1. Код программных модулей

Layout.cshtml

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>@ViewBag.Title Магазины, товары, продажи</title>

@Styles.Render("~/Content/css")

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

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

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

@RenderSection("scripts", required: false)

<style>

footer {

height: 100px;

position: fixed;

bottom: 0;

width: 100%;

}

</style>

</head>

<body>

<div class="navbar navbar-inverse navbar-fixed-top">

<div class="container">

<div class="navbar-header">

<button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

@Html.ActionLink("Good shops", "Index", "Home", null, new { @class = "navbar-brand" })

</div>

<div class="navbar-collapse collapse">

<ul class="nav navbar-nav">

<li>@Html.ActionLink("Главная", "Index", "Home")</li>

<li>@Html.ActionLink("О нас", "About", "Home")</li>

<li>@Html.ActionLink("Обратная связь", "Index", "Feedback")</li>

@if (Request.IsAuthenticated && User.IsInRole("Admin"))

{

<li>@Html.ActionLink("Управление магазинами", "ManageShops", "Admin")</li>

<li>@Html.ActionLink("RolesAdmin", "Index", "RolesAdmin")</li>

<li>@Html.ActionLink("UsersAdmin", "Index", "UsersAdmin")</li>

}

</ul>

@Html.Partial("\_LoginPartial")

</div>

</div>

</div>

<div class="container body-content">

@RenderBody()

<footer style="font-size: 12px">

<hr />

<div class="row">

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

Сейчас используется максимально упрощённая версия для добавления вашего магазина в каталог.

</div>

</div>

<div class="row">

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

<input type="button" class="btn-xm btn-primary" value="Я хочу добавить магазин" onclick="window.location.href = 'shops#/shops/create'">

</div>

</div>

</footer>

</div>

</body>

</html>

GSRepository.cs  
  
using System;

using System.Collections.Generic;

using System.Linq;

using GS.DAL;

using GS.DAL.Models;

using System.Data.Entity;

namespace GS.Core

{

public class GsRepository : IDisposable, IShopRepository

{

protected GSDbContext context = new GSDbContext();

public IEnumerable<Shop> GetAllShops()

{

return context.Shops.Include(item => item.User);

}

public Shop FindById(int shopId)

{

return context.Shops.FirstOrDefault(s => s.Id == shopId);

}

public IEnumerable<News> GetShopAllNewses(int shopId)

{

return context.Shops.FirstOrDefault(s => s.Id == shopId).News;

}

public IEnumerable<Product> GetAllShopProducts(int shopId)

{

return context.Shops.FirstOrDefault(s => s.Id == shopId).Products;

}

public bool CreateNewShop(string userId, Shop shop)

{

if (shop != null)

{

var user = context.AspNetUsers.First(u => u.Id == userId);

if (user.Shops == null)

{

user.Shops = new List<Shop>();

}

user.Shops.Add(shop);

context.SaveChanges();

return true;

}

return false;

}

protected void Dispose(bool disposing)

{

if (disposing)

{

if (context != null)

{

context.Dispose();

context = null;

}

}

}

public void Dispose()

{

this.Dispose(true);

GC.SuppressFinalize(this);

}

public void AddNewsToShop(string userId, string shopId, News news)

{

var user = context.AspNetUsers.First(u => u.Id == userId);

var shop = user.Shops.First(s => s.Id.ToString() == shopId);

shop.News.Add(news);

context.SaveChanges();

}

public News GetNews(int newsId)

{

return context.News.First(n => n.Id == newsId);

}

public void UpdateShopActive(int shopId, bool isAcitive)

{

Shop shop = context.Shops.First(s => s.Id == shopId);

shop.IsActive = isAcitive;

context.SaveChanges();

}

public void UpdateShopStatus(int shopId, State status)

{

Shop shop = context.Shops.First(s => s.Id == shopId);

shop.State = status;

context.SaveChanges();

}

public void UpdateShopStatus(int shopId, int status)

{

Shop shop = context.Shops.First(s => s.Id == shopId);

shop.State = (State) status;

context.SaveChanges();

}

}

}

ManageShops.cs  
  
@model IEnumerable<GS.DAL.Models.Shop>

@{

ViewBag.Title = "Управление магазинами";

