Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

Лабораторная работа 5

по дисциплине «Технологии разработки WEB-приложений»

«Работа с механизмами авторизации в технологии ASP.NET MVC»

|  |  |  |
| --- | --- | --- |
| Студент |  | Ковальчук А. Д. |
| Группа | АИ-14 |  |
| Руководитель |  | Лаврухина Т. В. |

Липецк – 2018 г.

Цель работы

На практике познакомиться с механизмами авторизации Identity технологии ASP.NET MVC.

Задание

Реализовать авторизацию на основе ASP.NET Identity. Каждый пользователь должен иметь свой профиль с его личной «корзиной»/«избранным».

# Результаты выполнения работы

## Описание предметной области

В качестве предметной области выбран Web-сервис для ведения и блогов и чтения статей.

## Коды страниц

### Коды контроллеров

AccountController

using System;

using System.Globalization;

using System.Linq;

using System.Security.Claims;

using System.Threading.Tasks;

using System.Web;

using System.Web.Mvc;

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.Owin;

using Microsoft.Owin.Security;

using Lab1.Models;

namespace Lab1.Controllers

{

[Authorize]

public class AccountController : Controller

{

ApplicationSignInManager \_signInManager;

ApplicationUserManager \_userManager;

public AccountController()

{

}

public AccountController(ApplicationUserManager userManager, ApplicationSignInManager signInManager)

{

UserManager = userManager;

SignInManager = signInManager;

}

public ApplicationSignInManager SignInManager

{

get

{

return \_signInManager ?? HttpContext.GetOwinContext().Get<ApplicationSignInManager>();

}

set

{

\_signInManager = value;

}

}

public ApplicationUserManager UserManager

{

get

{

return \_userManager ?? HttpContext.GetOwinContext().GetUserManager<ApplicationUserManager>();

}

set

{

\_userManager = value;

}

}

//

// GET: /Account/Login

[AllowAnonymous]

public ActionResult Login(string returnUrl)

{

ViewBag.ReturnUrl = returnUrl;

return View();

}

//

// POST: /Account/Login

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)

{

if (!ModelState.IsValid)

return View(model);

// Сбои при входе не приводят к блокированию учетной записи

// Чтобы ошибки при вводе пароля инициировали блокирование учетной записи, замените на shouldLockout: true

var result = await SignInManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe, shouldLockout: false);

switch (result)

{

case SignInStatus.Success:

return RedirectToLocal(returnUrl);

case SignInStatus.LockedOut:

return View("Lockout");

case SignInStatus.RequiresVerification:

return RedirectToAction("SendCode", new { ReturnUrl = returnUrl, model.RememberMe });

case SignInStatus.Failure:

default:

ModelState.AddModelError("", "Неудачная попытка входа.");

return View(model);

}

}

//

// GET: /Account/Register

[AllowAnonymous]

public ActionResult Register()

{

return View();

}

//

// POST: /Account/Register

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Register(RegisterViewModel model)

{

if (ModelState.IsValid)

{

var user = new ApplicationUser {

LastName = model.LastName,

FirstName = model.FirstName,

MiddleName = model.MiddleName,

UserName = model.Email,

BirthDay = model.BirthDay,

Email = model.Email,

Sex = model.Sex,

PhoneNumber = model.PhoneNumber

};

var result = await UserManager.CreateAsync(user, model.Password);

if (result.Succeeded)

{

await UserManager.AddToRoleAsync(user.Id, "user");

await UserManager.SetLockoutEnabledAsync(user.Id, true);

await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);

// Дополнительные сведения о том, как включить подтверждение учетной записи и сброс пароля, см. по адресу: http://go.microsoft.com/fwlink/?LinkID=320771

// Отправка сообщения электронной почты с этой ссылкой

// string code = await UserManager.GenerateEmailConfirmationTokenAsync(user.Id);

// var callbackUrl = Url.Action("ConfirmEmail", "Account", new { userId = user.Id, code = code }, protocol: Request.Url.Scheme);

// await UserManager.SendEmailAsync(user.Id, "Подтверждение учетной записи", "Подтвердите вашу учетную запись, щелкнув <a href=\"" + callbackUrl + "\">здесь</a>");

return RedirectToAction("Index", "Home");

}

AddErrors(result);

}

// Появление этого сообщения означает наличие ошибки; повторное отображение формы

return View(model);

}

//

// POST: /Account/LogOff

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult LogOff()

{

AuthenticationManager.SignOut(DefaultAuthenticationTypes.ApplicationCookie);

return RedirectToAction("Index", "Home");

}

#region Вспомогательные приложения

// Используется для защиты от XSRF-атак при добавлении внешних имен входа

const string XsrfKey = "XsrfId";

IAuthenticationManager AuthenticationManager

{

get

{

return HttpContext.GetOwinContext().Authentication;

}

}

void AddErrors(IdentityResult result)

{

foreach (var error in result.Errors)

ModelState.AddModelError("", error);

}

ActionResult RedirectToLocal(string returnUrl)

{

if (Url.IsLocalUrl(returnUrl))

return Redirect(returnUrl);

return RedirectToAction("Index", "Home");

}

#endregion

}

}

ManageController

using System;

using System.Linq;

using System.Threading.Tasks;

using System.Web;

using System.Web.Mvc;

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.Owin;

using Microsoft.Owin.Security;

using Lab1.Models;

using Microsoft.AspNet.Identity.EntityFramework;

namespace Lab1.Controllers

{

[Authorize]

public class ManageController : Controller

{

ApplicationSignInManager \_signInManager;

