**Первый route в Laravel**

Запуск сервера в терминале **php artisan serve**

Нахождение роутеров **App -> routes -> web**

Route:: **get**(url/ ‘posts’,**function**(){

**Return** ‘this is my page’;

}) ;

**Первый controller в Laravel**

**App -> HTTP -> controller**

Создание контроллера **php artisan make:controller MyPlaceController**

Изменение в Routes

Rute::get(url’/posts’ , ‘PostController@index’);

Записываем в controller ()

public function index() {

return ‘this is my page’; // вырезаем из routs

}

Ошибка Target class [MyPlaceController] does not exist

App ->Providers->RouteServiceProvider.php

Строка 29. //protected $namespace = ‘App\\Http\\Controllers’; (убрать коментирование)

**Первая миграция в Larabel**

Запускаем php сервер

заходим phpMyAdmin имя root пароль нет

В php MyAdmin нажимаем Создать БД

Пишем Имя БД Любое Кодировка utf8mb4\_general\_ci Создать

Открыть MySQL Workbench

Нажимаем на плюсик. Далее пишем имя любое TestConfiguration

Далее заходим в созданный аккаунт

Заходим в Laravel

App->database-> migrations

Далее создадим модель сразу прикреплённая к миграции

Php artisan make:model Post –m

Далее заходим в миграцию CreatePostTable

Schema::create('posts', function (Blueprint $table) {

            $table->id(); //Создаёт колонку с индивидуальным числом атрибута

            $table->timestamps(); // создаст 2 колонки creat ad(время когда объект заноситься в базу) uddate ad(время обновления объекта)

        });

Далее создаём колонки которые нужны для проекта.

$table->id();

            $table->string('title');

            $table->text('content');

            $table->string('image')->nullable();

            $table->integer('likes')->nullable();

            $table->boolean('is\_published')->default(1);

            $table->timestamps();

Далее накатываем миграцию

Команда php artisan migrate

Ошибка Unknown database ‘larave’ -> app->.env

DB\_DATABASE=my\_bd 14 строка (меняем название своей таблицы)

Ошибка cold not find driver

Заходим php.ini' фаил и раскоментируем extension=php\_pdo\_mysql.dll

**Модель. Метод чтения данных из базы (READ)**

Создание дампа

App->Controller->PostController

Заходим в PostController

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PostController extends Controller

{

    public function index() {

        $str = "string";

        var\_dump($str);

    }

Вывод:

string(6) "string"

Показывает Тип, сколько символов и сам текст.

Создание дампа, при котором после вывода останавливается программа и подсвечивает место созданного кода.

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PostController extends Controller

{

    public function index() {

        $str = "string";

        dd($str);

    }

}

Вывод:

"**string**"

Создание дампа, эта функция делает тоже самое что преведущая запись, но не останавливает код.

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PostController extends Controller

{

    public function index() {

        $str = "string";

        dump($str);

    }

}

Создание первого поста.

Заходим в базу данных и заводим данные в таблицу

Далее нажимаем apply применить

**Вытягивание данных из базы**

Что бы вытянуть данные, нужно сначало обратиться к классу Post:: функция find и id

Импортирование класса

use App\Model\Post;

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\Post;

class PostController extends Controller

{

    public function index() {

        $post = Post::find(1);

        dd($post);

    }

}

Вывод

App\Models\Post {#624▼

#connection: "**mysql**"

#table: "**posts**"

#primaryKey: "**id**"

#keyType: "**int**"

+incrementing: **true**

#with: []

#withCount: []

+preventsLazyLoading: **false**

#perPage: **15**

+exists: **true**

+wasRecentlyCreated: **false**

#escapeWhenCastingToString: **false**

#attributes: array:8 [▼

"id" => **1**

"title" => "**Мой первый пост**"

"content" => "**Как я создавал свой первый пост**"

"image" => "**32423**"

"likes" => **324**

"is\_published" => **1**

"created\_at" => **null**

"updated\_at" => **null**

]

#original: array:8 [▶]

#changes: []

#casts: []

#classCastCache: []

#attributeCastCache: []

#dates: []

#dateFormat: **null**

#appends: []

#dispatchesEvents: []

#observables: []

#relations: []

#touches: []

+timestamps: **true**

#hidden: []

#visible: []

#fillable: []

#guarded: array:1 [▶]

}

Что бы достучаться с любого атрибута можно указать ->

class PostController extends Controller

{

    public function index() {

        $post = Post::find(1);

        dd($post->title);

    }

Вывод:

"**Мой первый пост**"