**Laravel Installation Guidelines**

1. **Install Xampp**
2. **Install Composer**

* **During the installation of the composer, it requires the *php.exe,* this can be directed to the xampp -> php -> php.exe**
* **After the composer has completely installed, can test it on command prompt by enter ‘composer’ then enter. If there is a list displayed out, this means the composer has been successfully installed.**

1. **Configure the path**

* **Open the system properties and environment variables, the add in:**
* **Variable: PHP\_HOME ; Value:D\xampp\php\**
* **Variable: Path ; Value: %PHP\_HOME%**

1. **Create table**

* **Type in the command to create table in database:**
* **Php artisan make:migration create\_name\_table –create=”name”**

1. **Meet Composer**

* **Download the composer and install it.**
* **In the command prompt, type this (composer require phpspec/phpspec)**
* **Create a new project, type this (composer create-project laravel/laravel your-project-name --prefer-dist)**
* **Set the directory of the project, type this (php –S localhost:8888 –t public)**

1. **Virtual Machines and Homestead**

* **Install Vagrant and VirtualBox**
* **After install both of these, type in (vagrant box add laravel/homestead)**
* **Then type in (composer global require “laravel/homestead=~2.0”)**

1. **A gentle introduction to routing, controllers, and views**

* **Eg: Route::get(‘/’, ‘WelcomeController@index’);**

**~ This is the home page, so there is only ‘/’ on the front of the get route only**

**~ get is like hit the page**

**~ WelcomeController is the file and index is the method name**

* **Eg: Route::get(‘contact’, ‘WelcomeController@contact’);**

**~ the contact at the front is the path name**

**~ The method of the contact is:**

**Public function contact()**

**{ return view (‘pages.contact’); }**

**~ pages.contact means folder -> pages and file -> contact.blade**

**~.blade == .html**

1. **Passing Data to Views**

* **We can create PagesController using the command prompt, type in**

**(php artisan make:controller PagesController) if you want a plain text in the controller, type this (php artisan make:controller PagesController --plain)**

* **First method: On the method, create a variable and pass it to the view page**

**Eg:**

**Public function contact**

**{**

**$name = ‘Abc’;**

**return view (‘pages.contact’)->with (‘name’ , $name);**

**}**

**Second method: Array method, multiple way**

**Eg:**

**Public function contact**

**{**

**return view (‘pages.contact’)->with ([**

**‘first’ => ‘Sam’ ,**

**‘last’ => ‘Son’**

**]);**

**}**

**Third method:**

**Eg:**

**Public function contact**

**{**

**$first = ‘a’;**

**$last = ‘b’;**

**return view (‘pages.contact’ , compact(‘first’ , ‘last’));**

**}**

**Then at the view, type this to display the pass data**

**First method:**

**Eg: <?= $name; ?>**

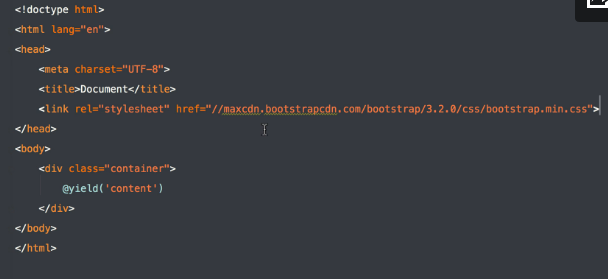
**Second method:**

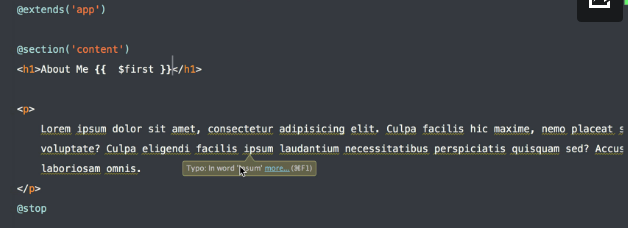
**Eg: {{ $name }}**

**Third method:**

**Eg: {!! $name !!}**

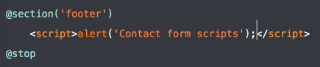
1. **Blade 101**

* **This is like the master page,**
* **First we need to add in @yield(‘’) in the blade file (master) page**
* ****
* **For the child file, add in @extends(‘app’) and @section**
* **Eg:**

****

**-If we want to add in a javascript alert, we can add a new @yield and create a section in the child file**

**Eg:**

****

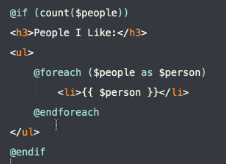
* **We also can add a condition on child file that display the transferred data,**

**Eg:**

****

* **The condition consist of if, foreach, forelse**
* **The following example show if there is a people in the controller class (master class), it will display out, else it won’t display it out**

**Eg:**

****

1. **Environments and Configuration**

* **This is regarding to configure the database. We can change the database name on the (.env) file.**