ApplicationUserManager \_userManager;

public ManageController()

{

}

public ManageController(ApplicationUserManager userManager, ApplicationSignInManager signInManager)

{

UserManager = userManager;

SignInManager = signInManager;

}

public ApplicationSignInManager SignInManager

{

get

{

return \_signInManager ?? HttpContext.GetOwinContext().Get<ApplicationSignInManager>();

}

set

{

\_signInManager = value;

}

}

public ApplicationUserManager UserManager

{

get

{

return \_userManager ?? HttpContext.GetOwinContext().GetUserManager<ApplicationUserManager>();

}

set

{

\_userManager = value;

}

}

//

// GET: /Manage/Index

[Authorize]

public async Task<ActionResult> Index(ManageMessageId? message)

{

ViewBag.StatusMessage =

message == ManageMessageId.ChangePasswordSuccess ? "Ваш пароль изменен."

: message == ManageMessageId.SetPasswordSuccess ? "Пароль задан."

: message == ManageMessageId.SetTwoFactorSuccess ? "Настроен поставщик двухфакторной проверки подлинности."

: message == ManageMessageId.Error ? "Произошла ошибка."

: message == ManageMessageId.AddPhoneSuccess ? "Ваш номер телефона добавлен."

: message == ManageMessageId.RemovePhoneSuccess ? "Ваш номер телефона удален."

: "";

var userId = User.Identity.GetUserId();

using (ApplicationDbContext db = new ApplicationDbContext())

{

ViewBag.CurrentUser = db.Users.Find(User.Identity.GetUserId());

ViewBag.RolesCurrentUser = UserManager.GetRoles(User.Identity.GetUserId());

}

var model = new IndexViewModel

{

HasPassword = HasPassword(),

PhoneNumber = await UserManager.GetPhoneNumberAsync(userId),

TwoFactor = await UserManager.GetTwoFactorEnabledAsync(userId),

Logins = await UserManager.GetLoginsAsync(userId),

BrowserRemembered = await AuthenticationManager.TwoFactorBrowserRememberedAsync(userId)

};

return View(model);

}

//

// GET: /Manage/ChangePassword

public ActionResult ChangePassword()

{

return View();

}

//

// POST: /Manage/ChangePassword

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> ChangePassword(ChangePasswordViewModel model)

{

if (!ModelState.IsValid)

return View(model);

var result = await UserManager.ChangePasswordAsync(User.Identity.GetUserId(), model.OldPassword, model.NewPassword);

if (result.Succeeded)

{

var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());

if (user != null)

await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);

return RedirectToAction("Index", new { Message = ManageMessageId.ChangePasswordSuccess });

}

AddErrors(result);

return View(model);

}

//

// GET: /Manage/Administration

[Authorize(Roles = "admin")]

public ActionResult Administration()

{

using (ApplicationDbContext db = new ApplicationDbContext())

{

ViewBag.Roles = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(db)).Roles.ToList();

}

var allUsers = UserManager.Users.ToList();

return View(allUsers);

}

//

// POST: /Manage/Lock

[Authorize(Roles = "admin")]

public async Task<ActionResult> Lock(string UserID)

{

ApplicationUser user = await UserManager.FindByIdAsync(UserID);

if (user!=null)

{

user.LockoutEndDateUtc = DateTime.Now.AddDays(100);

await UserManager.UpdateAsync(user);

}

return new RedirectResult(Url.Action("Administration", "Manage", new { }));

}

//

// POST: /Manage/Unlock

[Authorize(Roles = "admin")]

public async Task<ActionResult> Unlock(string UserID)

{

ApplicationUser user = await UserManager.FindByIdAsync(UserID);

if (user != null)

{

user.LockoutEndDateUtc = null;

await UserManager.UpdateAsync(user);

}

return new RedirectResult(Url.Action("Administration", "Manage", new { }));

}

protected override void Dispose(bool disposing)

{

if (disposing && \_userManager != null)

{

\_userManager.Dispose();

\_userManager = null;

}

base.Dispose(disposing);

}

#region Вспомогательные приложения

// Используется для защиты от XSRF-атак при добавлении внешних имен входа

IAuthenticationManager AuthenticationManager

{

get

{

return HttpContext.GetOwinContext().Authentication;

}

}

void AddErrors(IdentityResult result)

{

foreach (var error in result.Errors)

ModelState.AddModelError("", error);

}

bool HasPassword()

{

var user = UserManager.FindById(User.Identity.GetUserId());

if (user != null)

return user.PasswordHash != null;

return false;

}

public enum ManageMessageId

{

AddPhoneSuccess,

ChangePasswordSuccess,

SetTwoFactorSuccess,

SetPasswordSuccess,

RemoveLoginSuccess,

RemovePhoneSuccess,

Error

}

#endregion

}

}

Методы контроллера BlogsController, которые отвечают за вкладку «Избранное»

[Authorize]

public ActionResult Favorites(int? page, MessageStatus? message)

{

try

{

string CurrentUserID = User.Identity.GetUserId();

if (CurrentUserID == null) return RedirectToAction("Index", "Blogs", new { MessageStatus.UserForFavoritesNotFound });

//Определитель сообщения

ViewBag.Message =

message == MessageStatus.ArticleDeletedFavorites ? "Статья успешно удалена из вашего избранного!"

: message == MessageStatus.DeletedArticleNotFoundFavorites ? "У вас в Избранном нет статьи, которую вы пытаетесь удалить."

: message == MessageStatus.ArticleDeletedFromFaavoritesNotFound ? "Не найдена статья для удаления ее из Избранного."

