.

You can however configure your DNS:

```
cp phpunit.xml.dist phpunit.xml
```

Edit the DSN section:

```
...
<php>
    <var name="DSN" value="utf8://root@localhost:3306/dalmp" />
</php>
...
```

Install composer and required packages:

```
curl -sS https://getcomposer.org/installer | php -- --install-dir=bin
```

Install phpunit via composer:

```
./bin/composer.phar install --dev
```

For example to test only the Cache\Memcache:

```
./bin/phpunit --testsuite CacheMemcache --tap -c phpunit.xml
```

To run all the tests:

```
./bin/phpunit --tap -c phpunit.xml
```

3.12 Examples

https://github.com/nbari/DALMP/tree/master/examples.

3.13 Issues

Please report any problem, bug, here: https://github.com/nbari/DALMP/issues

3.14 Navicat

Navicat Premium is a multi-connections Database Administration Tool which allows you to bridge up to 6 databases within a single application: MySQL, MariaDB, SQL Server, SQLite, Oracle and PostgreSQL, create a quick and easy access to your databases all at once. Learn more

3.14.1 Thanks

Many thanks Navicat for supporting Open Source projects.