Ark Database Documentation

Release 0.1.0

Liu Dong

Nov 24, 2017

Contents

Contents:

CHAPTER 1

Introduction

Ark Database is an efficient PHP library to work with databases.

1.1 What's included

- Connection manager
- Query builder
- Model
- Model factory

1.2 Supported Drivers

- Mysql
- Sqlite
- Postgresql

1.3 Requirements

- PHP5.4+
- PDO

1.4 Installation

It's recommended to install with composer:

```
composer require ark/database
```

CHAPTER 2

Quick Start

2.1 Create Database Connection

First of all, create a database connection with DSN, username and password. Take the Mysql Database for example:

```php
<?php
use Ark\Database\Connection;

$db = new Connection('mysql:host=localhost;dbname=testdb', 'username', 'password');
```

2.2 Query Builder

```php
<?php
// Query
$db->builder()
    ->select('*')
    ->from('user')
    ->where('age > :age and created_at > :time', [
        ':age' => 20,
        ':time' => time() - 3600
    ])
    ->limit(10)
    ->queryAll();

// Insert
$db->builder()
    ->insert('user', [
        'name' => 'user1',
        'password' => 'pa$$word',
    ]);
```

2.3 Work with Model

```php
<?php
// Create model factory
$factory = $db->factory('@user');

// Insert
$factory->insert([
    'name' => 'user1',
    'age' => 20,
]);

// Get model
$user = $factory->findOneByName('user1');

// Update
$user->email = 'user1@example.com';
$user->save();

// Delete
$user->delete();
```

CHAPTER 3

Database Connection

Connection is the fundamental of this library.

3.1 Setup Connection

```php
<?php
use Ark\Database\Connection;

$db = new Connection($dsn, $username, $password, $options);
```

`$username` and `$password` are not required for Sqlite.

3.2 DSN

The Data Source Name, or DSN, contains the information required to connect to the database.

For more information, refer to the PDO manual.

3.2.1 Mysql

```
mysql:host=localhost;port=3333;dbname=testdb
```

3.2.2 Sqlite

```
sqlite:/path/to/db
```

3.3 Database Options

The connection can be customized with the `$options` param. Commonly used options are listed below:

- `prefix`: Table prefix, use {{table}} as table name, prefix will be prepended automatically.
- `auto_slave`: Use slave connections when perform read only operations. (Note that it only works for query builder and model, not the connection it self)
- `reconnect`: Reconnect automatically in case of `Mysql gone away`, default value is false.
- `reconnect_retries`: Times to retry when lose connection, default value is 3.
- `reconnect_delay_ms`: Milliseconds to wait before try to reconnect, default is 1000.
- `PDO::ATTR_ERRMODE` Error reporting. Possible values:
  - `PDO::ERRMODE_SILENT`: Just set error codes.
  - `PDO::ERRMODE_WARNING`: Raise E_WARNING.
  - `PDO::ERRMODE_EXCEPTION`: Throw exceptions.
- `PDO::ATTR_TIMEOUT`: Specifies the timeout duration in seconds. Not all drivers support this option, and its meaning may differ from driver to driver. For example, sqlite will wait for up to this time value before giving up on obtaining an writable lock, but other drivers may interpret this as a connect or a read timeout interval. Requires int.

For more options, refer to the PDO manual.

3.4 Usage

The connection instance can act just like class PDO. For example, you can exec a SQL statement:

```
<?php
$db->exec("INSERT INTO user (username, email) VALUES ('test', 'test@localhost')");
```

Or you can query some results with a SQL statement:

```
foreach ($db->query("SELECT * FROM user") as $user) {
    echo $user['username']."\n";
}
```

Additionaly, it provide two important shortcut methods for you to work with Query Builder and Model Factory:

```
<?php
$builder = $db->builder();
$factory = $db->factory('@user');
```

3.5 Multiple Connections

You can add configurations for more than one connection, and switch between them.

```
<?php
$db = new Connection($dsn, $username, $password, $options);
$db->addConnection('connection2', $dsn, $username, $password, $options);
```

```
$db->addConnection('connection3', $dsn, $username, $password, $options);

$db->switchConnection('connection2');
$db->switchConnection('connection3');
$db->switchConnection('default');
```

A useful usecase is "auto slave", you can setup replication for your database, and then use the *auto_slave* option to improve performance.

```
<?php
$db = new Connection($dsn, $username, $password, [
    'auto_slave' => true
]);
$db->addConnection('slave', $dsnSlave, $username, $password);

// insert into master(default connection)
$db->builder()
    ->insert('user', $userdata);

// query is performed at slave database automatically
$db->builder()
    ->select('name')
    ->from('user')
    ->queryValue();
```

CHAPTER 4

Query Builder

Query builder makes it easier to working with SQL statements.

4.1 Select

Query builder provide a chainable API to build SQL query like a charm.

```
<?php
$users = $db->builder()
    ->select('username, email')
    ->from('user')
    ->where('level > :level', array(':level' => $level))
    ->limit(10)
    ->orderBy('username ASC')
    ->queryAll();
```

In addition to `queryAll` to fetch all rows, you can use:

- `queryRow` to get the first row
- `queryValue` to get the first column of first row (the value)
- `queryColumn` to get the first column of all rows

4.1.1 Parameter Binding

Both named placeholder and question mark placeholder are supported:

```
<?php
$db->builder()
    ->where('username = ? and password = ?', [$name, $password]);

$db->builder()
```

```
    ->where('username = :name and password = :password', [
        ':name' => $name,
        ':password' => $password
    ]);
```

4.1.2 Condition

TODO

4.2 Insert

```
<?php
$db->builder()
    ->insert('user', array(
        'userame' => 'test',
        'email' => 'test@example.com',
    ));
```

4.3 Update

```
<?php
$db->builder()
    ->update(
        'user',
        array(
            'email' => 'test1@example.com',
        ), // columns to be updated
        'username = :username', // conditions
        array(':username' => $username) // params
    );
```

4.4 Delete

```
$db->builder()
    ->delete(
        'user',
        'username = :username',
        array(':username' => $username)
    );
```

CHAPTER 5

Model

Define a model by extending the class `Ark\Database\Model`. You need to override the `config` method to specify table name, pk or relations of your model.

```
<?php
use Ark\Database\Model;
class User extends Model {
    static public function config() {
        return array(
            'table' => 'user',
            'pk' => 'uid',
        );
    }
}
```

To be convienient, it's not necessary to create a class for a simple model. You'll see that in the next section.

CHAPTER 6

Model Factory

With model factory, you can manipulate data in object-oriented (or ORM) way.

6.1 Get the Factory

First of all, declare which model you will be working with.

```
<?php
// Model factory for User class
$factory = $db->factory('User');

// Or for a table without model class created, just use "@{table_name}" as the model!
$factory = $db->factory('@user');
```

6.2 Find

```
<?php
//Get all users
$users = $factory->findAll();
foreach($users as $user){
    echo $user->email;
}

//find(findByPK)
$user = $factory->find(1);

//findByXX
$users = $factory->findByAge(30);

//findOneByXX
$user = $factory->findOneByUsername('user1');
```

6.3 Update

```php
<?php
//create a model
$user = $factory->create();

//and then set data
$user->username = 'user2';
$user->email = 'email2';

//You can also create this model with initial data
$user = $factory->create(array(
    'username' => 'user2',
    'email' => 'email2',
));

//save model
$user->save();

//delete model
$user->delete();
```

CHAPTER 7

Indices and tables

- genindex
- modindex
- search