: message == MessageStatus.ArticleDeleted ? "Статья удалена!"

: message == MessageStatus.DeletedArticleNotFound ? "Статья, которую вы пытаетесь удалить, не существует."

: message == MessageStatus.EditedArticleNotFound ? "Статья, которую вы пытались отредактировать, не существует."

: "";

//Определитель цвета оповещения: 1 - красный, 0 - синий

ViewBag.MessageCode =

message == MessageStatus.DeletedArticleNotFoundFavorites ? 1

: message == MessageStatus.ArticleDeletedFromFaavoritesNotFound ? 1

: message == MessageStatus.DeletedArticleNotFound ? 1

: message == MessageStatus.EditedArticleNotFound ? 1

: 0;

ViewBag.Categorys = db.Categorys;

ViewBag.Articles = db.Articles;

ViewBag.Authors = db.Users;

List<Favorites> favorites = db.Favorites.Where(x => x.UserID == CurrentUserID).ToList();

favorites.Reverse();

ViewBag.CountPage = (int)Math.Ceiling((double)favorites.Count / CAFP);

ViewBag.FirstPageDiasbled = "";

ViewBag.LastPageDiasbled = "";

// Если статей меньше CAFP в Избранном, то выводим все

if (page == null)

{

// Если больше CAFP, но пользователь не указывал какую страницу открыть, то

// открываем превую

if (ViewBag.CountPage > 1)

{

favorites = favorites.GetRange(0, CAFP);

ViewBag.CurrentPage = 1; //В этом случае текущей страницей будет первая

ViewBag.FirstPageDiasbled = "disabled"; //Выключаем возврат на предыдующую страницу

}

}

else

{

// Если статей больше CAFP в Избранном, то выводим CAFP статей для указанной страницы

//Если на последней странице нет CAFP статей

if (CAFP \* (int)page >= favorites.Count)

{

favorites = favorites.GetRange(CAFP \* ((int)page - 1), favorites.Count() - CAFP \* ((int)page - 1));

ViewBag.LastPageDiasbled = "disabled"; //Выключаем переход на следующую страницу

}

else

{

//Если на странице есть CAFP статей, то берем их

favorites = favorites.GetRange(CAFP \* ((int)page - 1), CAFP);

}

ViewBag.CurrentPage = page; //Получаем тукущую страницу

if (page == 1) ViewBag.FirstPageDiasbled = "disabled";

}

return View(favorites);

}

catch (Exception ex)

{

logger.Debug(DateTime.Now.ToString("dd.MM.yyyy HH:mm:ss") + " in /Blogs/Favorites : " + ex.Message);

return RedirectToAction("Error", "Shared");

}

}

//Метод добавления статьи в избранное

[Authorize]

public ActionResult AddFavorites(int? ArticleID, int? page)

{

try

{

MessageStatus message = MessageStatus.AddFavoritesArticleOrUserNotFound;

string CurrentUserID = User.Identity.GetUserId();

if (db.Users.Find(CurrentUserID) != null && db.Articles.Find(ArticleID) != null)

{

message = MessageStatus.ArticleAlreadyFavorites;

if (db.Favorites.Where(x => x.UserID == CurrentUserID && x.ArticleID == ArticleID).ToList().Count() == 0)

{

db.Favorites.Add(new Favorites { UserID = CurrentUserID, ArticleID = (int)ArticleID });

db.SaveChanges();

message = MessageStatus.ArticleAddFaavorites;

}

return new RedirectResult(Url.Action("Index", "Blogs", new { db.Articles.Find(ArticleID).CategoryID, page, message }) + "#article" + ArticleID);

}

else

{

return RedirectToAction("Index", "Blogs", new { message });

}

}

catch (Exception ex)

{

logger.Debug(DateTime.Now.ToString("dd.MM.yyyy HH:mm:ss") + " in /Blogs/AddFavorites : " + ex.Message);

return RedirectToAction("Error", "Shared");

}

}

//Метод удаления статьи из избранного

[Authorize]

public ActionResult DelFavorites(int? ArticleID, int? page, int? NextArticleID)

{

try

{

string CurrentUserID = User.Identity.GetUserId();

MessageStatus message;

if (db.Users.Find(CurrentUserID) != null && db.Articles.Find(ArticleID) != null)

{

message = MessageStatus.DeletedArticleNotFoundFavorites;

if (db.Favorites.Where(x => x.UserID == CurrentUserID && x.ArticleID == ArticleID).ToList().Count > 0)

{

Favorites fav = db.Favorites.Where(x => x.UserID == CurrentUserID && x.ArticleID == ArticleID).ToList()[0];

db.Favorites.Remove(fav);

db.SaveChanges();

message = MessageStatus.ArticleDeletedFavorites;

}

return new RedirectResult(Url.Action("Favorites", "Blogs", new { page, message }) + "#article" + NextArticleID);

}

else

{

message = MessageStatus.ArticleDeletedFromFaavoritesNotFound;

if (db.Users.Find(CurrentUserID) != null)

{

return new RedirectResult(Url.Action("Favorites", "Blogs", new { page, message }) + "#article" + NextArticleID);

}

else

{

message = MessageStatus.DeleteFavoritesArticleAndUserNotFound;

return new RedirectResult(Url.Action("Index", "Blogs", new { page, message }) + "#article" + NextArticleID);

}

}

}

catch (Exception ex)

{

logger.Debug(DateTime.Now.ToString("dd.MM.yyyy HH:mm:ss") + " in /Blogs/DelFavorites : " + ex.Message);

return RedirectToAction("Error", "Shared");

