Simple for me to understand

# A) Blade::

- go to views and create a new file (master) : app.blade.php (here we will have html5 full --)

<html>

<head>

<link rel = “stylesheet” href = “https://www.getbootstrap.com/... “>

</head>

<title> </title>

### <body>

### @yield('content') //i want my content to be yeilded here

### </body>

Inside views eg. Home.blade.php files:

@extends(‘app’) //meaning we will extend to the app master page

So we want the content to be used:

@section(‘content’) or for any divisions we can give name like this

@stop

@section(‘footer’)

<script> alert(‘hello world’); </script>

@stop

# B) Include css:

- inside master blade page:

//css file needs to be inside public/assests/css

<link rel = “stylesheet” href = {{ asset(‘assets/css/bootstrap.css’) }} />

Or

<href = {{ asset(‘/css/your\_name.css’) }}

## c) doing simple conditional blade:

$people = [ ‘ram’ , ‘krishna’ , ‘shyam’]; // then return view

return view(‘pages.contact’ , compact(‘people’);

//inside our pages.contact:

@if (count($people))

<ul> @foreach ($people as $person)

<li> {{ $person }} </li>

@endforeach

</ul>

@endif

# C) Environment:

* have a look at .env and config folder once
* we need not give the host name and db password in config
* we can easily get it throught .env file and remember .env is ignored inside .gitignore file
* inside config/database.php u can easily choose the driver for your database either mysql , postgresql , or sqlite
* Choose the default one
* Inside the .env file give the name of the database , username and the password

# D) Migration:

* it’s like the version control for db
* We need not extract the db and then send it to others ..
* There are two methods up and down
* Run existing migration: php artisan migrate (default given migration: users)
* Eg. I want to change column user to username:
  + First roll back : php artisan migrate:rollback

Change the value to username : see below

$table->string('username'); //which was previously string(‘name’);

* + Then: php artisan migrate

## Creating New Mirgration:

* U can see the help as: php artisan help make:migration
* Make table:
  + Syntax: php artisan make:migration (name \_of\_migration) –create=”(table\_name)”
  + php artisan make:migration create\_articles\_table --create="articles" // table = “name” for existing table
* u can see this migration just made inside database/migrations
* Let’s do one table for articles:: (it’s similar how we used in PHPMyAdmin)
* Schema::*create*('articles', **function**(Blueprint $table)  
   {  
   $table->increments('id');  
   $table->string('title');  
   $table->text('body');  
   $table->timestamps();  
   $table->timestamp('published\_at');
* });
* For Down method use this:
* **public function** down()  
  {  
   Schema::*drop*(name\_table);  
  }
* Then: migrate it using: php artisan migrate
* U want to add new column to that table:
  + php artisan make:migration add\_excerpt\_to\_articles\_table --table=”articles”

# E) Eloquent:

* active record implementation of laravel
* If I have Articles table then eloquent model is called “Article” , if (table Users , model is user)
* So , let’s make model article:
  + Php artisan make:model Article // since Article is model name , the migration directly takes place , coz it easily gets the table Articles from our DB
  + You will see this model as the child of the App directory ie. App/Article.php ( you will get it)
* Tinker: it gives good command line interface , for working with Laravel codebase , we can do general php and more…
* How to assign values to the column of table:
* First create a new object:
  + $article = new App\Article;
  + $article->title = ‘My First Article’;
  + $article->body = ‘Lorem ipsum …. ‘;
  + //don’t worry for timestamp
  + $article->published\_at = Carbon\Carbon::now(); //Carbon is library used by laravel
* You can view that using $article->toarray();
* Saving the data to database:
  + $article->save();
  + //returns true if correct
* Fetching all the data:
  + App\Article::all->toArray(); //fetch everything of Article model i.e Articles table and give it to array;
* If you want to update it: simple overwrite in the given object:
  + $article->title = ‘new title’;
  + $article->save();
  + $article->toArray();

## Selecting from table using eloquent (SELECT \* FROM with id)

* + $article = App\Article::find(1); //find the data with id 1 from the Model
  + $article->toArray(); //see the result

### Select from table with variable value (SELECT FROM table with value = ‘kkjk’)

* + $article = App\Article::where(‘body’ ,’this is my body’)->get(); // I want to get the result
  + I get the collection with this

## Select from table with variable the very first result:

* + $article = App\Article::where(‘body’ ,’jkjkjk’)->first();

## Faster Eloquent Way:

### Saving the data :

* $article = App\Article::create([ ‘title’ => ‘my title’ , ‘body’ => ‘my body’ , ‘published\_at’ => Carbon\Carbon::now() ]); //we send the data as a associative array
* //Mass Assignment this is very very important thing Laravel is protecting us so just see it :P
* So go to App\Article.php , we will make some code hereby:
  + Eg. Protected $fillable = [ ]; //array is created , which defines which attributes can be mass assigned //fillable field for the article or which attribute I am ok being mass assigned , id cant be included similarly user\_id
* **protected** $fillable = [  
   'title',  
   'body',  
   'published\_at'  
  ];

