**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

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

ФАИС

Кафедра «Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

по дисциплине **«ИГИ»**

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

Выполнил: студент гр. ИП-32

Лось Д.И.

Принял: доцент Асенчик О.Д.

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

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

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

Гомель 2019

**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.

Копии экранов браузера, демонстрирующие ускорение обработки HTTP запроса при наличии кэширования различными методами:

Ссылка на GitHub: https://github.com/DmitriyLos/Lab4\_IGI

**Листинг программы:**

Исходный код класса Startup.cs:

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

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

services.AddDbContext<STOContext>();

services.AddMemoryCache();

services.AddDistributedMemoryCache();

services.AddSession();

services.Configure<CookiePolicyOptions>(options =>

{

options.CheckConsentNeeded = context => true;

options.MinimumSameSitePolicy = SameSiteMode.None;

});

services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);

services.AddMvc(options =>

{

options.CacheProfiles.Add("Caching",

new CacheProfile()

{

Location = ResponseCacheLocation.Client,

Duration = 252

});

options.CacheProfiles.Add("NoCaching",

new CacheProfile()

{

Location = ResponseCacheLocation.None,

NoStore = true

});

});

}

public void Configure(IApplicationBuilder app, IHostingEnvironment env,

ILoggerFactory loggerFactory)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

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

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseCookiePolicy();

app.UseOperatinCache("operation");

app.UseSession();

app.UseMvc(routes =>

{

routes.MapRoute(

name: "default",

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

});

loggerFactory.AddFile(Path.Combine(Directory.GetCurrentDirectory(), "logger.txt"));

var logger = loggerFactory.CreateLogger("FileLogger");

app.Run(async (context) =>

{

logger.LogInformation("Processing request {0}", context.Request.Path);

await context.Response.WriteAsync("Hello World!");

});

}

}

Исходный код класса контекста данных:

public class STOContext : DbContext

{

public DbSet<Car> Cars { get; set; }

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

public DbSet<Owner> Owners { get; set; }

public DbSet<Worker> Workers { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

var builder = new ConfigurationBuilder();

builder.SetBasePath(Directory.GetCurrentDirectory());

builder.AddJsonFile("appsettings.json");

var config = builder.Build();

string connectionString = config.GetConnectionString("DefaultConnection");

var options = optionsBuilder

.UseSqlServer(connectionString)

.Options;

}

}

Коды класса (классов) контроллеров.

[CatchExceptionFilter]

public class CarController : Controller

{

private int pageSize = 15;

private STOContext db;

private Car \_car = new Car

{

Model = "",

Colour = "",

StateNumber = "",

};

public CarController(STOContext CarContext)

{

db = CarContext;

}

[HttpGet]

public IActionResult Index(SortState sortOrder)

{

Car sessionCar = HttpContext.Session.GetObject<Car>("Car");

string sessionSortState = HttpContext.Session.GetString("SortStateCar");

int? page = HttpContext.Session.GetInt32("CarPage");

if (page == null)

{

page = 0;

HttpContext.Session.SetInt32("CarPage", 0);

}

if (sessionCar != null)

{

\_car = sessionCar;

}

if (sessionSortState != null)

if (sortOrder == SortState.No)

sortOrder = (SortState)Enum.Parse(typeof(SortState), sessionSortState);

ViewData["ModelSort"] = sortOrder == SortState.ModelDesc ? SortState.ModelAsc : SortState.ModelDesc;

ViewData["ColourSort"] = sortOrder == SortState.ColourDesc ? SortState.ColourAsc : SortState.ColourDesc;

HttpContext.Session.SetString("SortState", sortOrder.ToString());

IQueryable<Car> Cars = Sort(db.Cars, sortOrder,

\_car.Model, \_car.Colour,

\_car.StateNumber , (int)page);

CarsViewModel CarsView = new CarsViewModel

{

CarViewModel = \_car,

PageViewModel = Cars,

PageNumber = (int)page

};

return View(CarsView);