}

<h2>Управление магазинами</h2>

<br />

<table id="tableManageShops" class="display" width="100%">

<thead>

<tr>

<th hidden>Id</th>

<th>Пользователь</th>

<th>Название магазина</th>

<th>Ссылка</th>

<th>Подтверждённый?</th>

<th>Активный?</th>

</tr>

</thead>

<tbody>

@foreach (var shop in Model)

{

<tr>

<td hidden>

@Html.DisplayFor(modelItem => shop.Id)

</td>

<td>

@Html.DisplayFor(modelItem => shop.User.UserName)

</td>

<td>

@Html.DisplayFor(modelItem => shop.Name)

</td>

<td>

..

</td>

<td>

<div class="btn-group" data-toggle="buttons">

@if ((int)shop.State == 1)

{

<label class="btn btn-default active">

<input type="radio" name="isApproved" value="1"> Да

</label>

<label class="btn btn-default">

<input type="radio" name="isApproved" value="0"> Пропустить

</label>

<label class="btn btn-default">

<input type="radio" name="isApproved" value="-1"> Нет

</label>

}

else if ((int)shop.State == 0)

{

<label class="btn btn-default">

<input type="radio" name="isApproved" value="1"> Да

</label>

<label class="btn btn-default active">

<input type="radio" name="isApproved" value="0"> Пропустить

</label>

<label class="btn btn-default">

<input type="radio" name="isApproved" value="-1"> Нет

</label>

}

else

{

<label class="btn btn-default">

<input type="radio" name="isApproved" value="1"> Да

</label>

<label class="btn btn-default">

<input type="radio" name="isApproved" value="0"> Пропустить

</label>

<label class="btn btn-default active">

<input type="radio" name="isApproved" value="-1"> Нет

</label>

}

</div>

</td>

<td>

<div class="btn-group" data-toggle="buttons">

@if (shop.IsActive)

{

<label class="btn btn-default active">

<input type="radio" name="isActive" value="true"> Да

</label>

<label class="btn btn-default">

<input type="radio" name="isActive" value="false"> Нет

</label>

}

else

{

<label class="btn btn-default">

<input type="radio" name="isActive" value="true"> Да

</label>

<label class="btn btn-default active">

<input type="radio" name="isActive" value="false"> Нет

</label>

}

</div>

</td>

</tr>

}

</tbody>

</table>

<link href="~/Scripts/dataTables/jquery.dataTables.min.css" rel="stylesheet" />

<script src="~/Scripts/dataTables/jquery.dataTables.min.js"></script>

<link rel="stylesheet" type="text/css" href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap3-dialog/1.34.5/css/bootstrap-dialog.min.css">

<script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap3-dialog/1.34.5/js/bootstrap-dialog.min.js"></script>

<script>

$(document).ready(function () {

$("input[type='radio']").on('change', function () {

if (this.value == '0') {

$(this).parents().eq(3).remove();

return;

}

confirm(this);

});

$('#tableManageShops').DataTable({

});

});

function confirm(button) {

var type = BootstrapDialog.TYPE\_WARNING;

var message = "";

if (button.name == 'isActive') {

if (button.value == 'true') {

message = "Магазин будет добавлен в каталог? Вы согласны?";

} else {

message = "Магазин не прошёл модерацию. Вы уверены?";

}

operation = "changeActiveShop";

} else {

if (button.value == '1') {

message = "Магазин будет разблокирован. Продолжить?";

} else {

message = "Магазин будет заблокирован. Вы уверены?";

}

operation = "changeStateShop";

}

var requestData = {

ActionType: operation,

ShopId: $(button).parents().eq(3).find('td').first().text().trim(),

Value: button.value

};

BootstrapDialog.show({

title: 'Подтвердите изменение информации!',

message: message,

type: type,

buttons: [

{

label: 'Да',

action: function (dialog) {

dialog.close();

$.ajax('ChangeStatusShop', {

type: 'POST',

data: JSON.stringify(requestData),

contentType: "application/json; charset=utf-8",

dataType: "json",

success: function () {

BootstrapDialog.alert({

title: 'Успешно',

message: 'Изменения были схранены!',

type: BootstrapDialog.TYPE\_SUCCESS

});