}

}

### Коды некоторых файлов представлений

Account/Login

@using Lab1.Models

@model LoginViewModel

@{

ViewBag.Title = "Авторизация";

ViewBag.Page4 = "nav-item active";

}

<h2 class="display-4 mb-3">Авторизация</h2>

@if (!Request.IsAuthenticated)

{

using (Html.BeginForm("Login", "Account", new { ReturnUrl = ViewBag.ReturnUrl }, FormMethod.Post, new { @class = "form-horizontal", role = "form" }))

{

@Html.AntiForgeryToken()

<div class="container alert alert-success" role="alert">

<h4>Используйте локальную учетную запись для входа</h4>

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="row mb-2 align-items-end">

<div class="col-lg-3 text-lg-right">

@Html.LabelFor(m => m.Email)

</div>

<div class="col-lg-9">

@Html.TextBoxFor(m => m.Email, new { @class = "form-control" })

@Html.ValidationMessageFor(m => m.Email, "", new { @class = "text-danger" })

</div>

</div>

<div class="row mb-2 align-items-end">

<div class="col-lg-3 text-lg-right">

@Html.LabelFor(m => m.Password)

</div>

<div class="col-lg-9">

@Html.PasswordFor(m => m.Password, new { @class = "form-control" })

@Html.ValidationMessageFor(m => m.Password, "", new { @class = "text-danger" })

</div>

</div>

<div class="row mb-2">

<div class="offset-lg-3 col-lg-9">

<div class="checkbox">

@Html.CheckBoxFor(m => m.RememberMe)

@Html.LabelFor(m => m.RememberMe)

</div>

</div>

</div>

<div class="row mb-2">

<div class="offset-lg-3 col-lg-9 form-inline">

<input type="submit" class="btn btn-primary mr-1 mb-1" value="Вход">

<input type="button" class="btn btn-secondary mb-1" href="/Account/Register" value="Регистрация">

</div>

</div>

</div>

}

}

else

{

<div class="container alert alert-danger" role="alert">

Вы уже зарегистрированы и авторизованы.

</div>

}

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

}

Account/Register

@model Lab1.Models.RegisterViewModel

@{

ViewBag.Page3 = "nav-item active";

}

<h2 class="display-4 mb-3">Регистрация</h2>

@if (!Request.IsAuthenticated)

{

using (Html.BeginForm("Register", "Account", FormMethod.Post, new { @class = "form-horizontal", role = "form" }))

{

@Html.AntiForgeryToken()

<div class="container alert alert-warning" role="alert">

@Html.ValidationSummary("", new { @class = "text-danger mb-2" })

<div class="row mb-2 align-items-end">

<div class="col-md-3 text-md-right">

@Html.LabelFor(m => m.LastName)<label class="text-danger">\*</label>

</div>

<div class="col-md-9">

@Html.TextBoxFor(m => m.LastName, new { @class = "form-control", @placeholder = "Введите фамилию" })

</div>

</div>

<div class="row mb-2 align-items-end">

<div class="col-md-3 text-md-right">

@Html.LabelFor(m => m.FirstName)<label class="text-danger">\*</label>

</div>

<div class="col-md-9">

@Html.TextBoxFor(m => m.FirstName, new { @class = "form-control", @placeholder = "Введите имя" })

</div>

</div>

<div class="row mb-2 align-items-end">

<div class="col-md-3 text-md-right">

@Html.LabelFor(m => m.MiddleName)

</div>

<div class="col-md-9">

@Html.TextBoxFor(m => m.MiddleName, new { @class = "form-control", @placeholder = "Введите отчество" })

</div>

</div>

<div class="row align-items-end">

<div class="col-md-3 text-md-right">

@Html.LabelFor(m => m.BirthDay)<label class="text-danger">\*</label>

</div>

<div class="col-md-9">

@Html.EditorFor(m => m.BirthDay, new { htmlAttributes = new { @class = "form-control" } })

</div>

</div>

<div class="row mt-1 mb-2 align-items-end">

<div class="col-md-3 text-md-right">

@Html.LabelFor(m => m.Email)<label class="text-danger">\*</label>

</div>

<div class="col-md-9">

@Html.TextBoxFor(m => m.Email, new { @class = "form-control", @placeholder = "Введите Email" })

</div>

</div>

<div class="row mb-2 align-items-end">

<div class="col-md-3 text-md-right">

@Html.LabelFor(m => m.Password)<label class="text-danger">\*</label>

</div>

<div class="col-md-9">

@Html.PasswordFor(m => m.Password, new { @class = "form-control", @placeholder = "Введите пароль" })

</div>

</div>

<div class="row mb-2 align-items-end">

<div class="col-md-3 text-md-right">

@Html.LabelFor(m => m.ConfirmPassword)<label class="text-danger">\*</label>

</div>

<div class="col-md-9">

@Html.PasswordFor(m => m.ConfirmPassword, new { @class = "form-control", @placeholder = "Повторите пароль" })

</div>

</div>

<div class="row mb-2 align-items-end">

<div class="col-md-3 text-md-right">

@Html.LabelFor(m => m.PhoneNumber)

</div>

<div class="col-md-9">

@Html.TextBoxFor(m => m.PhoneNumber, new { @class = "form-control", @placeholder = "Введите номер телефона" })

</div>

</div>

<div class="row mb-2 align-items-end">

<div class="col-md-3 text-md-right">

@Html.LabelFor(m => m.Sex)<label class="text-danger">\*</label>

</div>

<div class="col-sm-9">

<label class="radio-inline">

@Html.RadioButtonFor(m => m.Sex, "Мужской") Муж

</label>

<label class="radio-inline ml-sm-2">

@Html.RadioButtonFor(m => m.Sex, "Женский") Жен

</label>

</div>

</div>

<div class="offset-md-3 col-md-9"><label class="text-danger">\*</label> - обязательные для заполнения поля</div>

<div class="row mb-2">

<div class="offset-md-3 col-md-9">

<input type="submit" class="btn btn-primary" value="Регистрация">

</div>

</div>

</div>

}

}

else

{

<div class="container alert alert-danger" role="alert">

Вы уже зарегистрированы и авторизованы.

