Blocks

Release 1.0.0

Feb 16, 2020

Contents

CHAPTER 1

Welcome to Blocks

1.1 Welcome

Blocks is a very minimal PHP framework for building web applications. So why another PHP framework? Why not use the other frameworks available? I found that most frameworks are too large and contain too many dependencies or might be too micro and will have to install other dependencies. So with me not liking to work with most PHP frameworks , I decided to develop a minimal framework which is not too large and not too small either.

1.2 Who is Blocks for?

Blocks is right for you if:

- you want a framework with a small footprint
- you want a framework with very few configurations

1.3 Requirements

- PHP 7.* or newer

1.4 How to read this documentation

If you are new to Blocks, I recommend you read this documentation from start to finish.This documentation begins by explaining Blocks's concepts and architecture before venturing into specific topics like controllers and models, routing, and error handling.

CHAPTER 2

Installation

2.1 Manual Installation

The Blocks framework repository holds the released versions of the framework.

Develop your app inside the app folder, and the public folder will be your public-facing document root. Do not change anything inside the system folders!

2.2 Via Composer Create-Project

Alternatively, you may also install Blocks by issuing the Composer create-project command in your terminal:

```
composer create-project blocks/framework projectName
```

2.3 Installation

Download the latest version , and extract it to become your project root.

2.4 Setting Up

- Download and install Composer
- Download and install Node js
- Run `npm install` (Installing command-line tool (CLI))
- Run `composer install`
- In **gulpfile.js** on **line 61** change proxy from **http://localhost/blocks** to **http://localhost/<your directory name>**

- In **.htaccess** in the **public** directory change **line 3** from **RewriteBase /blocks/public** to **RewriteBase /<your directory name>/public**

CHAPTER 3

Running Your App

This would not be suitable for app development, but is suitable for contributing to the framework.

3.1 Initial Configuration & Set Up

1. Open the app/Config/app.php with a text editor and set your URLROOT. The default value of URLROOT is set to to 'localhost/blocks'.
2. If you intend to use a database, open the app/Config/database.php file with a text editor and set your database settings.

3.2 CLI

Blocks comes Blocks comes with command-line tool(CLI) which helps you in many operations. For more information on the commands click here.

3.3 Local Development Server

Blocks comes with command-line tool which will help you set up a development server. This server also minifies all of your css,scss and js codes.

`npx blocks start` OR `npx blocks s`

CHAPTER 4

Blocks Overview

A fresh install of blocks has eight directories /app, /bin, /database, /public, setup and possibly /docs, /node_modules and /vendor

/app directory:

```
/Config         Stores the configuration files
/Controllers    Controllers determine the program flow
/Core           Contains Blocks main classes
/Helpers        Helpers store collections of standalone functions
/Models         Models work with the database to represent the business entities.
/Views          Views make up the HTML that is displayed to the client.
```

/bin directory:

```
/scripts         Contains scripts to be used by the frameword
```

/database directory

Contains all migrations created

/public directory

Contains all files that should be accessible in the browser. This is where your CSS,JS and images should be placed

/setup directory Contains database and migrations configuration files

/docs directory Contains the documentation

/vendor directory Contains all PHP dependencies

/node_modules directory Contains all node dependencies used by the CLI

CHAPTER 5

Controllers

Controllers handles routing in Blocks. Controllers are class files that are associated to the URL. Consider the URL below:

```
http://localhost/blocks/home/sayName/adams/asad
```

From this URL Blocks tries to find the controller **home** and call the method **sayName**. **adams** and **asad** here are extra parameters passed to the controller.

5.1 A simple Controller

So let's create a very simple controller and let's see how it works. Create a file About.php and enter in the following code, then save it to **app/Controllers** directory.

Note: Is a good coding convention start all classes with an upper case. Therefore **About.php**

```
1  <?php
2
3  class About extends Controller {
4
5          public function index() {
6                  echo "This is the about page";
7          }
8  }
```

See also:

All controllers must extend the **Controller** class

Now if you should visit:

```
http://localhost/blocks/home
```

OR:

```
http://localhost/blocks/home/index
```

You would see:

```
This is the about page
```

Note: Index methods are automatically called

5.2 Passing params from URI to Controller

If your url contains more than 2 segments the rest will be passed to the controller as parameters:

```
http://localhost/blocks/home/index/adams
```

adams is therefore passed as a parameter and accepted in the controller as shown below

```
1  <?php
2
3  class About extends Controller {
4
5          public function index($name) {
6                  echo "My name is".$name;
7          }
8  }
```

CHAPTER 6

Views

View is just a web page you would want to display. This can be a header or a footer etc. Views can be loaded in a controller.

