**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

ФАИС

Кафедра «Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5

по дисциплине **«ИГИ»**

# на тему: «Обработка HTTP запросов средствами фреймворка ASP.NET MVC»

Выполнил: студент гр. ИП-32

Новик В.В.

Принял: доцент Асенчик О.Д.

Дата сдачи отчета: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата допуска к защите: ­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гомель 2019

**Цель работы.** Получить навыки создания типовых web-приложений для работы с информацией из реляционных баз данных, ознакомиться со способами аутентификации и авторизации в ASP.NET MVC Сore.

**ЗАДАНИЕ**

Используя разработанное ранее создать интерфейс Web-приложения для работы с данными, хранящимися в базе данных, на основе ASP.NET Core MVC Framework и Entity Framework Core.

Web- приложение должно удовлетворять следующим требованиям:

* Любой пользователь для работы с приложением должен пройти аутентификацию.
* Поддерживать реализацию не менее двух ролевых политик для доступа к классам и (или) методам контроллеров.
* Управлять пользователями: просматривать, создавать, удалять и редактировать данные учетных записей.
* Осуществлять ввод, редактирование, добавление и просмотр данных не менее чем из трех таблиц реляционной базы согласно варианту.
* Вводимые и редактируемые данные должны проходить валидацию на стороне сервера и клиента.
* Осуществлять разбиение таблично представленных данных, выбранных их реляционного источника, на страницы, их фильтрацию и сортировку.
* Сохранять состояния элементов форм, используемых для фильтрации данных, и использовать их при повторном посещении соответствующей Web-страницы.
* Использовать не менее одного разработанного html- хэлпера и не менее одного Tag-хэлпера. Например, хэлперы для разбиения на страницы, сортировки, генерации табличных данных и др.
* Использовать не менее одного частичного представления (Partial View).
* Иметь единое стилевое оформление, основанное на использовании одной или нескольких мастер-страниц, и систему навигации (строка меню, гиперссылки, кнопки), которая обеспечивает оптимальный путь перехода между двумя произвольно выбранными страницами в соответствии с логикой приложения.

Представления разработанного приложения, работающими с таблицами, стоящими на стороне отношения «многие» в схеме базы данных, должны выводить вместо кодов внешних ключей смысловые значения из связанных таблиц, стоящих на стороне отношения «один».

Для проверки преподавателем следует разместить разработанный проект на GitHub.

Коды класса (классов) контроллеров.

public class AccountController : Controller

{

private readonly UserManager<IdentityUser> \_userManager;

private readonly SignInManager<IdentityUser> \_signInManager;

public AccountController(UserManager<IdentityUser> userManager, SignInManager<IdentityUser> signInManager)

{

\_userManager = userManager;

\_signInManager = signInManager;

}

public IActionResult Index()

{

return View();

}

[HttpGet]

public IActionResult Login(string returnUrl = null)

{

return View(new LoginViewModel { ReturnUrl = returnUrl });

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Login(LoginViewModel model)

{

if (ModelState.IsValid)

{

var result =

await \_signInManager.PasswordSignInAsync(model.Login, model.Password, model.RememberMe, false);

if (result.Succeeded)

{

if (!string.IsNullOrEmpty(model.ReturnUrl) && Url.IsLocalUrl(model.ReturnUrl))

{

return Redirect(model.ReturnUrl);

}

else

{

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

}

}

else

{

ModelState.AddModelError("", "Неправильный логин и (или) пароль");

}

}

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> LogOff()

{

await \_signInManager.SignOutAsync();

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

}

}

[Authorize(Roles = "admin")]

public class UserController : Controller

{

private readonly UserManager<IdentityUser> \_userManager;

public UserController(UserManager<IdentityUser> userManager)

{

\_userManager = userManager;

}

public IActionResult Index()

{

return View(\_userManager.Users.ToList());

}

[HttpGet]

public IActionResult Create()

{

return View();

}

[HttpPost]

public async Task<IActionResult> Create(CreateViewModel model)

{

if (ModelState.IsValid)

{

IdentityUser user = new IdentityUser { Email = model.Email, UserName = model.Login };

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

if (result.Succeeded)

{

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

}

else

{

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

}

}

return View(model);

}

public async Task<IActionResult> Edit(string id)

{

IdentityUser user = await \_userManager.FindByIdAsync(id);

if (user == null)

{

return NotFound();

}

EditUserViewModel model = new EditUserViewModel { Id = user.Id, Email = user.Email, Login = user.UserName };

return View(model);