</div>

}

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

}

Blogs/Favorites

@using Lab1.Models;

@using System.Linq;

@using System.Data.Entity;

@using Microsoft.AspNet.Identity

@model List<Favorites>

@{

ViewBag.Page3 = "nav-item active";

ViewBag.Hidden = "hidden = \"hidden\"";

DbSet<Article> articles = ViewBag.Articles;

DbSet<Category> categorys = ViewBag.Categorys;

DbSet<ApplicationUser> authors = ViewBag.Authors;

}

@functions {

public string isActive(bool Condition)

{

if (Condition)

{

return "active";

}

else

{

return "";

}

}

public string isHidden(bool Condition)

{

if (Condition)

{

return "hidden = \"hidden\"";

}

else

{

return "";

}

}

public string colorMess(int? code)

{

if (code == 1)

{

return "danger";

}

return "primary";

}

public int getNextArtID(List<Favorites> favorites, int ArticleID)

{

if (favorites.Count() > 1)

{

for (int i = 0; i < favorites.Count(); i++)

{

if (favorites[i].ArticleID == ArticleID)

{

if (i + 1 != favorites.Count())

{

return favorites[i + 1].ArticleID;

}

if (i - 1 != -1)

{

return favorites[i - 1].ArticleID;

}

}

}

}

return 0;

}

}

<h2 class="display-4 mb-3">Избранное</h2>

@\*Оповещения\*@

<div class="alert alert-@colorMess(ViewBag.MessageCode) col-12" @isHidden(ViewBag.Message == "") role="alert">

@ViewBag.Message

</div>

@\*Выводим избранные статьи\*@

@foreach (Favorites fav in Model)

{

Article art = articles.Where(x => x.ID == fav.ArticleID).ToList()[0];

ApplicationUser author = authors.Where(x => x.Id == art.AuthorID).ToList()[0];

Category cat = categorys.Where(x => x.ID == art.CategoryID).ToList()[0];

<div class="row" id="@("article" + art.ID)">

<div class="alert alert-secondary col-12" role="alert">

@\*Тема и набор кнопок\*@

<div class="row align-items-center">

<div class="col-lg-9 text-center">

<h3>@art.Subject <span class="badge badge-primary">@cat.Name</span></h3>

</div>

<div class="col-lg-3 text-center text-lg-right">

@\*Скрываем кнопки, если это не текущий пользователь\*@

@if (Request.IsAuthenticated)

{

<a class="btn btn-danger mb-1 btn-sm" href="/Blogs/Delete/?Favorites=true&ArticleID=@art.ID&page=@ViewBag.CurrentPage&NextArticleID=@getNextArtID(Model, art.ID)" @isHidden(art.AuthorID != User.Identity.GetUserId())>Удалить</a>

<a class="btn btn-warning mb-1 btn-sm" href="/Blogs/Edit/?Favorites=true&ArticleID=@art.ID" @isHidden(art.AuthorID != User.Identity.GetUserId())>Редактировать</a>

}

<button type="button" class="btn btn-secondary btn-sm mb-1" data-toggle="collapse"

href="#@("articleText" + art.ID)" aria-expanded="false"

aria-controls="@("articleText" + art.ID)">

Развернуть/Свернуть

</button>

</div>

</div>

<blockquote class="blockquote">

<hr />

@\*Сворачиваемое содержимое с текстом статьи\*@

<div class="collapse multi-collapse" id="@("articleText" + art.ID)">

<p>@Html.Raw(Html.Encode(art.Text).Replace("\r\n", "</p><p>"))</p>

<hr />

</div>

@\*Кнопка добавления в избранное, подпись автора, дата и время\*@

<div class="row align-items-center">

<div class="col-sm-4 text-right text-sm-left">

<a class="btn btn-info btn-sm" href="/Blogs/DelFavorites/?ArticleID=@art.ID&page=@ViewBag.CurrentPage&NextArticleID=@getNextArtID(Model, art.ID)">Удалить из избранного</a>

</div>

<div class="col-sm-8">

<footer class="blockquote-footer text-right">@author.LastName @author.FirstName - @art.DT.ToString("dd.MM.yyyy HH:mm") / <cite title="MyBlog">MyBlog</cite></footer>

</div>

</div>

</blockquote>

</div>

</div>

}

@\*Выводим преключатели между страницами, если статей больше 10\*@

@if (ViewBag.CountPage > 1)

{

<nav aria-label="Page navigation example">

<ul class="pagination justify-content-center">

@\*Ссылка перехода на предыдущую страницу\*@

<li class="page-item @ViewBag.FirstPageDiasbled">

<a class="page-link"

href="/Blogs/Favorites/?page=@(ViewBag.CurrentPage-1)">Назад</a>

</li>

@\*Если страниц не больше 5, то выводим ссылки на все\*@

@if (ViewBag.CountPage <= 5)

{

for (int i = 0; i < ViewBag.CountPage; i++)

{

<li class="page-item @isActive(ViewBag.CurrentPage==i+1)"><a class="page-link" href="/Blogs/Favorites/?page=@(i+1)">@(i + 1)</a></li>

