ClickBlocks\DB\DB

General information

Inheritance	no
Child classes	no
Interfaces	no
Source	Framework/db/core/db.php

Class DB is used for interaction with related databases and provides low-level operations with objects of the related databases through PDO extension. Using this class you can get information about a database-specific PDO driver, execute SQL queries and fetch data from a database. The class also provides the means for caching and logging results of queries execution.

Public non-static properties

logging

```
public boolean $logging = false
```

Turns on or off the logging of queries execution results. If this property is TRUE the result of each query execution will be logged in the SQL log file, otherwise it won't. Default value of this property (as well as the path to the log file) can be set in the configuration variable **logging** in section **db**:

```
...
[db]
logging = 1 ; turns on query logging
log     = "Temporary/sql.log" ; sets path to the log file.
...
```

cacheExpire

```
public integer $cacheExpire = 0
```

Determines the default query cache lifetime in seconds. Caching don't happen if value of this property equals 0. If the property value less than 0 then the cache lifetime is equal to the database cache vault lifetime. You can set default value of this property via the configuration file like this:

```
...
[db]
logging     = 1
log         = "Temporary/sql.log"
cacheExpire = -1 ; use the database cache vault lifetime.
...
```

cacheGroup

```
public string $cacheGroup = 'db'
```

Determines the default cache group of all cached queries. Initial value of this property can be specified through the configuration file:

```
...
[db]
logging     = 1
log         = "Temporary/sql.log"
cacheExpire = 3600
```

```
cacheGroup  = "db" ; set initial value of $cacheGroup
...
```

charset

```
public string $charset = null
```

Default charset is used for connections to MySQL or PostgreSQL DBMS.

dsn

```
public string $dsn = null
```

DSN for the default database connection.

username

```
public string $username = null
```

Username for the default database connection.

password

```
public string $password = null
```

Password for the default database connection.

options

```
public array $options = []
```

Options of the default database connection.

Public static methods

getAvailableDrivers()

```
public static array getAvailableDrivers()
```

Returns the list of all currently available PDO drivers. If no drivers are available, it returns an empty array.

getPDOType()

```
public static integer getPDOType(mixed $var)
```

$var	mixed	a PHP variable.

Returns PDO data type of the given PHP variable. These PDO data types are the values of the appropriate PDO constants that represent SQL data types: `PDO::PARAM_INT`, `PDO::PARAM_BOOL`, `PDO::PARAM_STR`, `PDO::PARAM_NULL`.

Public non-static methods

__construct()

```
public void __construct(string $dsn = null, string $username = null, string $password = null, array $options = null)
```

$dsn	string	the DSN for the default database connection.
$username	string	the username for the default database connection.
$password	string	the password for the default database connection.
$options	array	the options of the default database connection.

Constructor of the DB class. Allows to set parameters of the default database connection.

__sleep()

```
public void __sleep()
```

This method is automatically called before serialization process of an object of the class.

__wakeup()

```
public void __wakeup()
```

This method is automatically called after unserialization process of an object of the class.

__get()

```
public PDO|ClickBlocks\DB\SQLBuilder __get(string $param)
```

$param	string	the property name.

This method is used for property overloading. Class DB has two overloaded property: **sql** and **pdo**. The value of **sql** is an instance of class **ClickBlocks\DB\SQLBuilder** that provides unified way to construct complex SQL queries. And the value of **pdo** is an instance of class **\PDO** that represents the internal interface between PHP and database layer.

getCache()

```
public ClickBlocks\Cache\Cache getCache()
```

Returns an instance of the cache class for caching results of SQL queries execution.

setCache()

```
public void setCache(ClickBlocks\Cache\Cache $cache)
```

$cache	ClickBlocks\Cache\Cache	the instance of some caching class.

Sets the instance of the caching class. This class will be used for caching SQL queries.

connect()

```
public void connect(string $dsn = null, string $username = null, string $password = null, array $options = null)
```

$dsn	string	the DSN of the database connection.
$username	string	the username of the database connection.
$password	string	the password of the database connection.

$options	array	the options of the database connection.