}

[HttpPost]

public async Task<IActionResult> Edit(EditUserViewModel model)

{

if (ModelState.IsValid)

{

IdentityUser user = await \_userManager.FindByIdAsync(model.Id);

if (user != null)

{

user.Email = model.Email;

user.UserName = model.Login;

var result = await \_userManager.UpdateAsync(user);

if (result.Succeeded)

{

return RedirectToAction("Index");

}

else

{

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

}

}

}

return View(model);

}

[HttpPost]

public async Task<ActionResult> Delete(string id)

{

IdentityUser user = await \_userManager.FindByIdAsync(id);

if (user != null)

{

IdentityResult result = await \_userManager.DeleteAsync(user);

}

return RedirectToAction("Index");

}

public async Task<IActionResult> ChangePassword(string id)

{

IdentityUser user = await \_userManager.FindByIdAsync(id);

if (user == null)

{

return NotFound();

}

ChangePasswordViewModel model = new ChangePasswordViewModel { Id = user.Id, Login = user.UserName };

return View(model);

}

[HttpPost]

public async Task<IActionResult> ChangePassword(ChangePasswordViewModel model)

{

if (ModelState.IsValid)

{

IdentityUser user = await \_userManager.FindByIdAsync(model.Id);

if (user != null)

{

IdentityResult result =

await \_userManager.ChangePasswordAsync(user, model.Password, model.NewPassword);

if (result.Succeeded)

{

return RedirectToAction("Index");

}

else

{

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

}

}

else

{

ModelState.AddModelError(string.Empty, "Пользователь не найден");

}

}

return View(model);

}

}

Коды классов разработанных хэлперов.

public static class PageHtmlHelper

{

public static HtmlString CreateLinksToPages(this IHtmlHelper html, PageViewModel pageView)

{

TagBuilder div = new TagBuilder("div");

if (pageView.HasPreviousPage) {

TagBuilder a = new TagBuilder("a");

a.Attributes.Add("href", "Index?page=" + (pageView.PageNumber - 1));

a.Attributes.Add("class ", "btn btn-default btn");

a.InnerHtml.Append("Назад");

div.InnerHtml.AppendHtml(a);

}

if (pageView.HasNextPage) {

TagBuilder a = new TagBuilder("a");

a.Attributes.Add("href", "Index?page=" + (pageView.PageNumber + 1));

a.Attributes.Add("class ", "btn btn-default btn");

a.InnerHtml.Append("Вперёд");

div.InnerHtml.AppendHtml(a);

}

var writer = new System.IO.StringWriter();

div.WriteTo(writer, HtmlEncoder.Default);

return new HtmlString(writer.ToString());

}

}

public class PageLinkTagHelper : TagHelper

{

private IUrlHelperFactory urlHelperFactory;

public PageLinkTagHelper(IUrlHelperFactory helperFactory)

{

urlHelperFactory = helperFactory;

}

[ViewContext]

[HtmlAttributeNotBound]

public ViewContext ViewContext { get; set; }

public PageViewModel PageModel { get; set; }

public string PageAction { get; set; }

public override void Process(TagHelperContext context, TagHelperOutput output)

{

IUrlHelper urlHelper = urlHelperFactory.GetUrlHelper(ViewContext);

output.TagName = "div";

TagBuilder tag = new TagBuilder("ul");

tag.AddCssClass("pagination");

if (PageModel.HasPreviousPage)

{

TagBuilder prevItem = CreateTag(PageModel.PageNumber - 1, urlHelper, "Назад");

tag.InnerHtml.AppendHtml(prevItem);

}

if (PageModel.HasNextPage)

{

TagBuilder nextItem = CreateTag(PageModel.PageNumber + 1, urlHelper, "Вперёд");

tag.InnerHtml.AppendHtml(nextItem);

}

output.Content.AppendHtml(tag);

}

TagBuilder CreateTag(int pageNumber, IUrlHelper urlHelper, string text)

{

TagBuilder item = new TagBuilder("li");

TagBuilder link = new TagBuilder("a");

if (pageNumber == this.PageModel.PageNumber)

{

item.AddCssClass("active");

}

else

{

link.Attributes["href"] = urlHelper.Action(PageAction, new { page = pageNumber });

}

link.InnerHtml.Append(text);

item.InnerHtml.AppendHtml(link);

return item;

}

}

Исходные коды представлений (Views).