## Update in Faster Way:

* Get the data:
  + $article = App\Article::find(2); //get the data with id = 2;
  + $article->body = ‘updated one’;
* Next method (similar to create): //first u need to get the id
  + $article->update([‘body’ => ‘updated body’]);

# F) Model View Controller Workflow:

* We are working with the articles:

### Route:

* //For our Articles  
  Route::*get*('articles' , 'ArticlesController@index');
* We need to have articlescontroller

### Controller:

**php artisan make: controller ArticlesController –plain**

* Inside the controller:
  + Import model Article on top by ::> **use App\Article;**
* **public function** index()  
  {  
   $articles = Article::*all*();  
    
   **return** $articles;  
  }
* in the above case JSON returns all the parsed data … so we want to make it clean …. Check this we will updating the same index function inside the ArticlesController
* **So we will pass this data to a view ok ?**
* **public function** index()  
  {  
   $articles = Article::*all*();  
    
   **return** view(‘articles.index’ , compact(‘articles’); //we will send $articles array to view: Views/Articles/index.blade.php  
  }

### View:

* Inside Articles/index.blade.php
  + First extend to master page
  + Use the sections
* **@section(**'content'**)** <h1> Articles </h1>  
   <hr>  
   **@foreach(**$articles **as** $article**)** <article>  
   <h2> {{ $article->title }} </h2>  
   <div class = "body"> {{ $article->body }} </div>  
   </article>  
   **@endforeach  
   @stop**
* We need to give link to the titles:

<h2><a href="articles/{{ $article->id }}">

article->title }}</a></h2>

### Passing id through articles :: public/articles/2

* Go to routes
* Route::*get*('articles/{id}' , 'ArticlesController@show');
* Inside the pages controller new function is to be made:
* **public function** show($id)  
  {  
   $article = Article::*findorfail*($id);  
   **return** view('articles.show',compact('article'));  
  }
* }
* We need to check for the show.blade.php also:
* **@extends(**'app'**)  
    
  @section(**'content'**)** <h1>{{ $article->title }} </h1>  
   <article>  
   {{ $article->body }}  
   </article>  
    
  **@stop**

# G) Creating new Article (use of form and save data)

* Route:
* Route::*get*('articles/create' , 'ArticlesController@create');
* Controller:
* **public function** create()  
  {  
   **return** view('articles.create');  
  }

# Giving Link to any pages:

<a href={{ action('ArticlesController@create') }}>Create</a>  
<a href={{ action('PagesController@contact') }}>Contact Me! </a>

## Form Package: html

* composer require illuminate/html (illumate is used by laravel itself)
* you can check these components at: github.com/illuminate/html

### Service Providers:

* + Go to config/app.php … there is big list of service providers
  + Add to list
* 'Illuminate\Html\HtmlServiceProvider',

### Aliases:

* + Scroll down to Aliases list :
  + Aliases gives the sort of global way to access the namespaces
  + Add new item for Form and HTML to last:
* 'Form' => 'Illuminate\Html\FormFacade',  
  'Html' => 'Illuminate\Html\HtmlFacade',

## Form:

## Opening and closing a form:

After using these things now we can work on with form so lets just try:

* In the beginning the after opening the form we see that it sends request to the same page:

{!! Form::*open*() !!}

{!! Form::close() !!}

### Adding a nice form:

* {!! Form::*open*() !!}  
   <div class="form-group">  
   {!! Form::*label*('title' ,'Title:') !!}   
   {!! Form::*text*('title' ,**null** , ['class' => 'form-control' , 'placeholder' => 'Give the title' ,]) !!} //null means value , name is first one  
   </div>  
    
   <div class="form-group">  
   {!! Form::*label*('body' ,'Body:') !!}  
   {!! Form::*textarea*('body' , **null** , ['class' => 'form-control']) !!}  
   </div>  
    
   <div class="form-group"> //no nid 2 give name for btn  
   {!! Form::*submit*('Write Article' , ['class' => 'btn btn-primary form-control']) !!}  
   </div>  
    
    
  {!! Form::*close*() !!}

# Posting data:

Since we are posting the data we need to now do something let’s check it very easy

### Routes:

* Route::*post*('articles' , 'ArticlesController@store'); //send post req to articles page

But our form by default sends the req to current page, so we will have to change it also:

Inside the ArticlesController we need a method to show the data passed by the form:

Firstly add: @ top of COntroller

**use** Request;

View all the data:

**public function** store()  
{  
 $input = Request::*all*(); //this gives all

$title = Request::get(‘title’);  
 **return** $input;  
}

### Changing the form Action:

{!! Form::*open*(['url' => 'articles']) !!}

## Creating a new Post:

**public function** store()  
{  
 $input = Request::*all*();  
 $input['published\_at'] = Carbon::*now*();  
  
 Article::*create*($input); //we had other ways also  
 **return** redirect('articles');  
}

### View the latest one: inside index function

$articles = Article::*latest*()->get();