},

error: function (errMesage) {

BootstrapDialog.alert({

title: 'Внимание',

message: 'Не удалось сохранить!' + errMesage,

type: BootstrapDialog.TYPE\_DANGER

});

}

});

}

},

{

label: 'Нет',

action: function (dialog) {

dialog.close();

}

}

]

});

}

</script>

AdminController.cs  
  
using System;

using System.Collections.Generic;

using System.Runtime.Serialization;

using System.Web.Mvc;

using System.Web.Script.Serialization;

using GoodShops.Models;

using GoodShops.ProjectResources;

using GS.Core;

using GS.DAL.Models;

using WebGrease.Css.Ast.Selectors;

namespace GoodShops.Controllers

{

[System.Web.Mvc.Authorize(Roles = "Admin")]

public class AdminController : Controller

{

public ActionResult ManageShops()

{

GsRepository repository = new GsRepository();

IEnumerable<Shop> shops = repository.GetAllShops();

return View(shops);

}

public void ChangeStatusShop(ChangeStateShopModel obj)

{

var gsRepository = new GsRepository();

if (obj.ActionType == RequestConstants.ChangeStateShop)

{

gsRepository.UpdateShopStatus(obj.ShopId, Convert.ToInt32(obj.Value));

return;

}

gsRepository.UpdateShopActive(obj.ShopId, Convert.ToBoolean(obj.Value));

}

}

}

ChangeStateShopModel.cs

namespace GoodShops.Models

{

public class ChangeStateShopModel

{

public string ActionType { get; set; }

public string Value { get; set; }

public int ShopId { get; set; }

}

}

FeedbackModel.cs  
  
using System.ComponentModel.DataAnnotations;

namespace GoodShops.Models

{

public class FeedbackModel

{

[Required]

[Display(Name = "Имя пользователя")]

[StringLength(50)]

public string UserName { get; set; }

[Required]

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

[RegularExpression(@"^\w+([-+.']\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*$", ErrorMessage = "Неверный формат электронной почты")]

public string Email { get; set; }

[Required]

[StringLength(800)]

[Display(Name = "Сообщение")]

[DataType(DataType.MultilineText)]

public string MessageText { get; set; }

}

}

FeedbackController.cs  
  
sing System;

using System.Web.Helpers;

using System.Web.Mvc;

using GoodShops.Models;

using ProjectResources;

namespace GoodShops.Controllers

{

public class FeedbackController:Controller

{

[HttpGet]

public ActionResult Index()

{

var feedback = new FeedbackModel();

feedback.Email = User.Identity.Name;

feedback.UserName = User.Identity.Name;

return View(feedback);

}

[HttpPost]

public void Index(FeedbackModel feedbackModel)

{

try

{

WebMail.SmtpServer = "smtp.inbox.ru";

WebMail.SmtpPort = 2525;

WebMail.EnableSsl = true;

WebMail.UserName = MainRes.AdminEmail;

WebMail.Password = MainRes.AdminPassword;

var message = String.Format("Сообщение от {0}. Имя {1}. {2}", feedbackModel.Email,

feedbackModel.UserName, feedbackModel.MessageText);

WebMail.Send(User.Identity.Name, "Обратная связь", message, MainRes.AdminEmail);

}

catch (Exception)

{

}

}

}

}

Feedback/Index.cshtml  
  
@model GoodShops.Models.FeedbackModel

@{

ViewBag.Title = "Обратная связь";

}

<h2>Вы можете связаться с нами. Мы обязательно вам ответим.</h2>

@using (@Html.BeginForm("Index", "Feedback", FormMethod.Post, new { enctype = "multipart/form-data" }))

{

<fieldset>

<div class="row">

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

<div class="editor-label">

@Html.LabelFor(model => model.UserName)

</div>

</div>

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

@Html.EditorFor(model => model.UserName, new { htmlAttributes = new { @class = "form-control top-buffer" } })

</div>

</div>

<div class="row">

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

@Html.ValidationMessageFor(model => model.UserName)

