Описание методов

**Класс class.\_pdo.php**

create

public static function create ($dbdriver = *DB\_DRIVER*, $login = *DB\_LOGIN*, $password = *DB\_PASSWORD*, $dbname = *DB\_NAME*, $hostorsock = *DB\_SOCKET*, $port = *DB\_PORT*)

**Описание:**

Singleton для объекта класса - статический метод, возвращающий объект класса \_PDO. По умолчанию параметры берут из соответствующих констант. Объявление которых может содержаться, например, в конфигурационном файле.

**Параметры:**

* *$dbdriver* - драйвер доступа к СУБД. На данный момент поддерживаются СУБД MySQL и PostgreSQL, допустимые значения: pgsql, mysql;
* *$login* - логин пользователя для доступа к базе данных;
* *$password* - пароль доступа к базу данных;
* *$dbname* - имя базы данных, к которой мы подключаемся;
* *$hostorsock* - имя, ip-адрес хоста или UNIX-сокет для подключения к базе данных;
* *$port* - порт, на котором БД слушает подключения.

**Пример использования:**

$dbconnect = \_PDO::create($dbdriver);

getDBDriver

public function getDBDriver ()

**Описание:**

Возвращает имя текущего драйвера подключения к БД.

**Параметры:** нет.

**Пример использования:**

$driver = $dbconnect->getDBDriver();

getDBH

public function getDBDriver ()

**Описание:**

Вернет объект подключения к базе данных.

**Параметры:** нет.

**Пример использования:**

$dbh = $dbconnect->getDBH();

query

public function query($query, array $params = [])

**Описание:**

Выполняет запрос к БД и возвращает результат. Поддерживает регулярные выражения.

**Параметры:**

* *$query* - текст запроса;
* *$params* - параметры запроса (для prepared-запросов).

**Пример использования:**

$dbconnect->query("SELECT table\_name FROM information\_schema.tables WHERE table\_schema = 'public'");

beginTransaction

public function beginTransaction ()

**Описание:**

Стартует транзакцию. Не выбрасывает исключение если транзакция стартуется повторно.

**Параметры:** нет.

**Пример использования:**

$dbconnect->beginTransaction();

commit

public function commit ()

**Описание:**

Коммитит транзакцию. Не выбрасывает исключение если открытой транзакции нет.

**Параметры:** нет.

**Пример использования:**

$dbconnect->commit();

rollBack

public function rollBack ()

**Описание:**

Откатывает транзакцию. Не выбрасывает исключение если нет открытой транзакции.

**Параметры:** нет.

**Пример использования:**

$dbconnect->rollBack();

getTables

private function getTables ($query)

**Описание:**

Возвращает имена таблиц, использующихся в запросе в виде массива.

**Параметры:**

* *$query* - текст запроса.

**Пример использования:**

$tables = $dbconnect->getTables(“SELECT \* FROM table”);

getEditTables

public function getEditTables ($query)

**Описание:**

Если запрос является запросом на изменение, то возвращает участвующие в запросе таблицы, иначе возвратит FALSE.

**Параметры:**

* *$query* - текст запроса.

**Пример использования:**

$tables = $dbconnect->getEditTables(“INSERT INTO table VALUES (1, 2, 3)”);

// table

parallelExecute

public function parallelExecute(array $batch)

**Описание:**

Параллельно выполняет запросы из массива запросов, переданного единственным параметром. Возвращает Массив некорректно отработавших запросов.

**Параметры:**

* *$batch* - массив запросов.

**Пример использования:**

$failed = $dbconnect->parallelExecute([“INSERT INTO table VALUES (1, 2, 3)”, “UPDATE table2 SET field1 = ‘value1’, field2 = ‘value2’]);

createQStrFromBatch

private function createQStrFromBatch (array $batch)

**Описание:**

Формирует строку для асинхронного выполнения методами asyncBatch и execBatch.

**Параметры:**

* *$batch* - массив запросов.

**Пример использования:**

$str = $dbconnect->dbh->exec($this->createQStrFromBatch($batch));

asyncBatch

public function asyncBatch(array $batch)

**Описание:**

Отправляет асинхронно пакет запросов на сервер. Использует php-расширение PGSQL.

**Параметры:**

* *$batch* - массив запросов.

**Пример использования:**

$dbconnect->asyncBatch([‘INSERT INTO test VALUES (1, 21)’,

‘INSERT INTO test VALUES (1, 22)’,

‘INSERT INTO test VALUES (1, 23)’,

‘INSERT INTO test VALUES (3, 71)’]);

execBatch

public function execBatch (array $batch)

**Описание:**

Выполнить пакет транзакций с проверкой результата выполнения. Если во время выполнения пакета запросов произошла ошибка метод выкинет исключение.

**Параметры:**

* *$batch* - массив запросов.

**Пример использования:**

$dbconnect->execBatch([‘INSERT INTO test VALUES (1, 21)’,

‘INSERT INTO test VALUES (1, 22)’,

‘INSERT INTO test VALUES (1, 23)’,

‘INSERT INTO test VALUES (3, 71)’]);

Как использовать

$dbconnect = \_PDO::create();

$params = [param1 => true, param2 = false];

$query = "INSERT INTO

test

(param1, param2)

VALUES

(:param1, :param2)

RETURNING

param1,

param2";

$result = $dbconnect->query($query, $params);

где $result - результат выполнения запроса.