}

[HttpPost]

public IActionResult Index(Car car)

{

var sessionSortState = HttpContext.Session.GetString("SortStateCar");

SortState sortOrder = new SortState();

if (sessionSortState != null)

sortOrder = (SortState)Enum.Parse(typeof(SortState), sessionSortState);

int? page = HttpContext.Session.GetInt32("CarPage");

if (page == null)

{

page = 0;

HttpContext.Session.SetInt32("CarPage", 0);

}

IQueryable<Car> cars = Sort(db.Cars, sortOrder,

car.Model, car.Colour,

car.StateNumber, (int)page);

HttpContext.Session.SetObject("Car", car);

CarsViewModel carsView = new CarsViewModel

{

CarViewModel = car,

PageViewModel = cars,

PageNumber = (int)page

};

return View(carsView);

}

private IQueryable<Car> Sort(IQueryable<Car> cars,

SortState sortOrder, string model, string colour, string statenumber, int page)

{

switch (sortOrder)

{

case SortState.ModelAsc:

cars = cars.OrderBy(s => s.Model);

break;

case SortState.ModelDesc:

cars = cars.OrderByDescending(s => s.Model);

break;

case SortState.ColourAsc:

cars = cars.OrderBy(s => s.Colour);

break;

case SortState.ColourDesc:

cars = cars.OrderByDescending(s => s.Colour);

break;

}

cars = cars.Include(o => o.Model).Include(o => o.Colour)

.Include(o => o.StateNumber).Where(o => o.Model.Contains(model ?? ""))

.Where(o => o.Colour.Contains(colour ?? ""))

.Where(o => o.StateNumber.Contains(statenumber ?? ""))

.Skip(page \* pageSize).Take(pageSize);

return cars;

}

[HttpGet]

public IActionResult Add()

{

List<Car> cars = CarContext.GetPage(0, pageSize);

return View(cars);

}

[HttpPost]

public string Add(string model, string colour, string statenumber)

{

return "Автомобиль " + model + " с гос. номером " + statenumber + "цвет " + colour

+ " успешно зарегистрирован";

}

}

Исходные классы классов фильтров.

public class CatchExceptionFilterAttribute : Attribute, IExceptionFilter

{

public void OnException(ExceptionContext context)

{

string actionName = context.ActionDescriptor.DisplayName;

string exceptionMessage = context.Exception.Message;

context.Result = new ContentResult

{

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

};

context.ExceptionHandled = true;

}

}

public class DataFilter : Attribute, IActionFilter

{

private string type;

STOContext db = new STOContext();

public DataFilter(string type)

{

this.type = type;

}

public void OnActionExecuting(ActionExecutingContext context)

{

if (type == "addCar")

context.HttpContext.Session.SetString("getCars", JsonConvert.SerializeObject(db.Cars.ToList()));

if (type == "getCars")

{

context.HttpContext.Session.SetString("getCars", JsonConvert.SerializeObject(db.Cars.ToList()));

}

if (type == "addOwner")

context.HttpContext.Session.SetString("getOwners", JsonConvert.SerializeObject(db.Workers.ToList()));

if (type == "getOwners")

{

context.HttpContext.Session.SetString("getOwners", JsonConvert.SerializeObject(db.Workers.ToList()));

}

if(type== "addOwner")

{

context.HttpContext.Session.SetString("getOwners", JsonConvert.SerializeObject(db.Workers.Select(p => new Worker() { FioWorker = p.FioWorker, DateOfEmployment = p.DateOfEmployment, Salary = p.Salary }).ToList()));

context.HttpContext.Session.SetString("getCars", JsonConvert.SerializeObject(db.Cars.Select(p => new Car() { Model = p.Model, Colour = p.Colour, StateNumber = p.StateNumber }).ToList()));

db.Owners.ToList();

db.Cars.ToList();

context.HttpContext.Items.Add("getOwners", db.Owners.ToList());

