МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО»

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

дисциплина «Избранные главы информатики»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

«**Обработка HTTP запросов средствами фреймворка ASP.NET MVC**»

Вариант № 16

Выполнил:

студент группы ИП-31, Самусев Р.А.

Принял:

доцент Асенчик О.Д.

Гомель 2018

**Цель работы:** Ознакомиться с особенностями реализации шаблона проектирования Model-View-Controller в ASP.NET Core и его возможностями для обработки поступающих от HTTP-клиента запросов посредством контроллеров и фильтров, а также способами взаимодействия с реляционными источниками посредством Entity Framework Core и приемами визуализации данных

**2. ЗАДАНИЕ**

Используя ранее разработанные объектную модель и простое приложение (сайт) в заданной предметной области, создать с использованием ASP.NET Core MVC Web-приложение содержащее:

2.1. классы (Models) моделирующие не менее чем три таблицы базы данных согласно вашему варианту;

2.2. класс контекста данных;

2.3. конфигурационный *appsettings.json* файл формата JSON, в котором хранятся настройки для создания соединения с базой данных.

2.4. классы контроллеров (Controllers) (по одному на каждую таблицу базы данных) для обработки HTTP запросов пользователя, выборки с использованием механизма внедрение зависимостей данных из таблиц, фильтрации и сортировки данных и вызова соответствующих представлений для отображения данных; контроллеры должны считывать состояния приложения из объекта Session и передавать эти значения в представления;

2.5. представления (Views) для отображения данных, выбранных контроллерами, позволяющие осуществлять фильтрацию и сортировку по одному или нескольким полям.

2.6. классы фильтры для контроллеров и (или) их методов действий, которые бы реализовывали:

* сохранение состояния элементов форм в виде объекта специальной структуры с использованием объекта Session;
* сохранение состояния сортировки на страницах в виде объекта специальной структуры с использованием объекта Session\*;
* ведение журнала действий пользователя путем записи фактов обращений к методам действий контроллера (альтернатива на выбор: кэширование представления);
* обработку исключений.

2.7. другие классы (классы моделей представлений (ViewModels), классы сервисов и др.).

*\* - на оценку 10 (десять).*

Для проверки преподавателем следует разместить разработанный проект на GitHub.

**Ход выполнения работы**

Ссылка проекта на github: <https://github.com/RSamuseu/RSamuseu/tree/lab4/lab4>

**Листинг класса Startup.cs**

using System.Threading.Tasks;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using AdvertisingAgency.Models;

using Microsoft.EntityFrameworkCore;

using Microsoft.AspNetCore.Mvc;

namespace AdvertisingAgency

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

public void ConfigureServices(IServiceCollection services)

{

string connection = Configuration.GetConnectionString("SQLConnection");

services.AddDbContext<adAgencyContext>(options => options.UseSqlServer(connection));

services.AddTransient<adAgencyContext>();

services.AddMemoryCache();

services.AddResponseCaching();

services.AddDistributedMemoryCache();

services.AddSession();

services.AddMvc().AddSessionStateTempDataProvider();

services.AddMvc(options =>

{

options.CacheProfiles.Add("Caching",

new CacheProfile()

{

Duration = 2 \* 16 + 240,

Location = ResponseCacheLocation.Any

});

options.CacheProfiles.Add("NoCaching",

new CacheProfile()

{

Location = ResponseCacheLocation.None,

NoStore = true

});

});

}

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

app.UseLastElementCache();

app.UseSession();

if (env.IsDevelopment())

{

app.UseBrowserLink();

app.UseDeveloperExceptionPage();

}

else

{

app.UseExceptionHandler("/Home/Error");

}

app.UseStaticFiles();

app.UseResponseCaching();

app.UseMvcWithDefaultRoute();

app.UseMvc(routes =>

{

routes.MapRoute(

name: "default",

template: "{controller=Home}/{action=Index}/{id?}");

});

}

}

}

**Листинг класса контекста данных adAgencyContext.cs**

using System;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using System.IO;

namespace AdvertisingAgency.Models

{

public class adAgencyContext:DbContext

{

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

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

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

public adAgencyContext(DbContextOptions<adAgencyContext> options)

: base(options)

{

}

}

}

**Листинги классов-контроллеров:**

HomeController.cs

using System;

using System.Diagnostics;

using System.Linq;

using Microsoft.AspNetCore.Mvc;

using AdvertisingAgency.Models;

using Newtonsoft.Json;

using Microsoft.AspNetCore.Http;

using Microsoft.Extensions.Caching.Memory;

using AdvertisingAgency.Filtres;

namespace AdvertisingAgency.Controllers

{

[SetToSession]

public class HomeController : Controller

{

public enum SortState

{

TypeAsc, // по типу по возрастанию

TypeDesc, // по типу по убыванию

DescriptAsc, // по описанию по возрастанию

DescriptDesc // по описанию по убыванию

}

readonly IMemoryCache \_memoryCache;

readonly adAgencyContext \_context;

public HomeController(IMemoryCache memoryCache, adAgencyContext context)