1. **Migrations**

* **This show how to create a table and update using the command prompt.**
* **There are 2 methods, up and down.**
* **First, type in (*php artisan migrate*) in the command prompt. This method is to migrate the existing or updated table that contain in the migration folder to the database.**
* **When want to rollback, type in (*php artisan migrate:rollback*)**
* **In order to create a new migration table, type in**

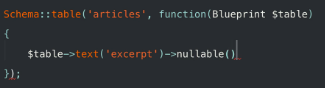
**(*php artisan make:migration create\_articles\_table -–create=”articles”*)**

* **If the database in in production and cannot change it by rollback, we need to type in**

**(*php artisan make:migration add\_excerpt\_to\_articles\_table –table=”articles”* )**

* **If we want to set the column to nullable, add in nullable();**

**Eg:**

****

* **In case when you rollback and an error pop out stated that no class found, need to type this to install the class file (*composer require doctrine/dbal*)**

1. **Eloquent 101**

* **We will create a model according to the table in the database.**
* **We will type in (*php artisan make:model Article*) to create a model for table Article.**
* **In order to add in the data to the database, we need to type in ( *php artisan tinker* ).**
* **Then, type in ( *$article = new App\Article;* ) to create an article object**
* **Add in data is like ( $article->title = ‘My First Article’)**

**( *$article->published\_at = Carbon\Carbon::now*(); ) -> this is the current date and time**

* **If want to view the current inserted data, type in ( *$article->toArray();* )**
* **Save the data into database, ( *$article->save();* )**
* **If want to view all the data, type in ( *App\Article::all()->toArray();* )**
* **If we want to update the data in the database, type in ( *$article->title = ‘My updated article’;* ), then save it.**
* **If we want to update the data according to the condition, type in**

**( *App\Article::find(1);* ) to locate the find first. Then type in the code as mentioned above.**

* **If use where instead of find method, we type in**

**( *$article = App\Article::where(‘body’, ‘Abc’)->get();* )**

* **If want to use the array method to update the data in database, type in this:**

**( *$article->update([‘body’ => ‘Update Article’]);* )**

* **There is an another way to insert the data into database through array, type in**

***( $article = App\Article::create([‘title’ => ‘New Article’, ‘body’ => ‘New body’, ‘published\_at’ => Carbon\Carbon::now()]);* ).**

**Mass exception will throw if you enter this code, to solve this, we need to add some code inside the Article file.**

****

1. **Basic Model/Controller/View Workflow**

* **First, create a new controller and this is the code**

**( *php artisan make:controller ArticlesController --plain* )**

* **Look for the articles->index.blade and ArticlesController to see the implementation**

1. **Forms**

* **First, we need to import a form using the command prompt, type in**

**( composer require illuminate/html )**

* **Add in ('Illuminate\Html\HtmlServiceProvider' ) in the Config -> app.php under the providers**
* **Add in 'Form' => 'Illuminate\Html\FormFacade',**

**'Html' => 'Illuminate\Html\HtmlFacade',**

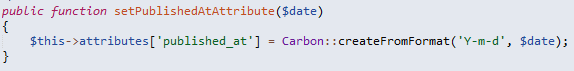
**under the aliases.**

* **The implementation can be reference at create.blade, form.blade and store method in ArticlesController**

1. **Dates, Mutators, and Scopes**

* **In order to get the current date and time, we need to add in one method in Article model class file**

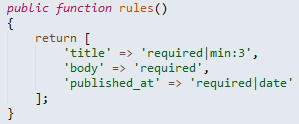
**Eg:**

****

1. **Form Requests and Controller Validation**

* **This is to make a validation on the form so that it will perform checking before the form is submit into the database.**
* **Add in a ArticleRequest.php in Http->Requests folder.**

**Eg:**

****

* **Then call the method at ArticlesController’s store method**

**Eg:**

****

1. **Eloquent Relationship**

* **This tutorial is about add in a relation between the user and article. A user might has many articles and one article is belongs to one user. So, we need to add in foreign key in the create article table and refresh the table all over again.**
* **Eg:**

**First, type in ( *php artisan migrate:refresh* ) in the command prompt**

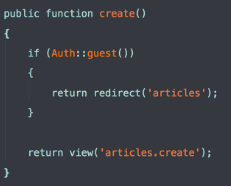
**Then, type in ( *php artisan tinker* )**

* **$user = new App\User;**
* **$user->username = ‘name’;**
* **$user->email = ‘email’;**
* **$user->password = bcrypt(‘password’);**
* **$user->save();**

1. **Oges like a middleware**

* **Allow authorized person to access into create article**

**Eg:**

****

* **There is another method, go to the Http->Requests->Kernel.php**

**~ $middleware is any middle that put inside this method will be run on every single requests.**

**~ $routeMiddleware attach the middleware on specific route**

1. **Midterm Review**

* **$fillable is use to protect the data from being attack by the bad people**
* **$dates is use to create a column for carbon instances**