}

if (type == "getOwners")

{

context.HttpContext.Session.SetString("getOwners", JsonConvert.SerializeObject(db.Owners.Select(p=>new Owner() { FioOwner=p.FioOwner,Phone=p.Phone,DriverLicense=p.DriverLicense}).ToList()));

context.HttpContext.Session.SetString("getCars", JsonConvert.SerializeObject(db.Cars.Select(p => new Car() { Model = p.Model, Colour = p.Colour, StateNumber = p.StateNumber }).ToList()));

db.Owners.ToList();

db.Cars.ToList();

context.HttpContext.Items.Add("getOwners", db.Owners.ToList());

}

}

public void OnActionExecuted(ActionExecutedContext context)

{

if (type == "addCar" && context.HttpContext.Session.Keys.Contains("allow"))

{

db.Cars.Add(JsonConvert.DeserializeObject<Car>(context.HttpContext.Session.GetString("addCar")));

db.SaveChanges();

}

if(type== "addOwner" && context.HttpContext.Session.Keys.Contains("allow"))

{

db.Owners.Add(JsonConvert.DeserializeObject<Owner>(context.HttpContext.Session.GetString("addOwner")));

db.SaveChanges();

}

if(type== "addOwner"&& context.HttpContext.Session.Keys.Contains("allow"))

{

Owner Owner = JsonConvert.DeserializeObject<Owner>(context.HttpContext.Session.GetString("Owner"));

string Worker = context.HttpContext.Session.GetString("Worker");

string Car = context.HttpContext.Session.GetString("Car");

db.Owners.Add(Owner);

db.SaveChanges();

}

}

}

public class LoggerFilter:Attribute,IResourceFilter

{

ILogger \_logger;

public LoggerFilter(ILoggerFactory loggerFactory)

{

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

}

public void OnResourceExecuted(ResourceExecutedContext context)

{

\_logger.LogInformation($" закончил работу {context.ActionDescriptor.DisplayName} - {DateTime.Now}");

}

public void OnResourceExecuting(ResourceExecutingContext context)

{

\_logger.LogInformation($" начал работу {context.ActionDescriptor.DisplayName} - {DateTime.Now}");

}

}

Коды других разработанных классов.

public class FileLogger : ILogger

{

private string filePath;

private object \_lock = new object();

public FileLogger(string path)

{

filePath = path;

}

public IDisposable BeginScope<TState>(TState state)

{

return null;

}

public bool IsEnabled(LogLevel logLevel)

{

return true;

}

public void Log<TState>(LogLevel logLevel, EventId eventId, TState state, Exception exception, Func<TState, Exception, string> formatter)

{

if (formatter != null)

{

lock (\_lock)

{

File.AppendAllText(filePath, formatter(state, exception) + Environment.NewLine);

}

}

}

}

Исходные коды представлений (Views):

@using Lab4.ViewModels

@model CarsViewModel

@{

ViewData["Title"] = "Index";

}

<div>

<div class="data-search">

@using (Html.BeginForm("Index", "Car", FormMethod.Post))

{

<fieldset>

Модель: @Html.TextBox("Model", Model.CarViewModel.Model)

Цвет: @Html.TextBox("Colour", Model.CarViewModel.Colour)

Номер: @Html.TextBox("StateNumber", Model.CarViewModel.StateNumber)

<input type="submit" value="Найти" />

</fieldset>

}

</div>

<div>

Автомобили

<table>

<tr>

<td>

<**a** **asp-controller**="Service" **asp-action**="Index" **asp-route-sortOrder**="@ViewBag.ColourSort">

Цвет

</**a**>

</td>

<td>

<**a** **asp-controller**="Service" **asp-action**="Index" **asp-route-sortOrder**="@ViewBag.ModelSort">

Модель

</**a**>

</td>

</tr>

@{

foreach (Car i in Model.PageViewModel)