{

\_memoryCache = memoryCache;

\_context = context;

}

[ResponseCache(CacheProfileName = "Caching")]

public IActionResult Index(string Adtype, SortState sortType = SortState.TypeAsc)

{

Advedirsments adFromMemoryCashe = (Advedirsments)\_memoryCache.Get(Request.Path.Value.ToLower());

ViewBag.adFromMemory = adFromMemoryCashe;

if (HttpContext.Session.Get("adSession") != null)

{

Advedirsments adSession = JsonConvert.DeserializeObject<Advedirsments>(HttpContext.Session.GetString("adSession"));

ViewBag.adSession = adSession;

}

if (HttpContext.Session.Get("sortTypeSession") != null)

{

sortType = JsonConvert.DeserializeObject<SortState>(HttpContext.Session.GetString("sortTypeSession"));

}

IQueryable<Advedirsments> advedirsments = \_context.Advedirsments;

if (!String.IsNullOrEmpty(Adtype))

{

advedirsments = advedirsments.Where(a => a.AdType.Contains(Adtype));

}

ViewBag.AdTypeSort = sortType == SortState.TypeAsc ? SortState.TypeDesc : SortState.TypeAsc;

ViewBag.DesriptionSort = sortType == SortState.DescriptAsc ? SortState.DescriptDesc : SortState.DescriptAsc;

switch (sortType)

{

case SortState.TypeAsc:

advedirsments = advedirsments.OrderBy(t => t.AdType);

break;

case SortState.TypeDesc:

advedirsments = advedirsments.OrderByDescending(t => t.AdType);

break;

case SortState.DescriptAsc:

advedirsments = advedirsments.OrderBy(d => d.AdDescription);

break;

case SortState.DescriptDesc:

advedirsments = advedirsments.OrderByDescending(d => d.AdDescription);

break;

default:

advedirsments = advedirsments.OrderBy(t => t.AdType);

break;

}

ViewBag.advedirsments = advedirsments.ToList();

return View();

}

[HttpPost]

public IActionResult AddType(Advedirsments advedirsment)

{

\_context.Advedirsments.Add(advedirsment);

\_context.SaveChanges();

return RedirectToAction("Index");

}

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}

CustomerController.cs

using System;

using System.Linq;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Http;

using AdvertisingAgency.Models;

using Microsoft.Extensions.Caching.Memory;

using Newtonsoft.Json;

using AdvertisingAgency.Filtres;

namespace AdvertisingAgency.Controllers

{

[SetToSession]

public class CustomerController : Controller

{

public enum SortState

{

NameAsc, // по имени по возрастанию

NameDesc, // по имени по убыванию

AddressAsc, // по адресу по возрастанию

AddressDesc, // по адресу по убыванию

TelephoneAsc, // по телефону по возрастанию

TelephoneDesc // по телефону по убыванию

}

readonly IMemoryCache \_memoryCache;

readonly adAgencyContext \_context;

public CustomerController(IMemoryCache memoryCache, adAgencyContext context)

{

\_memoryCache = memoryCache;

\_context = context;

}

[ResponseCache(CacheProfileName = "Caching")]

public IActionResult Customer(string CustomerName, SortState sortCustomer = SortState.NameAsc)

{

Customers customersFromMemoryCashe = (Customers)\_memoryCache.Get(Request.Path.Value.ToLower());

ViewData["customersFromMemoryCashe"] = customersFromMemoryCashe;

if (HttpContext.Session.Get("customerSession") != null)

{

Customers customerSession = JsonConvert.DeserializeObject<Customers>(HttpContext.Session.GetString("customerSession"));

ViewBag.customerSession = customerSession;

}

if (HttpContext.Session.Get("customerSortSession") != null)

{

sortCustomer = JsonConvert.DeserializeObject<SortState>(HttpContext.Session.GetString("customerSortSession"));

}

IQueryable<Customers> customers = \_context.Customers;

if (!String.IsNullOrEmpty(CustomerName))

{

customers = customers.Where(c => c.CustomerName.Contains(CustomerName));

}

ViewBag.CustomerNameSort = sortCustomer == SortState.NameAsc ? SortState.NameDesc : SortState.NameAsc;

ViewBag.CustomerAddressSort = sortCustomer == SortState.AddressAsc ? SortState.AddressDesc : SortState.AddressAsc;

ViewBag.CustomerTelephoneSort = sortCustomer == SortState.TelephoneAsc ? SortState.TelephoneDesc : SortState.TelephoneAsc;

switch (sortCustomer)

