В этой части добавим загрузку изображений и внесем изменение в действие контроллера по их сохранению, переименованию и добавлению информации в таблицу БД.

Во-первых, необходимо пояснить, что все сохранения в Laravel по умолчанию производятся в папку storage, которая не доступна из web, поэтому чтобы мы могли вставлять картинки, сохраненные таким способом, нам необходимо создать символьную ссылку на определенную папку, или использовать действия по передачи файлов каждый раз, где это необходимо. Но мы используем способ создания нового драйвера диска. Таких драйверов мы можем сделать сколько угодно: папка public, ftp, различные облачные хранилища.

Добавим в файл config\filesystems.php в массив 'disks' => [] настройки драйвера my\_files:

'my\_files' => [

'driver' => 'local',

'root' => public\_path() . "",

],

Теперь при использовании драйвера диска my\_files все наши сохранения будут производится в папку public.

Так как загружать изображения нам надо во время создания нового раздела или изменения существующего, а для этих действий мы используем метод save() контроллера MainController, то изменения необходимо внести в этот метод.

Вернем вызов фасада DB обратно в наш контроллер:

<?php

namespace App\Http\Controllers;

use App\Content;

use App\Http\Requests\ContentRequest;

**use Illuminate\Support\Facades\DB;**

Перепишем метод save() контроллера MainController и приведем его к следующему виду:

public function save(ContentRequest $request) {

if ($request->file("photo")) {

$time\_rep = time();

$exch = $request->file('photo')->getClientOriginalExtension();

$name = $time\_rep . '\_mary-dance\_ru' . '.' . $exch;

$path = $request->file('photo')->storeAs('img/main', $name, 'my\_files');

if ($request->has("id")) {

$cat = DB::table('contents')->where('id', $request->id)->update([

'title' => $request->title,

'pageContent' => $request->pageContent,

'photo' => $path

]);

$s = " изменен";

} else {

$cat = DB::table('contents')->insert([

[

'title' => $request->title,

'pageContent' => $request->pageContent,

'photo' => $path

]

]);

$s = " добавлен";

}

} elseif ($request->has("id")) {

$cat = Content::findOrFail($request->id);

$cat->fill($request->all())->save();

$s = " изменен";

} else {

$cat = Content::create($request->all());

$s = " добавлен";

}

return redirect()->action("MainController@all")->with("status", "Раздел " . $request->title . $s);

}

У нас есть переменная $cat, в которую заносится количество добавленных или измененных данных. Мы можем ее использовать для проверки – были ли внесены изменения в БД: if $cat { }. Если по каким-либо причинам запись не была сделана, то в переменной будет 0, а PHP трактует 0 как false, поэтому если условие вернет false, то можно будет перенаправить действие метода.

Еще в методе у нас много логики, и мы можем сделать рефакторинг кода и перенести это все в другие классы. Предлагаю сделать это самостоятельно.

Еще надо сделать автоматическое удаление старых фотографий, при изменении изображения в записи или при удалении раздела. Представим, что наш проект будет существовать очень долго и периодически разделы и изображения будут изменяться. Через какое-то время в папке img\main соберется большое количество абсолютно ненужных изображений. Вот именно эту проблему мы сейчас и решим.

Создадим в моделе Content метод для извлечения старой записи об изображении для его удаления при редактировании записи

public static function delPhoto($id) {

return DB::table("contents")->where("id", $id)->value("photo");

}

Занесем в переменную $delPhoto данные из заменяемого поля photo, а сразу после внесения изменения в запись удалим файл в переменной с фото с помощью метода delete() фасада Storage (который надо добавить) через указание драйвера диска. Изменим методы save() и destroy() в контроллере MainController:

public function save(ContentRequest $request) {

if ($request->file("photo")) {

**$delPhoto = Content::delPhoto($request->id);**

$time\_rep = time();

$exch = $request->file('photo')->getClientOriginalExtension();

$name = $time\_rep . '\_mary-dance\_ru' . '.' . $exch;

$path = $request->file('photo')->storeAs('img/main', $name, 'my\_files');

if ($request->has("id")) {

$cat = DB::table('contents')->where('id', $request->id)->update([

'title' => $request->title,

'pageContent' => $request->pageContent,

'photo' => $path

]);

**Storage::disk('my\_files')->delete($delPhoto);**

$s = " изменен";

} else {

$cat = DB::table('contents')->insert([

[

'title' => $request->title,

'pageContent' => $request->pageContent,

'photo' => $path

]

]);

$s = " добавлен";

}

} elseif ($request->has("id")) {

$cat = Content::findOrFail($request->id);

$cat->fill($request->all())->save();

$s = " изменен";

} else {

$cat = Content::create($request->all());

$s = " добавлен";

}

return redirect()->action("MainController@all")->with("status", "Раздел " . $request->title . $s);

}

public function destroy(Content $content) {

**if (isset($content->photo)) {**

**$delPhoto = Content::delPhoto($content->id);**

}

Content::destroy($content->id);

**Storage::disk("my\_files")->delete($delPhoto);**

return redirect()->action("MainController@all")

->with("status", "Раздел " . $content->title . " удалён");

}

Повесить на кнопку Удалить действие на JavaScript уточнение удаления.

Внесем изменения в файл contents\all.blade.php изменим ссылку на «кнопку» Удалить:

<a href="{{ action('MainController@destroy', ['id' => $conts->id]) }}" class="btn btn-success" **onclick="return confirm('Подтверждаете удаление?')**">

Удалить

</a>

Теперь при нажатии на кнопку Удалить, вначале появится окно с запросом на подтверждение удаления раздела. Если нажать Ок, раздел удалится, если же на кнопку Отмена, то окно сообщения просто закроется, удаления раздела не произойдет.