SimpleObject Documentation

Release latest

Feb 01, 2023

Contents

SimpleObject is a simple lightweight ORM based on ActiveRecord pattern.

CHAPTER 1

Contents

1.1 Introduction

1.2 Installation

SimpleObject should be installed using Composer, which is a tool for dependency management in PHP. Please visit the Composer website for more information.

To install SimpleObject require it using Composer:

```
php composer.phar require sanovskiy/simple-object
```

1.3 Configuration

At first you must initialize database connection. You can make this by calling SanovskiySimpleObjectUtil::init($options, $configName) method for each connection you use

Simple exanple:

```
require __DIR__.DIRECTORY_SEPARATOR.'..'.DIRECTORY_SEPARATOR.'vendor'.DIRECTORY_
↪SEPARATOR.'autoload.php';

use Sanovskiy\SimpleObject\Util;

try {
    Util::init([
        'dbcon' => [
            'driver' => 'mysql',
            'host' => 'localhost',
            'user' => 'root',
            'password' => '',
```

(continues on next page)

(continued from previous page)

```
            'database' => 'test',
            'charset' => 'utf8'
        ],
        'path_models' => '/full/path/to/models/directory',
        'models_namespace' => 'project\\models\\'
    ], 'default');

} catch (\Throwable $e) {
    die('Something went wrong. '.$e->getMessage());
}
```

After this you can access PDO connection directly through Util::getConnection('default')

Note: You can omit 'default' connection name because SimpleObject uses 'default' as the default connection name.

If you planning to use separate connections for read and write (master-slave database setup) can specify different connections for reading and writing

```
$config = [
    'default'      => [
        'dbcon'            => [
            'driver'   => 'mysql',
            'host'     => 'localhost',
            'user'     => 'root',
            'password' => '',
            'database' => 'test',
            'charset'  => 'utf8'
        ],
        'path_models'      => '/full/path/to/models/directory',
        'models_namespace' => 'project\\models\\',
        'read_connection'  => 'default_read'
    ],
    'default_read' => [
        'dbcon'            => [
            'driver'   => 'mysql',
            'host'     => 'localhost',
            'user'     => 'root',
            'password' => '',
            'database' => 'test',
            'charset'  => 'utf8'
        ],
        'path_models'      => '/full/path/to/models/directory',
        'models_namespace' => 'project\\models\\',
        'write_connection' => 'default'
    ],
];
foreach ($config as $name => $c) {
    Util::init($c, $name);
}
```

1.4 Generating models

Just call Util::reverseEngineerModels(bool $silent) to generate models for all configs initialized. If you supply TRUE as parameter, generator will not make any info output except errors.

1.5 Models

1.6 Structure

1.7 CRUD

Index