
Thutong Site Learning Center User Manual

Contributors:

Fiwa Lekhulani
Daniel Rocha
lebogang Ntatleng
Lesego Mabe
Tlou Lebelo
Oluwatosin Botti

Contents

| | | |
|----------|--|----------|
| 1 | General | 1 |
| 1.1 | File Header | 1 |
| 1.2 | Description of Classes | 1 |
| 2 | Coding Conventions | 2 |
| 2.1 | Naming Conventions | 2 |
| 2.2 | Formatting Conditions | 2 |
| 2.2.1 | line breaks | 2 |
| 2.2.2 | Indentation and Alignment | 2 |
| 2.2.3 | In-Code comment Conventions | 2 |
| 3 | Laravel and it's coding Stnadards | 3 |

This document contains the coding conventions created by the Digital Blacksmiths team, based on the various languages used to create the Thutong Learning Centre.

As the team makes use of Laravel Web Framework it makes use of PSR-4, PSR-4, PHP Code Sniffer and Style CI, which all helps simplify and ensure that the code is of good quality, readable and easy to maintain.

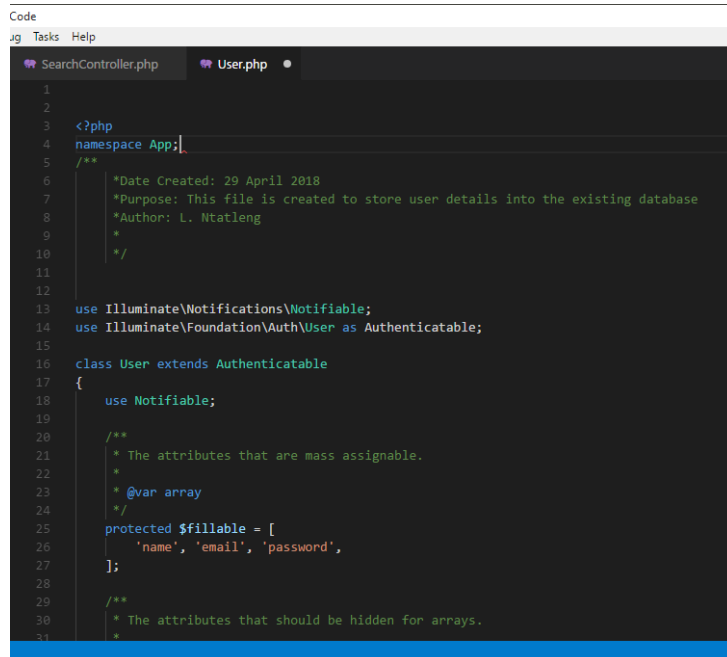
1 General

1.1 File Header

Each file has a description of what is coded within that specific file, whom it was made and/or edited by. This to make any other programmer of the core purpose of the file.

1.2 Description of Classes

This is a description of what a selected class and it's function implements and how to do so as seen in Figure 1.



```
Code
ug Tasks Help
SearchController.php User.php
1
2
3 <?php
4 namespace App;
5 /**
6  *Date Created: 29 April 2018
7  *Purpose: This file is created to store user details into the existing database
8  *Author: L. Ntatleng
9  *
10 */
11
12
13 use Illuminate\Notifications\Notifiable;
14 use Illuminate\Foundation\Auth\User as Authenticatable;
15
16 class User extends Authenticatable
17 {
18     use Notifiable;
19
20     /**
21      * The attributes that are mass assignable.
22      *
23      * @var array
24      */
25     protected $fillable = [
26         'name', 'email', 'password',
27     ];
28
29     /**
30      * The attributes that should be hidden for arrays.
31      *
```

Figure 1: File header and description

2 Coding Conventions

2.1 Naming Conventions

All variable, file and function names should be easy to understand yet descriptive for what its purpose is. Names should also use camel casing so they are more descriptive and easy to read.

1. Classes
e.g. `class StudentDetails ... ;`
2. Functions and Methods
e.g. `getStudentId();`
3. Variables
e.g. `var studentId;`

2.2 Formatting Conditions

Formatting Conditions are rules used to arrange program statements.

2.2.1 line breaks

- Programmers should avoid lines that are more than 150 characters; if so proceed to a new line.

2.2.2 Indentation and Alignment

- Lines of code that are not dependant on each other should be aligned. function or loop implementations are indented and those are aligned with each other.
- Indentation should be used to make code more clear, easy to read, understand, enhance and maintain.
- One line should be indented a minimum of one tab space or four spaces.

2.2.3 In-Code comment Conventions

1. Block Comments
 - Block comments are used to provide descriptions of files, methods, data structures and algorithms. They are usually used in file headers.

- A block comment should be preceded by a blank line to set it apart from the rest of the code.

2. Single-line Comments

- Short comments can appear on a single line indented to the level of the code that follows.
- They can describe the purpose of a variable or why a particular action is being done.

3. Multi-line Comments

- Multi-line comments are multiple lines of comments for the same use as single-line comments.

```
/*
don't use the global isFinite() because it returns true for null
values
*/
Number.isFinite(value)
```

```
/**
 * Register a binding with the container.
 *
 * @param string[] $abstract
 * @param \Closure|string|null $concrete
 * @param bool $shared
 * @return void
 */
public function bind($abstract, $concrete = null, $shared = false)
{
    //
}
```

Figure 2: Multi-line comment

Figure 3: Block and descriptive commenting

3 Laravel and it's coding Standards

Laravel makes use of PSR-2 Coding Standards which give rules as to the most ideal way to implement coding standards for PHP.

These can be found at: [StyleCI](#)

Laravel also makes use of StyleCI. Which will automatically merge any style fixes into the Laravel repository after pull requests are merged on Github. This allows us to focus on the content of the contribution and not the code style.

StyleCI take whatever rules you choose to have implemented in your coding standards and checks the code via terminal or Travis if it meets the requirements set out; if a file has errors it will make the appropriate changes for you.