{

case SortState.NameAsc:

customers = customers.OrderBy(c => c.CustomerName);

break;

case SortState.NameDesc:

customers = customers.OrderByDescending(c => c.CustomerName);

break;

case SortState.AddressAsc:

customers = customers.OrderBy(c => c.CustomerAddress);

break;

case SortState.AddressDesc:

customers = customers.OrderByDescending(c => c.CustomerAddress);

break;

case SortState.TelephoneAsc:

customers = customers.OrderBy(c => c.CustomerTelephone);

break;

case SortState.TelephoneDesc:

customers = customers.OrderByDescending(c => c.CustomerTelephone);

break;

default:

customers = customers.OrderBy(c => c.CustomerName);

break;

}

ViewBag.customers = customers.ToList();

return View();

}

[HttpPost]

public IActionResult AddCustomer(Customers customer)

{

\_context.Customers.Add(customer);

\_context.SaveChanges();

return RedirectToAction("Customer");

}

}

}

OrderController.cs

using System;

using System.Linq;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Http;

using AdvertisingAgency.Models;

using Microsoft.Extensions.Caching.Memory;

using AdvertisingAgency.Filtres;

using Newtonsoft.Json;

namespace AdvertisingAgency.Controllers

{

[SetToSession]

public class OrderController : Controller

{

public enum SortState

{

DateOrderAsc, // по дате договора по возрастанию

DateOrderDesc, // по дате договра по убыванию

PeriodAsc, // по периоду по возрастанию

PeriodDesc, // по периоду по убыванию

LocationAsc, // по локации по возрастанию

LocationDesc, // по локации по убыванию

PriceAsc, // по цене по убыванию

PriceDesc // по цене по убыванию

}

readonly IMemoryCache \_memoryCache;

readonly adAgencyContext \_context;

public OrderController(IMemoryCache memoryCache, adAgencyContext context)

{

\_memoryCache = memoryCache;

\_context = context;

}

[ResponseCache(CacheProfileName = "Caching")]

public IActionResult Order(string DateOrder, SortState sortOrder = SortState.DateOrderAsc )

{

Orders ordersFromMemoryCashe = (Orders)\_memoryCache.Get(Request.Path.Value.ToLower());

ViewBag.ordFromMemory = ordersFromMemoryCashe;

if (Request.Cookies["orderSession"] != null)

{

Orders orderSession = JsonConvert.DeserializeObject<Orders>(Request.Cookies["orderSession"].ToString());

ViewBag.orderSession = orderSession;

}

if (HttpContext.Session.Get("orderSortSession") != null)

{

sortOrder = JsonConvert.DeserializeObject<SortState>(HttpContext.Session.GetString("orderSortSession"));

}

IQueryable<Orders> orders = \_context.Orders;

DateTime dateorder = Convert.ToDateTime(DateOrder);

if (DateOrder != null)

{

orders = orders.Where(o => o.DateOrder.Date.Equals(dateorder));

}

ViewBag.DateOrderSort = sortOrder == SortState.DateOrderAsc ? SortState.DateOrderDesc : SortState.DateOrderAsc;

ViewBag.OrderPeriodSort = sortOrder == SortState.PeriodAsc ? SortState.PeriodDesc : SortState.PeriodAsc;

ViewBag.OrderLocationSort = sortOrder == SortState.LocationAsc ? SortState.LocationDesc : SortState.LocationAsc;

ViewBag.OrderCostSort = sortOrder == SortState.PriceAsc ? SortState.PriceDesc : SortState.PriceAsc;

switch (sortOrder)

{

case SortState.DateOrderAsc:

orders = orders.OrderBy(o => o.DateOrder);

break;

case SortState.DateOrderDesc:

orders = orders.OrderByDescending(o => o.DateOrder);

break;

case SortState.PeriodAsc:

orders = orders.OrderBy(o => o.DateBegin);

break;

case SortState.PeriodDesc:

orders = orders.OrderByDescending(o => o.DateEnd);

break;

case SortState.LocationAsc:

orders = orders.OrderBy(o => o.Location);

break;

case SortState.LocationDesc:

orders = orders.OrderByDescending(o => o.Location);

break;

case SortState.PriceAsc:

orders = orders.OrderBy(o => o.OrderCost);

break;

case SortState.PriceDesc:

orders = orders.OrderByDescending(o => o.OrderCost);

break;

default:

orders = orders.OrderBy(o => o.DateOrder);

break;

}

ViewBag.orders = orders.ToList();

return View();

}

[HttpPost]

public IActionResult AddOrder(Orders order)

{

\_context.Orders.Add(order);

\_context.SaveChanges();

return RedirectToAction("Order");