</div>

</div>

<div class="row">

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

<div class="editor-label">

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

</div>

</div>

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

@Html.EditorFor(model => model.Email, new { htmlAttributes = new { @class = "form-control top-buffer" } })

</div>

</div>

<div class="row">

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

@Html.ValidationMessageFor(model => model.Email)

</div>

</div>

<div class="row">

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

<div class="editor-label">

@Html.LabelFor(model => model.MessageText)

</div>

</div>

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

@Html.EditorFor(model => model.MessageText, new { htmlAttributes = new { @class = "form-control top-buffer" } })

</div>

</div>

<div class="row">

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

@Html.ValidationMessageFor(model => model.MessageText)

</div>

</div>

</fieldset>

<input type="submit" value="Отправить" class="btn btn-default" />

}

<style>

.top-buffer {

margin-bottom: 3px;

}

</style>

IdentityConfig.cs  
  
using System.Linq;

using System.Security.Claims;

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.EntityFramework;

using Microsoft.AspNet.Identity.Owin;

using Microsoft.Owin;

using Microsoft.Owin.Security;

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Threading.Tasks;

using System.Web;

namespace IdentitySample.Models

{

public class ApplicationUserManager : UserManager<ApplicationUser>

{

public ApplicationUserManager(IUserStore<ApplicationUser> store)

: base(store)

{

}

public static ApplicationUserManager Create(IdentityFactoryOptions<ApplicationUserManager> options,

IOwinContext context)

{

var manager = new ApplicationUserManager(new UserStore<ApplicationUser>(context.Get<ApplicationDbContext>()));

manager.UserValidator = new UserValidator<ApplicationUser>(manager)

{

AllowOnlyAlphanumericUserNames = false,

RequireUniqueEmail = true

};

manager.PasswordValidator = new PasswordValidator

{

RequiredLength = 6

};

manager.UserLockoutEnabledByDefault = true;

manager.DefaultAccountLockoutTimeSpan = TimeSpan.FromMinutes(5);

manager.MaxFailedAccessAttemptsBeforeLockout = 5;

manager.RegisterTwoFactorProvider("EmailCode", new EmailTokenProvider<ApplicationUser>

{

Subject = "SecurityCode",

BodyFormat = "Your security code is {0}"

});

manager.EmailService = new EmailService();

var dataProtectionProvider = options.DataProtectionProvider;

if (dataProtectionProvider != null)

{

manager.UserTokenProvider =

new DataProtectorTokenProvider<ApplicationUser>(dataProtectionProvider.Create("ASP.NET Identity"));

}

return manager;

}

}

public class ApplicationRoleManager : RoleManager<IdentityRole>

{

public ApplicationRoleManager(IRoleStore<IdentityRole,string> roleStore)

: base(roleStore)

{

}

public static ApplicationRoleManager Create(IdentityFactoryOptions<ApplicationRoleManager> options, IOwinContext context)

{

return new ApplicationRoleManager(new RoleStore<IdentityRole>(context.Get<ApplicationDbContext>()));

}

}

public class EmailService : IIdentityMessageService

{

public Task SendAsync(IdentityMessage message)

{

return Task.FromResult(0);

}

}

public class SmsService : IIdentityMessageService

{

public Task SendAsync(IdentityMessage message)

{

return Task.FromResult(0);

}

}

public class ApplicationSignInManager : SignInManager<ApplicationUser, string>

{

public ApplicationSignInManager(ApplicationUserManager userManager, IAuthenticationManager authenticationManager) :

base(userManager, authenticationManager) { }

public override Task<ClaimsIdentity> CreateUserIdentityAsync(ApplicationUser user)

{

return user.GenerateUserIdentityAsync((ApplicationUserManager)UserManager);

}

public static ApplicationSignInManager Create(IdentityFactoryOptions<ApplicationSignInManager> options, IOwinContext context)

{

return new ApplicationSignInManager(context.GetUserManager<ApplicationUserManager>(), context.Authentication);

}

}

}

Login.cshtml

@model IdentitySample.Models.LoginViewModel

@{

ViewBag.Title = "Вход";

}

<div class="row">

<div class="col-md-7">

<section id="loginForm">

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

{

@Html.AntiForgeryToken()