Using the example controller you created in the controller page, let's add a view to it.

6.1 Creating a View

Let's create a simple view and load it with the controller we have already created. Using a text editor, create a file called **dashboard.php** and save it to **app/Views** directory

```
    <html>
            <head>
                    <title> My Dashbord</title>
            </head>
            <body>
                    <h1Dashboard</h1>
                    <p>Welcome to your dashboard</p>
            </body>
    </html>
```

6.2 Loading a View

To load the view in our **index** method in the controller

```
$this->view('dashboard');
```

Note: You don't need the .php extension

6.3 Error Handling

Blocks let you show 3 error pages by default, these are 404,401 and 500 pages. Blocks automatically shows a 404 page if a controller is not found. You can load a an error page in a controller by loading a view of 404,401 or 500

```
$this->view('404');
```

The default templates of these error pages are located at **app/Core/Exceptions/Views**

CHAPTER 7

Models

Blocks uses Eloquent ORM from laravel to make modeling of data very easy.

7.1 A simple Model

So let's create a very simple **User** Model. Create a file User.php and enter in the following code, then save it to **app/Models** directory.

Note: Is a good coding convention start all classes with an upper case. Therefore **About.php**

Note: We are assuming you have a **user** table in your database

```
<?php
namespace Models;

use Illuminate\Database\Eloquent\Model as Eloquent;

class User extends Eloquent {

        protected $table = 'user';

}
```

See also:

All Models must extend the **IlluminateDatabaseEloquentModel** class and should be namespaced **Models**

Read Eloquent Docs

7.2 Migrations

In software engineering, schema migration (also database migration, database change management) refers to the management of incremental, reversible changes and version control to relational database schemas. A schema migration is performed on a database whenever it is necessary to update or revert that database's schema to some newer or older version. Migrations are performed programmatically by using a schema migration tool. Source: Wikipedia.

Blocks can make Database migrations only with it's command-line tool (CLI) Blocks-cli.

7.3 Resources

Please read these resources to be able to use migrations and models

- Eloquent ORM
- Schema Building
- Writing Migrations

CHAPTER 8

Helpers

Helpers are a collection of functions.

Helpers	Description
Session	Helper functions related to Sessions
Str	Helper functions related to Strings
Url	Helper functions related to URL

8.1 Sessions Helper

The following are the static methods available:

```
Session::isSet( [$key=''] )
```

Check if session is set

Parameters:

- $key (string) – Session's key

Returns: a Boolean

Return type: Boolean

```
Session::set( [$key='', $value=''] )
```

Method for setting a session

Parameters:

- $key (string) – Key for session
- $value (string) – Value to store in session

Returns: void

Return type: void

```
Session::get( [$key=''] )
```

Getting value in a sesion

Parameters:

- $key (string) – session to get

Returns: Session's value | null

Return type: String | null

```
Session::remove( [$key=''] )
```

Removing session

Parameters:

- $key (string) – session to get

Returns: void

Return type: void

8.2 String Helper

The following are the static methods available:

```
Str::uuid()
```

Generates a uuid

Returns: a uuid

Return type: String

```
Str::encrypt_password( [$string=''] )
```

Encrypts a string password

Parameters:

- $string (string) – String to encrypt

Returns: an encrypted password

Return type: String

```
Str::validate_password( [$password1='',$password2=''] )
```

Validating if password is correct

Parameters:

- $password1 (string) – Validating if this matches to password2
- $password2 (string) – Validating if this matches to password1

Returns: if password matches

Return type: Boolean

8.3 URL Helper

The following are the static methods available:

```
Url::redirect( [$page=''] )
```

redirecting to a page

Parameters:

- $page (string) – Page to redirect you to

Returns: void

Return type: void

```
Url::secure_public( {$path=''} )
```

Generates a fully secured qualified URL to the public directory or a path

Parameters:

- $path (string) – The path

Returns: secure URL to public directory or a path

Return type: String

```
Url::public( [$path=''] )
```

Generates a fully qualified URL to the public directory or a path

Parameters:

- $path (string) – The path

Returns: URL to public directory or a path

Return type: String

CHAPTER 9

The MIT License (MIT)