}

}}

**Исходный класс фильтра SetToSession.cs**:

using System;

using Microsoft.AspNetCore.Mvc.Filters;

using System.Text;

using Newtonsoft.Json;

namespace AdvertisingAgency.Filtres

{

public class SetToSession: Attribute, IActionFilter

{

public void OnActionExecuted(ActionExecutedContext context)

{

}

public void OnActionExecuting(ActionExecutingContext context)

{

if (context.ActionArguments.Count > 0)

{

string keyStr = "";

foreach (var item in context.ActionArguments.Keys)

{

keyStr += item;

}

string str = "";

foreach (var item in context.ActionArguments.Values)

{

str += JsonConvert.SerializeObject(item);

}

switch (keyStr)

{

case "advedirsments": context.HttpContext.Session.Set("adSession", Encoding.UTF8.GetBytes(str)); break;

case "customers": context.HttpContext.Session.Set("customerSession", Encoding.UTF8.GetBytes(str)); break;

case "orders": context.HttpContext.Session.Set("orderSession", Encoding.UTF8.GetBytes(str)); break;

case "sortType": context.HttpContext.Session.Set("sortTypeSession", Encoding.UTF8.GetBytes(str)); break;

case "sortCustomer": context.HttpContext.Session.Set("customerSortSession", Encoding.UTF8.GetBytes(str)); break;

case "sortOrder": context.HttpContext.Session.Set("orderSortSession", Encoding.UTF8.GetBytes(str)); break;

}

}

}

}

}

**Исходный класс фильтра LogFlter.cs**:

using Microsoft.AspNetCore.Mvc.Filters;

using Microsoft.Extensions.Logging;

using System;

using System.IO;

namespace AdvertisingAgency.Filtres

{

public class LogFilter : Attribute, IResourceFilter

{

string fileName;

ILogger \_logger;

public LogFilter(ILoggerFactory loggerFactory)

{

fileName = "logger";

\_logger = loggerFactory.CreateLogger("TimingResourceFilter");

}

public void OnResourceExecuted(ResourceExecutedContext context)

{

\_logger.LogInformation($"Path - {context.HttpContext.Request.Path}");

\_logger.LogInformation($"OnResourceExecuted - {DateTime.Now}");

File.AppendAllText(fileName, $"Path - {context.HttpContext.Request.Path}\n");

File.AppendAllText(fileName, $"OnResourceExecuted - {DateTime.Now}\n=================\n");

}

public void OnResourceExecuting(ResourceExecutingContext context)

{

\_logger.LogInformation($"OnResourceExecuting - {DateTime.Now}");

File.AppendAllText(fileName, $"OnResourceExecuting - {DateTime.Now}\n");

}

}

}

**Исходный класс фильтра ExceptionFilter.cs**:

using System;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace AdvertisingAgency.Filtres

{

public class ExceptionFilter: Attribute, IExceptionFilter

{

public void OnException(ExceptionContext context)

{

string actionName = context.ActionDescriptor.DisplayName;

string exceptionStack = context.Exception.StackTrace;

string exceptionMessage = context.Exception.Message;

context.Result = new ContentResult

{

Content = $"В методе {actionName} возникло исключение: \n {exceptionMessage} \n {exceptionStack}"

};

context.ExceptionHandled = true;

}

}

}

**Класс моделей данных**:

**Advedirsmnets.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace AdvertisingAgency.Models

{

public class Advedirsments

{

public Advedirsments() { }

public Advedirsments(string AdType, string AdDescription)

{

this.AdType = AdType;

this.AdDescription = AdDescription;

}

public int AdvedirsmentsId { get; set; }

public string AdType { get; set; }

public string AdDescription { get; set; }

public virtual ICollection<Orders> Orders { get; set; }

}

}

**Customers.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace AdvertisingAgency.Models

{

public class Customers

{

public Customers() { }

public Customers(string CustomerName, string CustomerAddress, string CustomerTelephone)

{

this.CustomerName = CustomerName;

this.CustomerAddress = CustomerAddress;

this.CustomerTelephone = CustomerTelephone;

}

public int CustomersId { get; set; }

public string CustomerName { get; set; }

public string CustomerAddress { get; set; }

public string CustomerTelephone { get; set; }

public virtual ICollection<Orders> Orders { get; set; }

}

}

**Orders.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace AdvertisingAgency.Models

{

public class Orders

{

public Orders() { }

public Orders(DateTime DateOrder, DateTime DateBegin, DateTime DateEnd, string Location, int OrderCost)

{

this.DateOrder = DateOrder;

this.DateBegin = DateBegin;

this.DateEnd = DateEnd;

this.Location = Location;

this.OrderCost = OrderCost;