Establishes new database connection. If the connection is already set then it will be closed before creating of new connection. If any parameters is not set then the its default value taken from the appropriate public property will be used.

disconnect()

```
public void disconnect()
```

Terminates the current database connection.

isConnected()

```
public boolean isConnected()
```

Returns TRUE if the database connection is established and FALSE otherwise.

getDBName()

```
public string getDBName()
```

Returns the database name of the current connection. If the connection is not set the method returns FALSE.

getHost()

```
public string getHost()
```

Returns the host name of the current connection. If the connection is not set the method returns FALSE.

getPort()

```
public integer getPort()
```

Returns the port of the current connection. If the connection is not set the method returns FALSE.

getDriver()

```
public string getDriver()
```

Returns the driver name of the current database connection and FALSE if the connection is not set.

getDSN()

```
public array getDSN()
```

Returns the full DSN information of the current database connection and FALSE if the connection is not set.

getEngine()

```
public string getEngine()
```

Returns the engine name (DBMS name) for the current connection and FALSE if the connection is not set.

getLastError()

```
public array getLastError()
```

Returns extended error information associated with the last operation on the database handle.

getColumnCase()

```
public mixed getColumnCase()
```

Returns the case of the column names.

setColumnCase()

```
public void setColumnCase(mixed $value)
```

$value	mixed	the case of the column names.

Sets the case of the column names.

getNullConversion()

```
public mixed getNullConversion()
```

Returns how the null and empty strings are converted.

setNullConversion()

```
public void setNullConversion(mixed $value)
```

$value	mixed	the conversion mode.

Sets how the null and empty strings are converted.

getAutoCommit()

```
public boolean getAutoCommit()
```

Returns whether creating or updating a DB record will be automatically committed. Some DBMS (such as sqlite) may not support this feature.

setAutoCommit()

```
public void setAutoCommit(boolean $value)
```

$value	boolean	determines whether to autocommit every single statement.

Sets whether creating or updating a DB record will be automatically committed (available in OCI, Firebird and MySQL).

getPersistent()

```
public boolean getPersistent()
```

Returns whether the connection is persistent or not.

setPersistent()

```
public void setPersistent(boolean $value)
```

$value	boolean	determines whether the connection is persistent or not.

Sets whether the connection is persistent or not.

getClientVersion()

```
public string getClientVersion()
```

Returns the version information of the DB driver.

getConnectionStatus()

```
public string getConnectionStatus()
```

Returns the status of the current connection.

getPrefetch()

```
public boolean getPrefetch()
```

Returns whether the connection performs data prefetching.

getServerInfo()

```
public string getServerInfo()
```

Returns the information of DBMS server.

getServerVersion()

```
public string getServerVersion()
```

Returns the version information of DBMS server.

getTimeout()

```
public integer getTimeout()
```

Returns the timeout settings for the connection.

getAffectedRows()

```
public integer getAffectedRows()
```

Returns the number of rows affected by the last SQL statement.

getLastInsertID()

```
public string getLastInsertID(string $sequenceName = null)
```

$sequenceName	string	name of the sequence object from which the ID should be returned.

Returns the ID of the last inserted row or sequence value.

wrap()

```
public string wrap(string $name, boolean $isTableName = false)
```

$name	string	the table or column name.
$isTableName	boolean	determines whether the first parameter is a table name.

Quotes a table or column name for use in a SQL statement.

quote()

```
public string|array quote(string|array $value, string $format = ClickBlocks\DB\SQLBuilder::ESCAPE_QUOTED_VALUE)
```

$value	string, array	any string or an array of string values.
$format	string	determines the format of the quoted value. This value must be one of the `ClickBlocks\DB\SQLBuilder::ESCAPE_*` constants.