@model lab3.ViewModels.Account.LoginViewModel

@{

ViewData["Title"] = "Login";

}

<h2>Вход в приложение</h2>

@if (!User.Identity.IsAuthenticated)

{

<**form** method="post" **asp-controller**="Account" **asp-action**="Login"

**asp-route-returnUrl**="@Model.ReturnUrl">

<**div** **asp-validation-summary**="ModelOnly"></**div**>

<div>

<**label** **asp-for**="Login"></**label**><br />

<**input** **asp-for**="Login" />

<**span** **asp-validation-for**="Login"></**span**>

</div>

<div>

<**label** **asp-for**="Password"></**label**><br />

<**input** **asp-for**="Password" />

<**span** **asp-validation-for**="Password"></**span**>

</div>

<div>

<**label** **asp-for**="RememberMe"></**label**><br />

<**input** **asp-for**="RememberMe" />

</div>

<div>

<input type="submit" value="Войти" />

</div>

</**form**>

}

else

{

<div>Вы уже авторизованы</div>

}

@model IEnumerable<Microsoft.AspNetCore.Identity.IdentityUser>

@{

ViewBag.Title = "Список пользователей";

}

<**a** **asp-action**="Create">Добавить пользователя</**a**>

<table class="table">

<tr><th>Логин</th><th>Почта</th></tr>

@foreach (var user in Model)

{

<tr>

<td>@user.UserName</td>

<td>@user.Email</td>

<td>

<**form** **asp-action**="Delete" **asp-route-id**="@user.Id" method="post">

<**a** class="btn btn-sm btn-primary" **asp-action**="ChangePassword" **asp-route-id**="@user.Id">Сменить пароль</**a**>

<**a** class="btn btn-sm btn-primary" **asp-action**="Edit" **asp-route-id**="@user.Id">Изменить</**a**>

<button type="submit" class="btn btn-sm btn-danger">

Удалить

</button>

</**form**>

</td>

</tr>

}

</table>

@model lab3.ViewModels.Account.CreateViewModel

@{

ViewData["Title"] = "Register";

}

<h2>Регистрация нового пользователя</h2>

<**form** method="post" **asp-controller**="Account" **asp-action**="Register">

<**div** **asp-validation-summary**="ModelOnly"></**div**>

<div>

<**label** **asp-for**="Login"></**label**><br />

<**input** **asp-for**="Login" />

<**span** **asp-validation-for**="Login"></**span**>

</div>

<div>

<**label** **asp-for**="Email"></**label**><br />

<**input** **asp-for**="Email" />

<**span** **asp-validation-for**="Email"></**span**>

</div>

<div>

<**label** **asp-for**="Password"></**label**><br />

<**input** **asp-for**="Password" />

<**span** **asp-validation-for**="Password"></**span**>

</div>

<div>

<**label** **asp-for**="PasswordConfirm"></**label**><br />

<**input** **asp-for**="PasswordConfirm" />

<**span** **asp-validation-for**="PasswordConfirm"></**span**>

</div>

<div>

<input type="submit" value="Регистрация" />

</div>

</**form**>

Исходные коды частичных представлений (Partial View).

@using lab3.Models

@model PageViewModel

<div>

@if (Model.HasPreviousPage)

{

<**a** **asp-action**="Index"

**asp-route-page**="@(Model.PageNumber - 1)"

class="btn btn-default btn">

<i class="glyphicon glyphicon-chevron-left"></i>

Назад

</**a**>

}

@if (Model.HasNextPage)

{

<**a** **asp-action**="Index"