}

public int OrdersId { get; set; }

public DateTime DateOrder { get; set; }

public DateTime DateBegin { get; set; }

public DateTime DateEnd { get; set; }

public string Location { get; set; }

public int OrderCost { get; set; }

public bool PaymentStatus { get; set; }

public int? EmployeeId { get; set; }

public int? CustomersId { get; set; }

public int AdvedirsmentsId { get; set; }

public virtual Advedirsments Advedirsments { get; set; }

public virtual Customers Customers { get; set; }

}

}

**Класс DbCacheMiddleware.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Http;

using Microsoft.Extensions.Caching.Memory;

using AdvertisingAgency.Models;

namespace AdvertisingAgency

{

public class CasheLast

{

private readonly RequestDelegate \_next;

private readonly IMemoryCache \_memoryCache;

public CasheLast(RequestDelegate next, IMemoryCache memCache)

{

\_next = next;

\_memoryCache = memCache;

}

public async Task Invoke(HttpContext context, adAgencyContext dbContext)

{

string path = context.Request.Path.Value.ToLower();

object data = null;

if (path == "/")

data = dbContext.Advedirsments.Last();

if (path == "/home/index")

data = dbContext.Advedirsments.Last();

if (path == "/customers/customers")

data = dbContext.Customers.Last();

if (path == "/orders/orders")

data = dbContext.Orders.Last();

var dataTmp = data;

if (!\_memoryCache.TryGetValue(path, out dataTmp))

{

\_memoryCache.Set(path, data, new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromSeconds(272)));

}

await \_next.Invoke(context);

}

}

public static class DbCacheExtensions

{

public static IApplicationBuilder UseLastElementCache(this IApplicationBuilder builder)

{

return builder.UseMiddleware<CasheLast>();

}

}

}

**Исходные коды представлений**:

Index.cshtml

@model AdvertisingAgency.Models.Advedirsments

@{

ViewData["Title"] = "Advedirsments";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<img src="~/images/images.jpg" />

<p></p>

<h1>Виды рекламы</h1>

@using (Html.BeginForm("Index", "Home", FormMethod.Get))

{

<p>

<label>Тип Рекламы</label>

@Html.TextBox("AdType")

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

</p>

}

<table border="1">

<tr>

<th>

<**a** **asp-action**="Index" **asp-route-sortType**="@ViewBag.AdTypeSort">

Тип

</**a**>

</th>

<th>

<**a** **asp-action**="Index" **asp-route-sortType**="@ViewBag.DesriptionSort">

Описание

</**a**>

</th>

</tr>

@foreach (var item in ViewBag.advedirsments)

{

<tr>

<td>@item.AdType</td>

<td>@item.AdDescription</td>

</tr>

}

</table>

@using (Html.BeginForm("AddType", "Home", FormMethod.Post))

{

<p class="add"> Добавить </p>

<p>

<label>Тип</label>

@{

Advedirsments adSession = (Advedirsments)ViewBag.adSesion;

if (adSession != null)

{

@Html.TextBox("AdType", adSession.AdType);

}

else

{

@Html.TextBox("AdType");

}

}

</p>

<p>

<label>Описание</label>

@{

if (adSession != null)

{

@Html.TextBox("AdDescription", adSession.AdDescription);

}

else

{

@Html.TextBox("AdDescription");

}

}

</p>

<p>

<input type="submit" value="Send">

</p>

}

@{

if (ViewBag.adFromMemory != null)

{

AdvertisingAgency.Models.Advedirsments adFromMemoryCashe = (AdvertisingAgency.Models.Advedirsments)ViewBag.adFromMemory;

<div class="infoMemory">

<p>MemoryCache</p>

@Html.DisplayNameFor(x => x.AdType):&nbsp @adFromMemoryCashe.AdType.ToString();<br />

@Html.DisplayNameFor(x => x.AdDescription):&nbsp @adFromMemoryCashe.AdDescription.ToString();<br />

</div>

}

}

Customer.cshtml

@model AdvertisingAgency.Models.Customers

@{

ViewData["Title"] = "Customers";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<img src="~/images/images.jpg" />

<h1>Заказчики</h1>

@using (Html.BeginForm("Customer", "Customer", FormMethod.Get))

{

<p>

<label>Имя заказчика</label>

@Html.TextBox("CustomerName")

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

</p>

}

<table>

<tr>

<th>

<a asp-action="Customer" asp-route-sortCustomer="@ViewBag.CustomerNameSort">

Имя

</a>

</th>

<th>

<a asp-action="Customer" asp-route-sortCustomer="@ViewBag.CustomerAddressSort">

Адрес

</a>

</th>

<th>

<a asp-action="Customer" asp-route-sortCustomer="@ViewBag.CustomerTelephoneSort">