Quotes a value (or an array of values) to produce a result that can be used as a properly escaped data value in an SQL statement. if **$value** is an array then all its elements will be quoted.

beginTransaction()

```
public boolean beginTransaction()
```

Initiates a transaction. Returns TRUE on success or FALSE on failure.

commit()

```
public boolean commit()
```

Commits a transaction. Returns TRUE on success or FALSE on failure.

rollBack()

```
public boolean rollBack()
```

Rolls back a transaction. Returns TRUE on success or FALSE on failure.

inTransaction()

```
public boolean inTransaction()
```

Checks if a transaction is currently active within the driver. Returns TRUE if a transaction is currently active, and FALSE if not.

addPattern()

```
public void addPattern(string $sql, integer|boolean $expire)
```

$sql	string	the PCRE-compatible regular expression.
$expire	integer, boolean	the cache expiration time or FALSE if we don't want to cache the necessary group of queries.

Adds a regular expression that determines group of SQL queries for caching.

dropPattern()

```
public void dropPattern(string $sql)
```

$sql	strring	the regular expression pattern.

Removes a regular expression that determines the group of SQL queries to cache.

execute()

```
public mixed execute(string $sql, array $data = [], string $type = self::EXEC, integer $mode = \PDO::FETCH_ASSOC, mixed $arg = nul
```

$sql	string	a SQL to execute.
$data	array	the input parameters of the SQL.
$type	string	type execution of the query. This parameter affects format of the data returned by the method.
$mode	integer	fetch mode for this SQL statement.
$arg	mixed	this argument have a different meaning depending on the value of the **$mode** parameter.
$ctorargs	array	arguments of custom class constructor when the **$mode** parameter is `PDO::FETCH_CLASS`.

Executes the SQL statement and returns result of its execution.

Available values of **$type** lists in the class constants:

- `DB::EXEC` - the method returns the affected rows.
- `DB::CELL` - the method returns a single value.
- `DB::COLUMN` - the method returns a numeric array of values of a database table column.
- `DB::ROW` - the method returns values of a single table row.
- `DB::ROWS` - the method returns values of a few rows.
- `DB::COUPLES` - the methods returns an array of a few rows values where the first column is a key and the other columns are the values.
- `NULL` - if this parameter is not defined the method returns an instance of **ClickBlocks\DB\Reader** class that allow to iterate all fetched rows.

cell()

```
public string cell(string $sql, array $data = [], integer $column = 0)
```

$sql	string	a SQL query to execute.
$data	array	the input parameters of the SQL.
$column	array	0-indexed number of the column you wish to retrieve from the row.

Executes the SQL query and returns the value of the given column in the first row of fetched data.

column()

```
public string column(string $sql, array $data = [], integer $column = 0)
```

$sql	string	a SQL query to execute.
$data	array	the input parameters of the SQL.
$column	integer	0-indexed number of the column you wish to retrieve from the row.

Executes the SQL query and returns the given column values of the result.

row()

```
public array row(string $sql, array $data = [], integer $mode = PDO::FETCH_ASSOC)
```

$sql	string	a SQL query to execute.
$data	array	the input parameters of the SQL.
$mode	integer	the fetch mode for this SQL statement.

Executes the SQL query and returns the first row of the result.

rows()

```
public array rows(string $sql, array $data = [], integer $mode = PDO::FETCH_ASSOC, mixed $arg = null, array $ctorargs = null)
```

$sql	string	a SQL query to execute.
$data	array	the input parameters of the SQL.
$mode	integer	the fetch mode for this SQL statement.
$arg	mixed	this argument have a different meaning depending on the value of the **$mode** parameter.
$ctorargs	array	arguments of custom class constructor when the **$mode** parameter is `PDO::FETCH_CLASS`.

Executes the SQL query and returns all fetched rows.

pairs()

```
public array pairs(string $sql, array $data = [])
```

$sql	string	a SQL query to execute.
$data	array	the input parameters of the SQL.

This method is similar to the method **rows** but returns a two-column result into an array where the first column is a key and the second column is the value.

groups()

```
public array groups(string $sql, array $data = [], integer $mode = PDO::FETCH_ASSOC)
```

$sql	string	a SQL query to execute.
$data	arrray	the input parameters of the SQL.
$mode	intege	the fetch mode for this SQL statement.

Executes the SQL query and returns all rows which grouped by values of the first column.

couples()

```
public array couples(string $sql, array $data = [], integer $mode = PDO::FETCH_ASSOC, mixed $arg = null, array $ctorargs = null)
```

$sql	string	a SQL query to execute.