}

}

else

{

// Если страниц больше 5, то выводим ссылки только 5 ближайших

// Если текущая страница имеет слева и справа хотя бы по 2, то

// Выводим слева и справа от текущей по 2 страницы

if (ViewBag.CurrentPage > 2 && ViewBag.CurrentPage < ViewBag.CountPage - 1)

{

for (int i = ViewBag.CurrentPage - 3; i < ViewBag.CurrentPage + 2; i++)

{

<li class="page-item @isActive(ViewBag.CurrentPage == i + 1)"><a class="page-link" href="/Blogs/Favorites/?page=@(i + 1)">@(i + 1)</a></li>

}

}

else

{

// Если страниц меню слева от текущей 1 или меньше,

// то выводим первые 5 ссылок на страницы

if (ViewBag.CurrentPage <= 2)

{

for (int i = 0; i < 5; i++)

{

<li class="page-item @isActive(ViewBag.CurrentPage == i + 1)"><a class="page-link" href="/Blogs/Favorites/?page=@(i + 1)">@(i + 1)</a></li>

}

}

else

{

// Если страниц меню справа от текущей 1 или меньше,

// то выводим последние 5 ссылок на страницы

if (ViewBag.CurrentPage >= ViewBag.CountPage - 1)

{

for (int i = ViewBag.CountPage - 5; i < ViewBag.CountPage; i++)

{

<li class="page-item @isActive(ViewBag.CurrentPage==i+1)"><a class="page-link" href="/Blogs/Favorites/?page=@(i+1)">@(i + 1)</a></li>

}

}

}

}

}

@\*Ссылка перехода на следующую страницу\*@

<li class="page-item @ViewBag.LastPageDiasbled">

<a class="page-link"

href="/Blogs/Favorites/?page=@(ViewBag.CurrentPage+1)">Далее</a>

</li>

</ul>

</nav>

}

### Модели

Также были созданы модели

Article

public class Article

{

public int ID { get; set; }

[Required]

[Display(Name = "Категория")]

public int CategoryID { get; set; }

[Required]

public string AuthorID { get; set; }

[Required]

public DateTime DT { get; set; }

[Required]

[Display(Name = "Тема")]

public string Subject { get; set; }

[Required]

[Display(Name = "Текст статьи")]

public string Text { get; set; }

}

Category

public class Category

{

public int ID { get; set; }

public string Name { get; set; }

}

Favorites

public class Favorites

{

public int ID { get; set; }

public string UserID {get; set;}

public int ArticleID { get; set; }

AccountViewModel

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace Lab1.Models

{

public class LoginViewModel

{

[Required]

[Display(Name = "Адрес электронной почты")]

[EmailAddress]

public string Email { get; set; }

[Required]

[DataType(DataType.Password)]

[Display(Name = "Пароль")]

public string Password { get; set; }

[Display(Name = "Запомнить меня")]

public bool RememberMe { get; set; }

}

public class RegisterViewModel

{

[Required]

[Display(Name = "Фамилия")]

public string LastName { get; set; }

[Required]

[Display(Name = "Имя")]

public string FirstName { get; set; }

[Display(Name = "Отчество")]

public string MiddleName { get; set; }

[DataType(DataType.Date)]

[Display(Name = "Дата рождения")]

public DateTime BirthDay { get; set; }

[Required]

[EmailAddress]

[Display(Name = "Email")]

public string Email { get; set; }

[Required]

[Display(Name = "Пол")]

public string Sex { get; set; }

[Phone]

[Display(Name = "Телефон")]

public string PhoneNumber { get; set; }

[Required]

[StringLength(100, ErrorMessage = "Значение {0} должно содержать не менее {2} символов.", MinimumLength = 6)]

[DataType(DataType.Password)]

[Display(Name = "Пароль")]

public string Password { get; set; }

[DataType(DataType.Password)]

[Display(Name = "Подтверждение пароля")]

[Compare("Password", ErrorMessage = "Пароль и его подтверждение не совпадают.")]

public string ConfirmPassword { get; set; }

}

}

Фрагмент IdentityModels

public class ApplicationUser : IdentityUser

{

public string FirstName { get; set; }

public string LastName { get; set; }

public string MiddleName { get; set; }

public DateTime BirthDay { get; set; }

public string Sex { get; set; }

public async Task<ClaimsIdentity> GenerateUserIdentityAsync(UserManager<ApplicationUser> manager)

{

// Обратите внимание, что authenticationType должен совпадать с типом, определенным в CookieAuthenticationOptions.AuthenticationType

var userIdentity = await manager.CreateIdentityAsync(this, DefaultAuthenticationTypes.ApplicationCookie);

// Здесь добавьте утверждения пользователя

return userIdentity;

}

}

public class ApplicationDbContext : IdentityDbContext<ApplicationUser>

{

public ApplicationDbContext()

: base("DefaultConnection", throwIfV1Schema: false)

{

}

public DbSet<Article> Articles { get; set; }

public DbSet<Category> Categorys { get; set; }

public DbSet<Favorites> Favorites { get; set; }

public static ApplicationDbContext Create()

{

return new ApplicationDbContext();

}

}

public class AppDbInitializer : DropCreateDatabaseAlways<ApplicationDbContext>

{

protected override void Seed(ApplicationDbContext db)

