
DotArray

Release 1.0

unknown

Jun 20, 2019

CONTENTS

1	User Guide	3
1.1	Overview	3

DotArray is a simple PHP Library that can access an array in a dotted manner.

- Easy to use.
- Support to access complex keys that are having dot in the name.
- Fluent access.
- Possibility to search & filter items.
- Dot access can, also, works in the “old school” array way (Vanilla PHP).

```
DotArray::create(['config' => ['some.dotted.key' => 'value']])->get('config.{some.  
↪dotted.key}')
```


1.1 Overview

1.1.1 Requirements

1. PHP 7.1

1.1.2 Installation

The recommended way to install DotArray is with [Composer](#). Composer is a dependency management tool for PHP that allows you to declare the dependencies your project needs and installs them into your project.

```
# Install Composer
curl -sS https://getcomposer.org/installer | php
```

You can add DotArray as a dependency using the composer.phar CLI:

```
php composer.phar require binary-cube/dot-array
```

Alternatively, you can specify DotArray as a dependency in your project's existing composer.json file:

```
{
  "require": {
    "binary-cube/dot-array": "*"
  }
}
```

After installing, you need to require Composer's autoloader:

```
require 'vendor/autoload.php';
```

You can find out more on how to install Composer, configure autoloading, and other best-practices for defining dependencies at [getcomposer.org](#).

1.1.3 License

Licensed using the [MIT license](#).

Copyright (c) 2018 Binary Cube

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without

limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.1.4 Bugs and feature requests

Have a bug or a feature request? Please first read the issue guidelines and search for existing and closed issues. If your problem or idea is not addressed yet, [please open a new issue](#).

1.1.5 Contributing

All contributions are more than welcomed. Contributions may close an issue, fix a bug (reported or not reported), add new design blocks, improve the existing code, add new feature, and so on. In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation. [Read the full Code of Conduct](#).

1.1.6 Versioning

Through the development of new versions, we’re going use the [Semantic Versioning](#).

Example: *1.0.0*. - Major release: increment the first digit and reset middle and last digits to zero. Introduces major changes that might break backward compatibility. E.g. *2.0.0* - Minor release: increment the middle digit and reset last digit to zero. It would fix bugs and also add new features without breaking backward compatibility. E.g. *1.1.0* - Patch release: increment the third digit. It would fix bugs and keep backward compatibility. E.g. *1.0.1*