$data	array	the input parameters of the SQL.
$mode	integer	the fetch mode for this SQL statement.
$arg	mixed	this argument have a different meaning depending on the value of the **$mode** parameter.
$ctorargs	array	arguments of custom class constructor when the **$mode** parameter is `PDO::FETCH_CLASS`.

Executes the SQL query and returns all rows into an array where the first column is a key and the other columns are the values.

query()

```
public ClickBlocks\DB\Reader query(string $sql, array $data = [], integer $mode = PDO::FETCH_ASSOC, mixed $arg = null, array $ctor
```

$sql	string	a SQL query to execute.
$data	array	the input parameters of the SQL.
$mode	integer	the fetch mode for this SQL statement.
$arg	mixed	this argument have a different meaning depending on the value of the **$mode** parameter.
$ctorargs	array	arguments of custom class constructor when the **$mode** parameter is `PDO::FETCH_CLASS`.

Returns an instance of class **ClickBlocks\DB\Reader** that implements an iteration of rows from a query result set.

getTableList()

```
public array getTableList(string $schema = null)
```

$scheme	string	the scheme (database name) of the tables.

Returns the table list of the given database. If the database is not specified the current database is used.

getTableInfo()

```
public array getTableInfo(string $table)
```

$table	string	the table name.

Returns the table metadata.

getColumnsInfo()

```
public array getColumnsInfo(string $table)
```

$table	string	the table name.

Returns metadata of the table columns.

insert()

```
public integer insert(string $table, mixed $data, array $options = null)
```

$table	string	the table name.

$data	mixed	the columns data.
$options	array	contains additional parameters that make sense for some DBMS. For example: `$options['updateOnKeyDuplicate']` for MySQL. If this boolean parameter is specified and a row is inserted that would cause a duplicate value in a UNIQUE index or PRIMARY KEY, an UPDATE of the old row is performed. The another parameter is `$options['sequenceName']` - name of the sequence object (required by PostgreSQL and Oracle DBMS).

Inserts new record in the database table. The method returns the ID of the last inserted row or sequence value.

update()

```
public integer update(mixed $table, mixed $data, mixed $where = null)
```

$table	mixed	information about the database tables whose rows are to be updated.
$data	mixed	information about updating columns.
$where	mixed	information about conditions of the updating.

Updates existing rows in the database. The method returns the number of rows affected by this SQL statement.

delete()

```
public integer delete(mixed $table, mixed $where = null)
```

$table	mixed	information about the database tables whose rows are to be deleted.
$where	mixed	information about conditions of the deleting.

Deletes existing rows in the database. The method returns the number of rows affected by this SQL statement.

Protected non-static properties

idsn

```
protected array $idsn = []
```

DSN information of the current connection.

patterns

```
protected array $patterns = []
```

Contains regular expression patterns for query caching.

cache

```
protected ClickBlocks\Cache\Cache $cache = null
```

An instance of **ClickBlocks\Cache\Cache** class for query caching.

affectedRows

```
protected integer $affectedRows = null
```

Contains a number of affected rows as the result of the last operation.

engines

```
protected $engines = ['mysql'   => 'MySQL',
                      'mysqli'  => 'MySQL',
                      'pgsql'   => 'PostgreSQL',
                      'sqlite'  => 'SQLite',
                      'sqlite2' => 'SQLite',
                      'mssql'   => 'MSSQL',
                      'dblib'   => 'MSSQL',
                      'sqlsrv'  => 'MSSQL',
                      'oci'     => 'OCI']
```

The mapping between PDO drivers and database engines.

Protected non-static methods

prepare()

```
protected void prepare(PDOStatement $st, string $sql, array $data)
```

$st	PDOStatement	the instance of a SQL statement.
$sql	string	the SQL query to execute.
$data	array	the data for the SQL query.

Prepares a SQL statement to execution.

assemble()

```
protected string assemble(string $sql, array $data)
```

$sql	string	the parametrized SQL query.
$data	array	the SQL data to be combined.

Combines the parametrized SQL query string and data in one string.