{

db.Categorys.Add(new Category { Name = "Новости" });

db.Categorys.Add(new Category { Name = "IT" });

db.Categorys.Add(new Category { Name = "HiTech" });

db.Categorys.Add(new Category { Name = "Кулинария" });

db.Categorys.Add(new Category { Name = "Кино" });

db.Categorys.Add(new Category { Name = "Музыка" });

db.Categorys.Add(new Category { Name = "Красота и здоровье" });

db.Categorys.Add(new Category { Name = "Транспорт" });

db.Categorys.Add(new Category { Name = "Финансы и бизнес" });

var roleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(db));

roleManager.Create(new IdentityRole("user"));

roleManager.Create(new IdentityRole("admin"));

ApplicationUserManager manager = new ApplicationUserManager(new UserStore<ApplicationUser>(db));

var user = new ApplicationUser

{

LastName = "Ковальчук",

FirstName = "Антон",

MiddleName = "Дмитриевич",

UserName = "antony.nor@gmail.com",

BirthDay = new DateTime(1996, 9, 28),

Email = "antony.nor@gmail.com",

Sex = "Мужской",

PhoneNumber = "+79158562772",

LockoutEnabled = false

};

var result = manager.Create(user, "123kT@");

if(result.Succeeded)

{

manager.AddToRole(user.Id, "admin");

}

ManageViewModel

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using Microsoft.AspNet.Identity;

using Microsoft.Owin.Security;

namespace Lab1.Models

{

public class IndexViewModel

{

public bool HasPassword { get; set; }

public IList<UserLoginInfo> Logins { get; set; }

public string PhoneNumber { get; set; }

public bool TwoFactor { get; set; }

public bool BrowserRemembered { get; set; }

}

public class ChangePasswordViewModel

{

[Required]

[DataType(DataType.Password)]

[Display(Name = "Текущий пароль")]

public string OldPassword { get; set; }

[Required]

[StringLength(100, ErrorMessage = "Значение {0} должно содержать символов не менее: {2}.", MinimumLength = 6)]

[DataType(DataType.Password)]

[Display(Name = "Новый пароль")]

public string NewPassword { get; set; }

[DataType(DataType.Password)]

[Display(Name = "Подтверждение нового пароля")]

[Compare("NewPassword", ErrorMessage = "Новый пароль и его подтверждение не совпадают.")]

public string ConfirmPassword { get; set; }

}

}

## Примеры работы программы

На рисунке 1 представлена страница Авторизации.

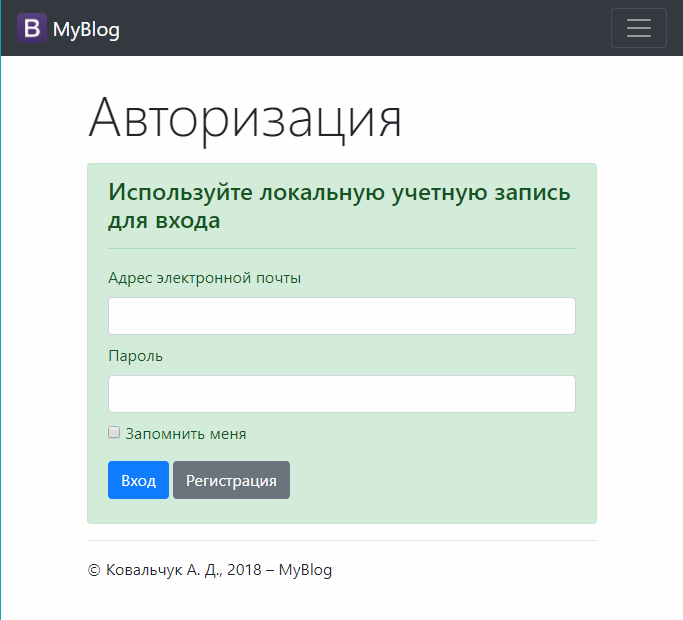


Рисунок 1 – Страница Авторизации

На рисунке 2 представлена страница Регистрации пользователя.