Телефон

</a>

</th>

</tr>

@foreach (var item in ViewBag.customers)

{

<tr>

<td>@item.CustomerName</td>

<td>@item.CustomerAddress</td>

<td>@item.CustomerTelephone</td>

</tr>

}

</table>

@{

AdvertisingAgency.Models.Customers customerSession =

(AdvertisingAgency.Models.Customers)ViewBag.customerSession;

}

@using (Html.BeginForm("AddCustomer", "Customer", FormMethod.Post))

{

<p class="add"> Добавить </p>

<p>

<label>Имя заказчика</label>

@{

if (customerSession != null)

{

@Html.TextBox("CustomerName", customerSession.CustomerName)

}

else

{

@Html.TextBox("CustomerName")

}

}

</p>

<p>

<label>Адрес</label>

@{

if (customerSession != null)

{

@Html.TextBox("CustomerAddress", customerSession.CustomerAddress)

}

else

{

@Html.TextBox("CustomerAddress")

}

}

</p>

<p>

<label>Телефон</label>

@{

if (customerSession != null)

{

@Html.TextBox("CustomerTelephone", customerSession.CustomerTelephone)

}

else

{

@Html.TextBox("CustomerTelephone")

}

}

</p>

<p>

<input type="submit" value="Send">

</p>

}

@{

if (@ViewData["customersFromMemoryCashe"] != null)

{

AdvertisingAgency.Models.Customers custFromMemoryCashe = (AdvertisingAgency.Models.Customers)ViewData["customersFromMemoryCashe"];

<div class="infoMemory">

<p>MemoryCache</p>

@Html.DisplayNameFor(x => x.CustomerName):&nbsp @custFromMemoryCashe.CustomerName.ToString();<br />

@Html.DisplayNameFor(x => x.CustomerAddress):&nbsp @custFromMemoryCashe.CustomerAddress.ToString();<br />

@Html.DisplayNameFor(x => x.CustomerTelephone):&nbsp @custFromMemoryCashe.CustomerTelephone.ToString();<br />

</div>

}

}

Order.cshtml

@model AdvertisingAgency.Models.Orders

@{

ViewData["Title"] = "Orders";

}

<img src="~/images/images.jpg" />

<h1>Заказы</h1>

@using (Html.BeginForm("Order", "Order", FormMethod.Get))

{

<p>

<label>Дата Договора</label>

@Html.TextBox("DateOrder")

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

</p>

}

<table>

<tr>

<th>

<a asp-action="Order" asp-route-sortOrder="@ViewBag.DateOrderSort">

Дата заключения договора

</a>

</th>

<th>

<a asp-action="Order" asp-route-sortOrder="@ViewBag.OrderPeriodSort">

Срок действия

</a>

</th>

<th>

<a asp-action="Order" asp-route-sortOrder="@ViewBag.OrderLocationSort">

Место расположения

</a>

</th>

<th>

<a asp-action="Order" asp-route-sortOrder="@ViewBag.OrderCostSort">

Цена услуги

</a>

</th>

</tr>

@foreach (var item in ViewBag.orders)

{

<tr>

<td>@item.DateOrder.ToShortDateString()</td>

<td>@item.DateBegin.ToShortDateString() - @item.DateEnd.ToShortDateString()</td>

<td>@item.Location</td>

<td>@item.OrderCost</td>

</tr>

}

</table>

@using (Html.BeginForm("AddOrder", "Order", FormMethod.Post))

{

<p class="add"> Добавить </p>

<p>

<label>Дата заключения договора</label>

<input type="date" name="DateOrder" />

</p>

<p>

<label>Начало</label>

<input type="date" name="DateBegin" />

</p>

<p>

<label>Конец</label>

<input type="date" name="DateEnd" />

</p>

<p>

<label>Место расположения</label>

@Html.EditorFor(order => order.Location)

</p>

<p>

<label>Цена услуги</label>

@Html.EditorFor(order => order.OrderCost)

</p>

<p>

<input type="submit" value="Send">

</p>

}

@{

if (ViewBag.ordFromMemory != null)

{

AdvertisingAgency.Models.Orders ordersFromMemoryCashe = (AdvertisingAgency.Models.Orders)ViewBag.ordFromMemory;

<div class="infoMemory">

<p>MemoryCache</p>

@Html.DisplayNameFor(x => x.DateOrder):&nbsp @ordersFromMemoryCashe.DateOrder.ToShortDateString();<br />

@Html.DisplayNameFor(x => x.DateBegin):&nbsp @ordersFromMemoryCashe.DateBegin.ToShortDateString();<br />

@Html.DisplayNameFor(x => x.DateBegin):&nbsp @ordersFromMemoryCashe.DateEnd.ToShortDateString();<br />

