Министерство науки и высшего образования Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

По дисциплине: «Веб-технологии»

Лабораторная работа № 12

«Исследование возможностей фреймворка Laravel для разработки веб-приложений»

Выполнила

ст. гр. ИС/б-17-2-о

Кайда Е.Н.

Проверил:

Овчинников А.Л.

Севастополь

2020

Цель работы

Ознакомиться с возможностями фреймворка Laravel и приобрести практические навыки реализации веб-приложений с его помощью.

Постановка задачи

В ходе выполнения лабораторной работы необходимо с использованием фреймворка Laravel реализовать «Редактор Блога» и «Загрузка сообщений блога» в разделе администратора персонального сайта и «Мой Блог» в пользовательском разделе.

Предусмотреть возможность редактирования и удаления записей блога, реализовав валидацию средствами Laravel. Для этого разработать: миграцию, Eloquent модель, маршруты роутинга, контроллеры, валидации, Blade-макеты пользовательской и административной части сайта (дизайн «Блога» должен остаться прежним).

ход работы

Был создан шаблон представления app.blade.php, который содержит в себе меню сайта и при помощи @yield включает в себя представления определённых страниц:

<!DOCTYPE html>

<html>

<head>

    <meta name="csrf-token" content="{{ csrf\_token() }}">

    <title>@yield('title-block')</title>

    <link rel="stylesheet" type="text/css" href="/css/style.css">

    <link rel="stylesheet" href="/css/foto.css">

    <link rel="stylesheet" href="/css/menu.css">

    <link rel="stylesheet" href="/css/study.css">

    <script src="/scripts/jquery-2.2.3.js"></script>

    <script src="/scripts/JQModalWindow.js"></script>

    @yield('scripts')

</head>

<body>

<input type="checkbox" id="nav-toggle" hidden>

    <nav class="nav">

        <label for="nav-toggle" class="nav-toggle" onclick></label>

        <h2 class="logo">

            <a>Little\_coon\_</a>

        </h2>

        <?php

            if(isset($\_SESSION['isAdmin']) && $\_SESSION['isAdmin']):

                echo "<h2>Admin</h2>";

                echo "<h3><a href='/exit'>Выход</a></h3>";

        ?>

        <ul>

            <!-- <li><a href="/admin/guest/upload">Загрузка отзывов</a> -->

            <li><a href="{{ route('blogUpload') }}">Загрузка сообщений блога</a>

            <!-- <li><a href="/admin/statistic/visitings">Статистика посещаемости</a> -->

            <?php

                else:

                    echo "<ul>";

                    if(isset($\_SESSION['isAdmin']) && !$\_SESSION['isAdmin']){

                        echo "<h2>".$\_SESSION['fio']."</h2>";

                        echo "<h3><a href='/exit'>Выход</a></h3>";

                    }else{

                        echo "<h3><a href='/login'>Вход</a></h3>";

                    }

                endif;

            ?>

            <li><a href="{{ route('index') }}">Главная</a>

            <li><a href="{{ route('blog') }}">Мой Блог</a>

            <li><a href="{{ route('blogEditor') }}">Редактор Блога</a>

        </ul>

    </nav>

    @yield('content')

</body>

</html>

А представление конкретной страницы выглядит теперь так (например, страница редактора блога blogEditor.blade.php):

@extends ('layouts.app')

@section('title-block')

  Редактор блога

@endsection

@section('content')

<section>

    <h1>Добавление записи Блога</h1>

    @include('common.errors')

    @include('common.msg')

    <form enctype="multipart/form-data"  action="{{ route('saveRecord') }}" method=POST >

    @csrf

        <p><input type="text" id="topic" name="topic" placeholder="Тема сообщения" data-toggle="popover"

            >

        </p>

        <p><textarea id="message" name="message" data-toggle="popover" placeholder="Ваше сообщение..."

            ></textarea>

        </p>

        <p>

            <input type="hidden" name="30000" value="30000" />

            <input name="userFile" type="file" accept="image/\*">

        </p>

        <input type="submit"  id="submit" value="Отправить" >

        <input type="reset" id="reset" value="Очистить" >

    </form>

</section>

@endsection

Для обращения к методам контроллеров были созданы следующие пути в файле web.php:

Route::get('/', function () {

    return view('index');

})->name('index');

Route::get('/blog', 'BlogController@blog')->name('blog');

Route::post('/blog/updateRecord', 'BlogController@updateRecord');

Route::get('/blogEditor', function () {

    return view('blogEditor');

})->name('blogEditor');

Route::post('/blog/saveRecord', 'BlogController@saveRecord')->name('saveRecord');