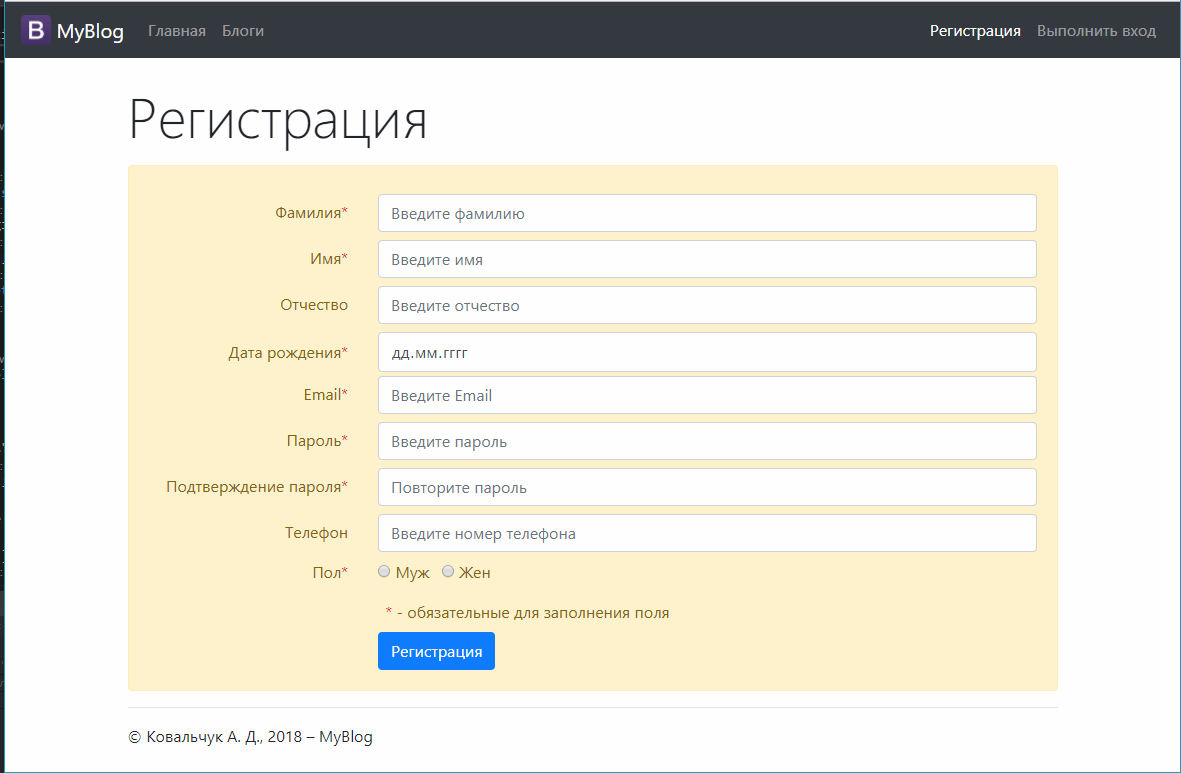


Рисунок 2 – Страница Регистрации пользователя

На рисунке 3 представлена страница Личной информации

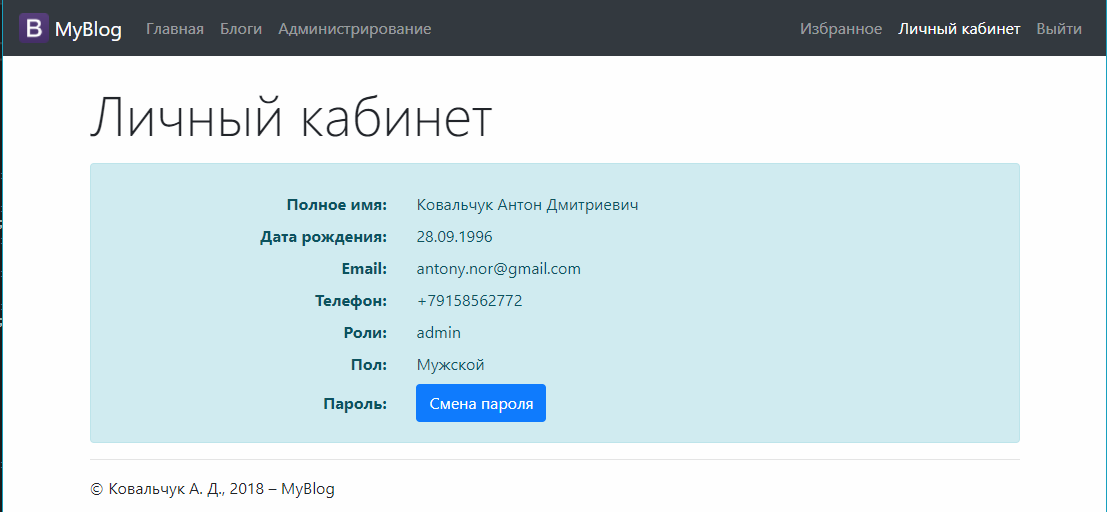


Рисунок 3 – Страница Личной информации

На рисунке 4 представлена страница «Избранное»

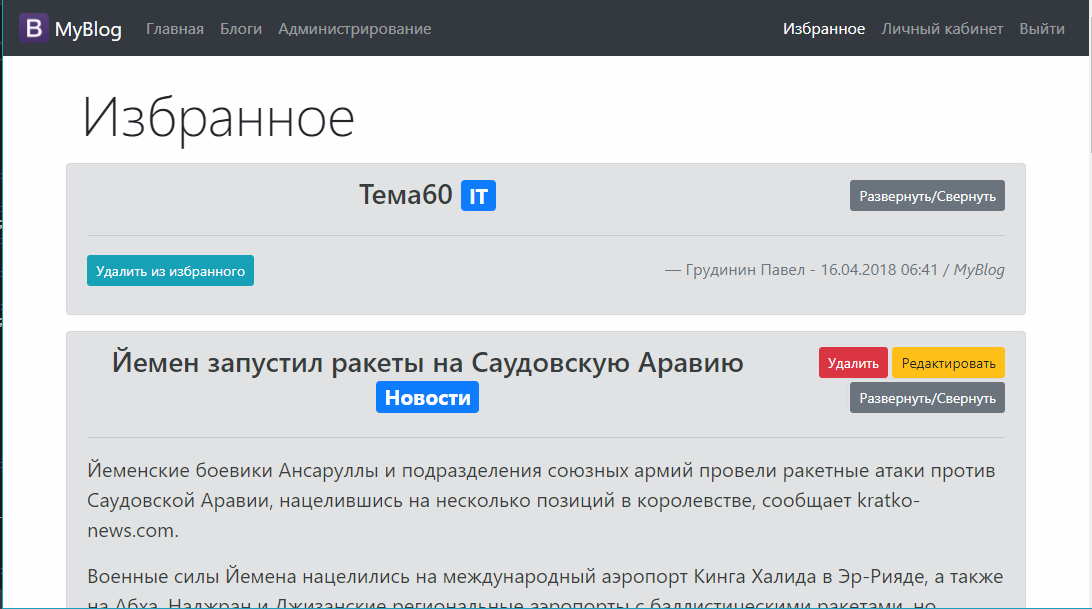


Рисунок 4 – Страница «Избранное»