1. **Flash Messaging**

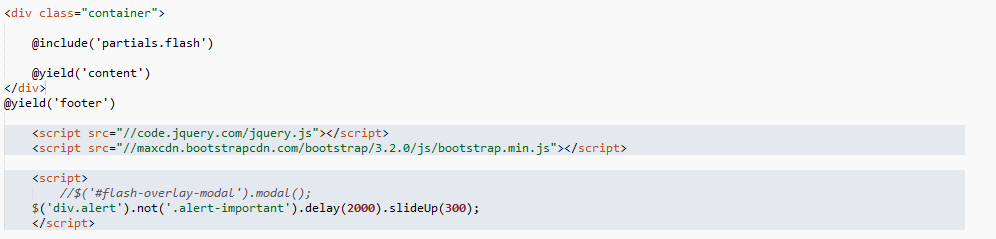
* **Session flash – temporary**
* **Session put – Permanent**
* **First, we need to add in the session code inside the create method in the article controller**

**Eg:**

****

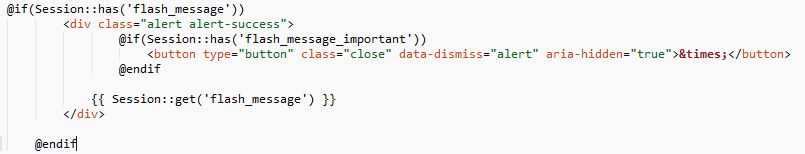
**Then, we need to add in bootstrap stype and javascript inside the app.blade(master page)**

**Eg:**

****

**After that, create a flash.blade to include the method inside**

**Eg:**

****

1. **Many to many relations**

* **Create a model name tag using command prompt**

**( *php artisan make:model name* )**

* **Create tags table**

**( *php artisan make:migration create\_tags\_table –create=tags* )**

1. **Syncing Tags**

* **Attach() – Add new row to the pivot table**
* **Detach() – Remove the data from the row of the pivot table**
* **Sync() – Sync everything up**

1. **Enhancing Select Elements**

* [**https://select2.github.io/**](https://select2.github.io/) **this is the link for style of multi box selection**
* **<link href="//cdnjs.cloudflare.com/ajax/libs/select2/4.0.0-beta.3/css/select2.min.css" rel="stylesheet" />**
* **<script src="//cdnjs.cloudflare.com/ajax/libs/select2/4.0.0-beta.3/js/select2.min.js"></script>**
* **These 2 links above need to be included inside the app.blade page. Master page**
* **There a ways that store the css style and the scripts in different folder and reference it, I didn’t apply it yet.**

1. **When you want a view partial to always receive data**

* **This is the code for the navigation bar**

**Eg:**

**<nav class="navbar navbar-inverse navbar-fixed-top">**

**<div class="container">**

**<div class="navbar-header">**

**<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">**

**<span class="sr-only">Toggle navigation</span>**

**<span class="icon-bar"></span>**

**<span class="icon-bar"></span>**

**<span class="icon-bar"></span>**

**</button>**

**<a class="navbar-brand" href="#">Project name</a>**

**</div>**

**<div id="navbar" class="collapse navbar-collapse">**

**<ul class="nav navbar-nav">**

**<li class="active"><a href="#">Home</a></li>**

**<li><a href="#about">About</a></li>**

**<li><a href="#contact">Contact</a></li>**

**</ul>**

**</div><!--/.nav-collapse-->**

**</div>**

**</nav>**

* **The navigation method in the AppServiceProvider can be separate into a new create service provider.**
* **I didn’t do it because there are some steps look blur and I need to understand it in detail**

**Build Your First App**

1. Install npm

* Install it and run ( node –v) in command prompt
* On the project’s command prompt -> type ( *npm install express*)
* Change the directory of C and D drive, type (d: or c:)

1. Install gulp

* Install the gulp in the system, open system command prompt -> type:

( *npm install –g hulp*)

* On the project folder, type in this code in command prompt

( *npm install –save-dev gulp* )

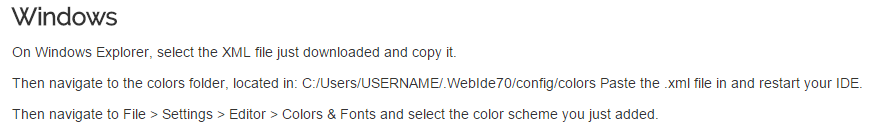
**Php Storm IDE**

1. **Install the laravel IDE helper**

**Download at** [**https://github.com/barryvdh/laravel-ide-helper**](https://github.com/barryvdh/laravel-ide-helper)

1. **Add theme to php storm manually**

* **Download the raw file and save it to .xml format**

****

**Twitter Bootstrap**

1. **<div class=”container col-md-8 col-md-offset-2>”**

* If you have a column size of 10, the offset will be 1
* If you have a column size of 8, the offset will be 2
* If you have a column size of 6, the offset will be 3
* If you have a column size of 4, the offset will be 4
* If you have a column size of 2, the offset will be 5