@Html.DisplayNameFor(x => x.Location):&nbsp @ordersFromMemoryCashe.Location.ToString();<br />

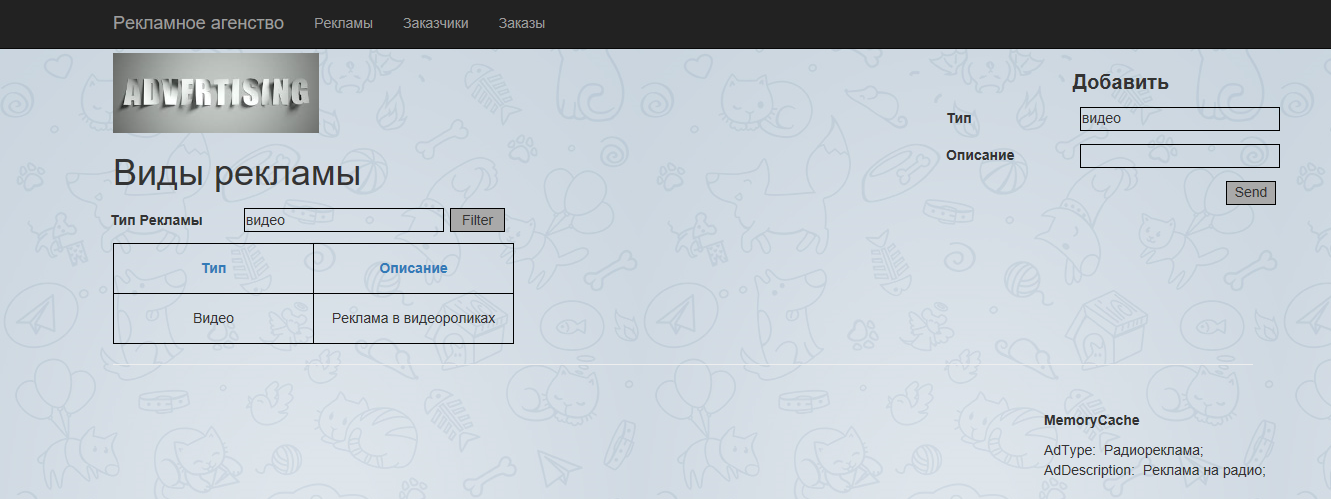
@Html.DisplayNameFor(x => x.OrderCost):&nbsp @ordersFromMemoryCashe.OrderCost.ToString();<br />