Route::get('/addComment', 'CommentController@addComment');

Route::get('/admin/blog/upload', 'BlogController@upload')->name('blogUpload');

Route::post('/admin/blog/upload', 'BlogController@loadBlogs')->name('loadBlogs');

Route::get('/login', 'UserController@auth')->name('auth');

Route::post('/login-login', 'UserController@login')->name('login');

Route::get('/exit', 'Controller@exit')->name('exit');

Route::delete('/blog/delete/{id}', function ($id) {

    Blog::findOrFail($id)->delete();

    return redirect('/blog');

})->name('deleteBlog');

Например при вызове url = '/blog/updateRecord' будет вызван метод updateRecord() контроллера BlogController.

Благодаря такой маршрутизации реализованы возможности:

* обновления записи,
* удаления:

blog.js

function deleteRecord(id){

    $("#form-delete").attr("action", "/blog/delete/"+id)

    $("#del-btn").click();

}

Web.php

Route::delete('/blog/delete/{id}', function ($id) {

    Blog::findOrFail($id)->delete();

    return redirect('/blog');

})->name('deleteBlog');

* сохранение новой записи в бд:

BlogRequest.php

public function rules()

    {

        return [

            'topic' => 'required|max:50',

            'message' => 'required',

        ];

    }

    public function messages()

    {

        return [

            'topic.required' => 'Тема сообщения не должна быть пустой',

            'topic.max' => 'Тема сообщения должна содержать не более 50 символов',

            'message.required' => 'Сообщение не должно быть пустым',

        ];

    }

BlogController.php

public function saveRecord(BlogRequest $req){

        $model = new Blog();

        $model->topic =  trim($\_POST['topic']);

        $model->message = trim($\_POST['message']);

        if ($req->hasFile('userFile')){

            $file = $req->file('userFile');

            $file->move(public\_path()."/img/", $req->file('userFile')->getClientOriginalName());

            $model->img\_name = $req->file('userFile')->getClientOriginalName();

            $model->img\_src = public\_path().'/img';

        }

        $status = $model->save();

        echo $status;

        return redirect()->route('blogEditor')->with('status', ($status)? "Добавление прошло успешно":"Ошибка сохранения");

    }

* добавление комментария:

public function addComment(){

        header ('Content-Type: text/javascript');

        $status = false;

        $respons = "";

        $row = [];

        if (!empty($\_REQUEST['comment'])){

            $commentModel = new Comment();

            $commentModel->id\_blog = $\_REQUEST['blog'];

            $commentModel->id\_user = $\_SESSION['id\_user'];

            $commentModel->message = $\_REQUEST['comment'];

            $status = $commentModel->save();

            $row = [null, $\_REQUEST['blog'], $\_SESSION['id\_user'], $\_REQUEST['comment'], date('Y-m-d H:i:s')];

            echo "var data = ".(json\_encode($row)).";\n";

            $respons = ($status) ? "'Комментарий сохранен'" : "'Ошибка сохранения комментария'";

       }else{

            $status = 0;

            $respons = "'Введите комментарий'";

       }

        echo "var status = ".$status.";\n";

        echo "var respons = ".$respons.";\n";

        echo "responsAddComment();\n";

       exit;

    }

* Считывание записей блога из файла и запись в бд:

public function loadBlogs(Request $request){

if ($request->hasFile('userFile')){

        $file = $request->file('userFile');

        $blogs = [];

        $fileR = fopen($file, "r");

        for($i=0; !feof($fileR); $i++){

            $str = fgets($fileR);

            if (empty($str)) continue;

            $blogs[$i] = explode(';', $str);

        }

        fclose($fileR);

        $counter = 0;

        foreach($blogs as $blog){

            $temp = $blog;

            $newBlog = new Blog();

            if(strlen($temp[0])>0){

                $newBlog->topic = $temp[0];

                $newBlog->message = $temp[1];

                $newBlog->img\_name = trim($temp[2]);

                $newBlog->img\_src = trim($temp[3]);

                $newBlog->created\_at = trim($temp[4]);

                $newBlog->save();

                $counter++;

            }

        }

        return redirect()->back()->with('status', "Успешно добавлено $counter записей блога");

    }

    return redirect()->back()->with('status', "Ошибка считывания из файла");

}

Подробный код всех представлен на сайте GitHub (Приложение А).

Выводы

В ходе выполнения лабораторной работы были изучены возможности фреймворка Laravel и приобрестены практические навыки реализации веб-приложений с его помощью.

.

Приложение А

1. Код сайта - https://github.com/kaidalena/web\_lr\_semestr\_6/laba5laravel