**asp-route-page**="@(Model.PageNumber + 1)"

class="btn btn-default btn">

Вперед

<i class="glyphicon glyphicon-chevron-right"></i>

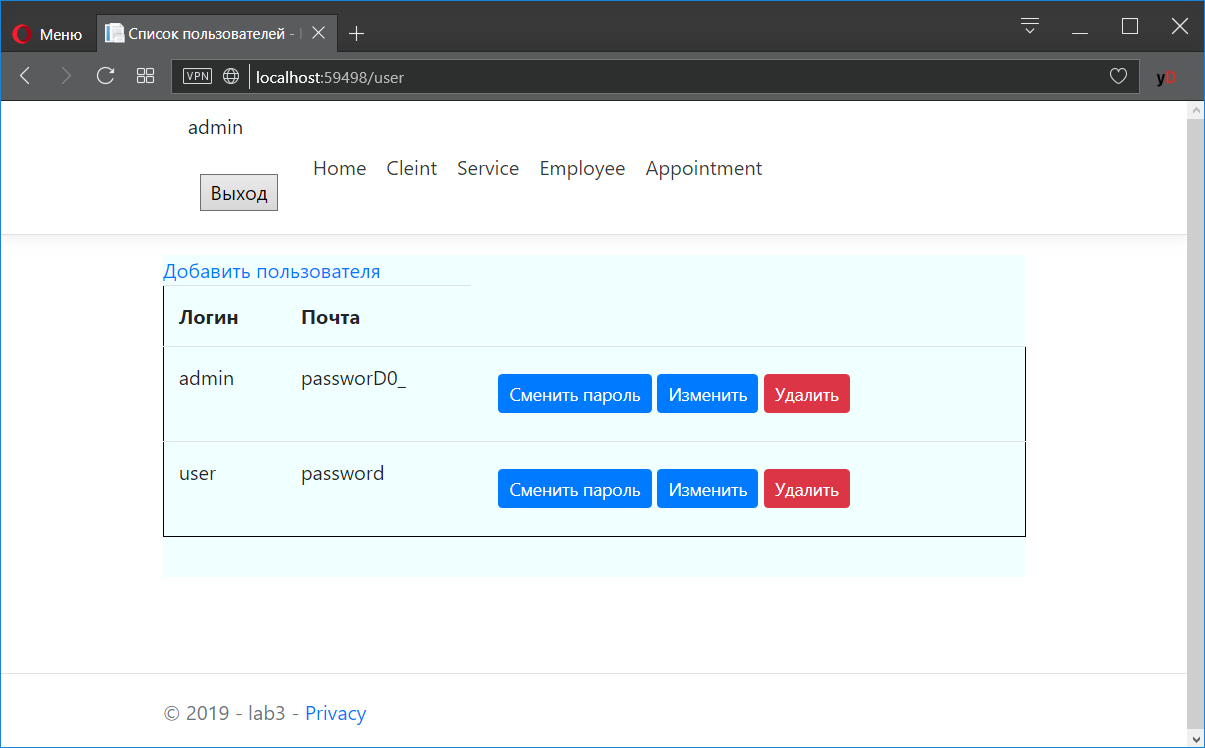
</**a**>

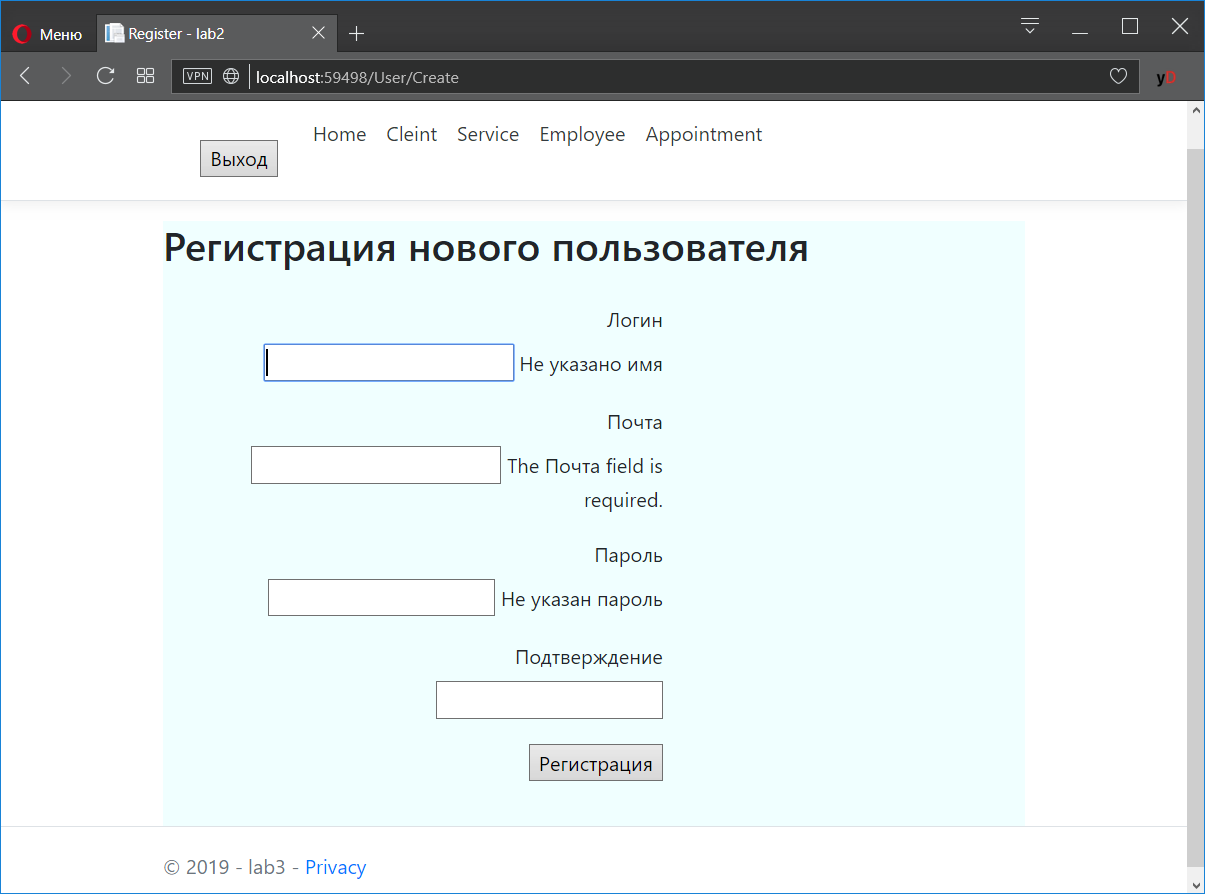
}

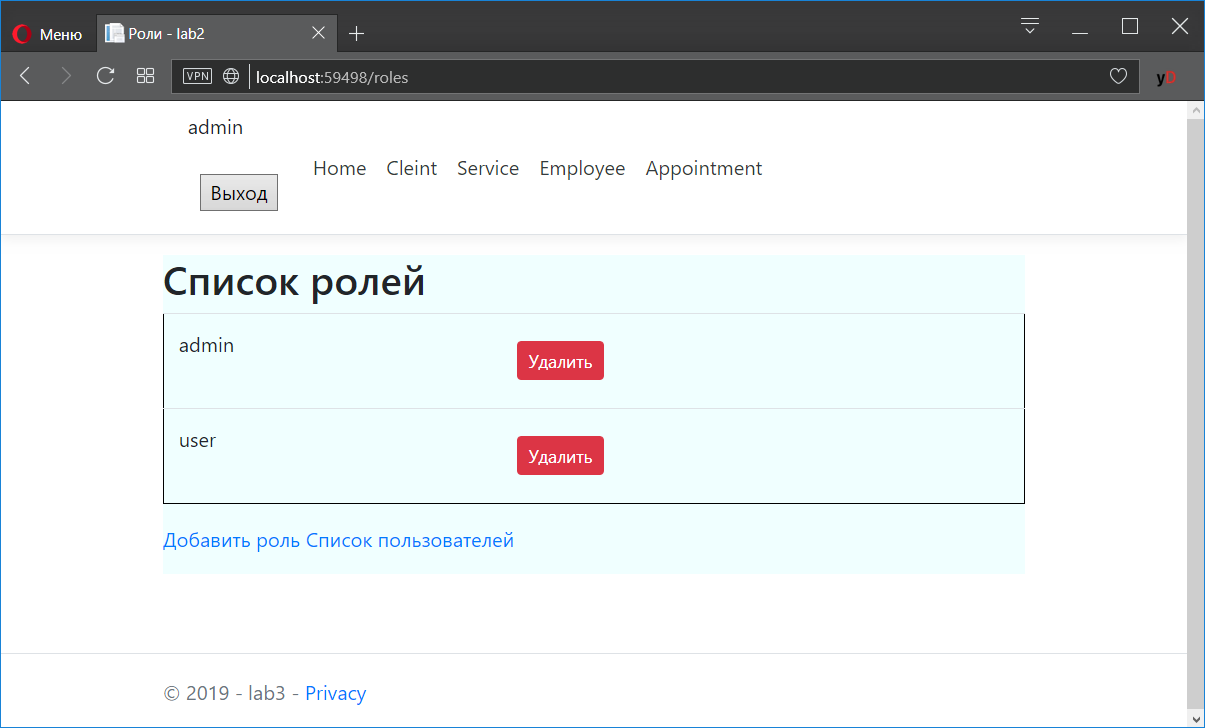
</div>

Копии экранов браузера, демонстрирующие отображение представлений (Views).

Копии экранов браузера, демонстрирующие отображение не прошедших валидацию данных.







**Вывод**. Изучил и использовал способы аутентификации и авторизации в ASP.NET MVC Сore.