</div>

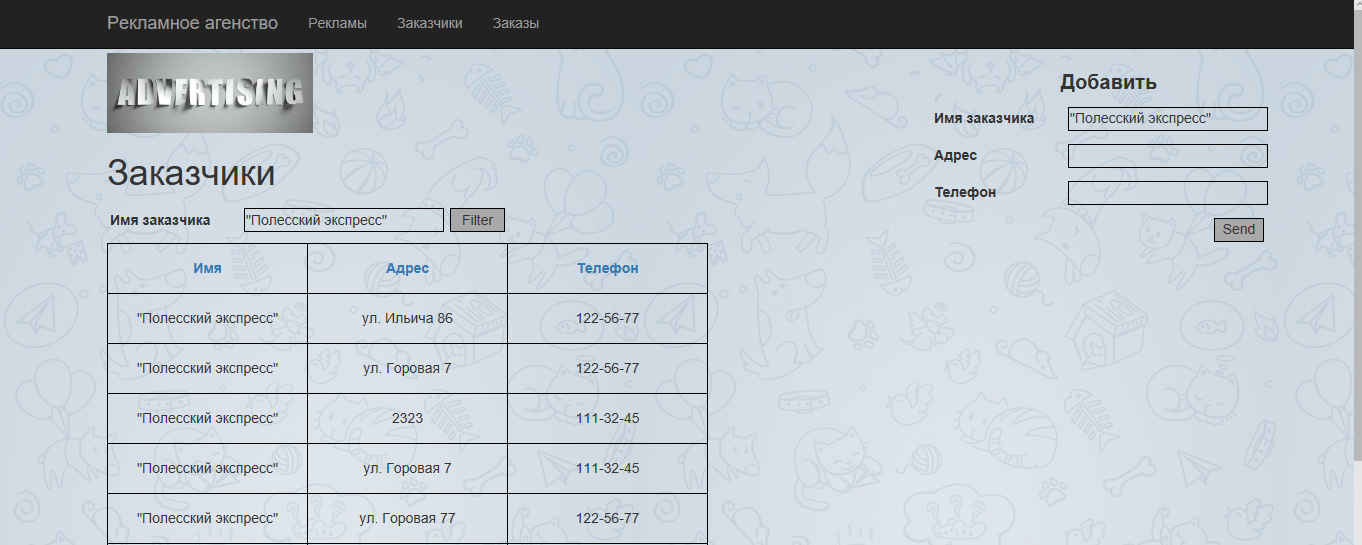
}

}

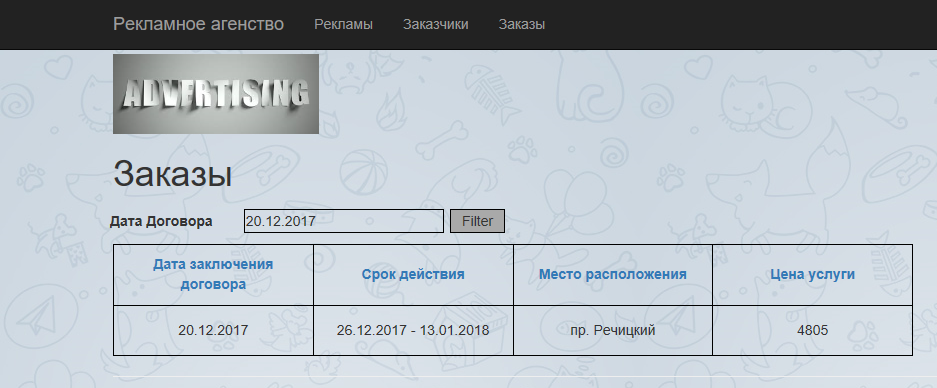
Результат фильтрации в таблице рекламы(Index.cshtml)



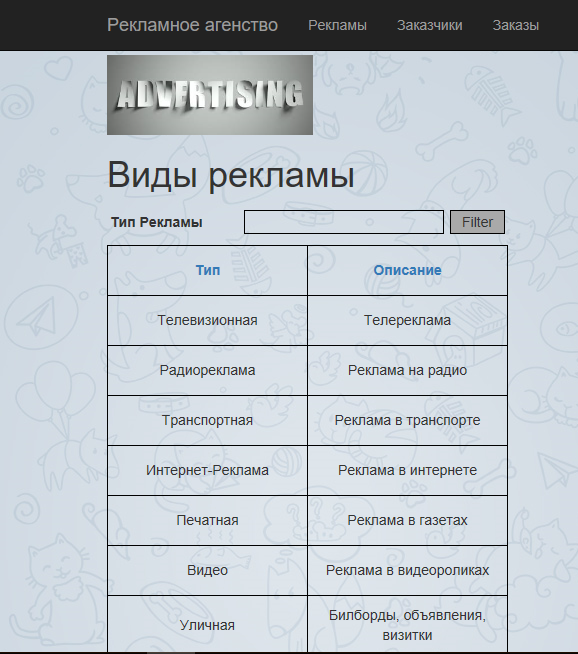
Результат фильтрации в таблице рекламы(Customer.cshtml)



Результат фильтрации в таблице рекламы(Order.cshtml)



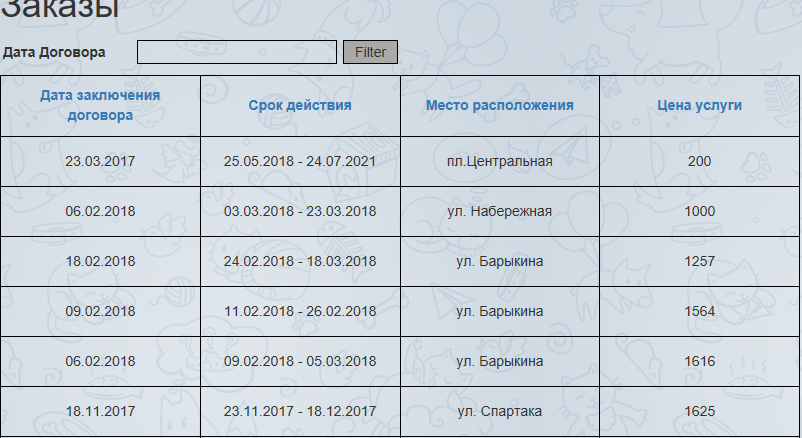
Демонстрация сортировки по убыванию в таблице Рекламы по полю «Описание» (Index.cshtml)



Результаты сортировки в таблице Заказчики по полю «Адрес» (Customer.cshtml)



Результаты сортировки в таблице Заказы по полю «Цена услуги» (Order.cshtml)



**Вывод**: в ходе выполнения данной лабораторной работы были получены навыки в работе с шаблонами проектирования Model-View-Controller в ASP.NET Core и его возможностями для обработки поступающих от HTTP-клиента запросов посредством контроллеров и фильтров, а также способами взаимодействия с реляционными источниками посредством Entity Framework Core и приемами визуализации данных. Задания лабораторной работы выполнены полностью