<h2>Используйте свои данные для входа</h2>

<hr />

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

<div class="form-group">

@Html.LabelFor(m => m.Email, new { @class = "col-md-4 control-label" })

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

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

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

</div>

</div>

<div class="form-group">

@Html.LabelFor(m => m.Password, new { @class = "col-md-4 control-label" })

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

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

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

</div>

</div>

<div class="form-group">

<div class="col-md-offset-4 col-md-8">

<div class="checkbox">

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

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

</div>

</div>

</div>

<div class="form-group">

<div class="col-md-offset-4 col-md-8">

<input type="submit" value="Войти" class="btn btn-default" />

</div>

</div>

<p>

@Html.ActionLink("Зарегистрировать нового пользователя?", "Register")

</p>

<p>

@Html.ActionLink("Забыли пароль?", "ForgotPassword")

</p>

}

</section>

</div>

<div class="col-md-5">

<section id="socialLoginForm">

@Html.Partial("\_ExternalLoginsListPartial", new IdentitySample.Models.ExternalLoginListViewModel { ReturnUrl = ViewBag.ReturnUrl })

</section>

</div>

</div>

@section Scripts {

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

}

Register.cshtml

@model IdentitySample.Models.RegisterViewModel

@{

ViewBag.Title = "Регистрация";

}

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

{

@Html.AntiForgeryToken()

<br/>

<h2>Регистрация нового пользователя</h2>

<hr />

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

<div class="form-group">

@Html.LabelFor(m => m.Email, new { @class = "col-md-2 control-label" })

<div class="col-md-10">

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

</div>

</div>

<div class="form-group">

@Html.LabelFor(m => m.Password, new { @class = "col-md-2 control-label" })

<div class="col-md-10">

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

</div>

</div>

<div class="form-group">

@Html.LabelFor(m => m.ConfirmPassword, new { @class = "col-md-2 control-label" })

<div class="col-md-10">

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

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

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

</div>

</div>

}

@section Scripts {

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

}

AccountController.cs  
  
using System.Globalization;

using IdentitySample.Models;

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.Owin;

using Microsoft.Owin.Security;

using System;

using System.Linq;

using System.Security.Claims;

using System.Threading.Tasks;

using System.Web;

using System.Web.Mvc;

using ProjectResources;

namespace IdentitySample.Controllers

{

[Authorize]

public class AccountController : Controller

{

public AccountController()

{

}

public AccountController(ApplicationUserManager userManager, ApplicationSignInManager signInManager )

{

UserManager = userManager;

SignInManager = signInManager;

}

private ApplicationUserManager \_userManager;

public ApplicationUserManager UserManager

{

get

{

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

}

private set

{

\_userManager = value;

}

}

//

// GET: /Account/Login

[AllowAnonymous]

public ActionResult Login(string returnUrl)

{

ViewBag.ReturnUrl = returnUrl;

return View();

}

private ApplicationSignInManager \_signInManager;

public ApplicationSignInManager SignInManager

{

get

{

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

}

private set { \_signInManager = value; }

}

//

// POST: /Account/Login

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

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

{

if (!ModelState.IsValid)

{

return View(model);

}

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 });

case SignInStatus.Failure:

default:

ModelState.AddModelError("", "Invalid login attempt.");

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 { UserName = model.Email, Email = model.Email };

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

if (result.Succeeded)

{

var code = await UserManager.GenerateEmailConfirmationTokenAsync(user.Id);

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

var emailBody =

string.Format(

"[{0}] Пожалуйста, подтвердите регистрацию перейдя по ссылке: <a href=\"{1}\">link</a>",

MainRes.ProjectName, callbackUrl);

var emailTitle = string.Format("[{0}]Подтверждения почты", MainRes.ProjectName);

await UserManager.SendEmailAsync(user.Id, emailTitle, emailBody);

ViewBag.Link = callbackUrl;

return View("DisplayEmail");

}

AddErrors(result);

}

// If we got this far, something failed, redisplay form

return View(model);

}

// POST: /Account/LogOff

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult LogOff()

{

AuthenticationManager.SignOut();

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

}

}

}