{

<tr>

<td>@i.Model</td>

<td>@i.Colour</td>

<td>@i.StateNumber</td>

</tr>

}

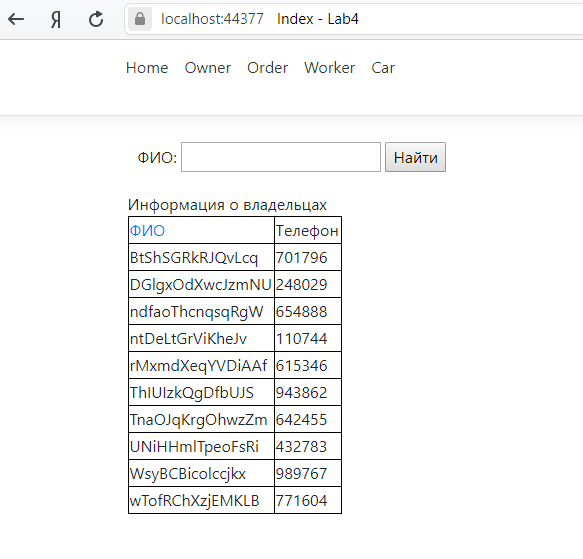
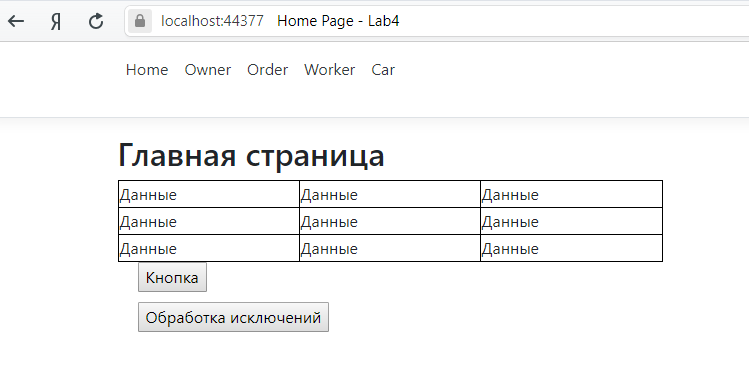
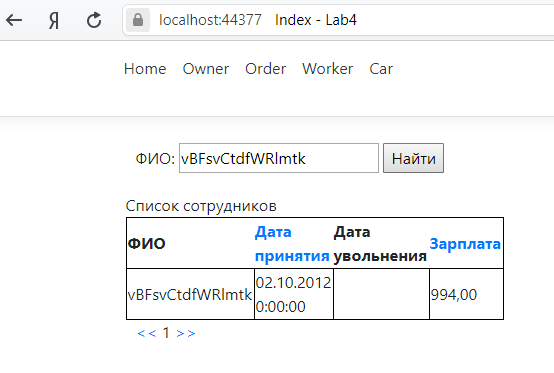
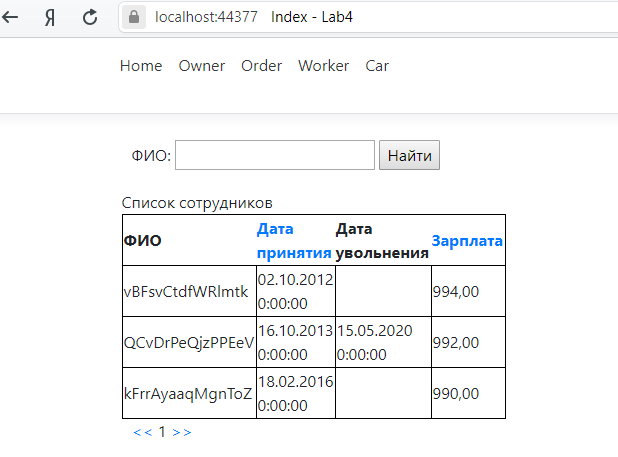
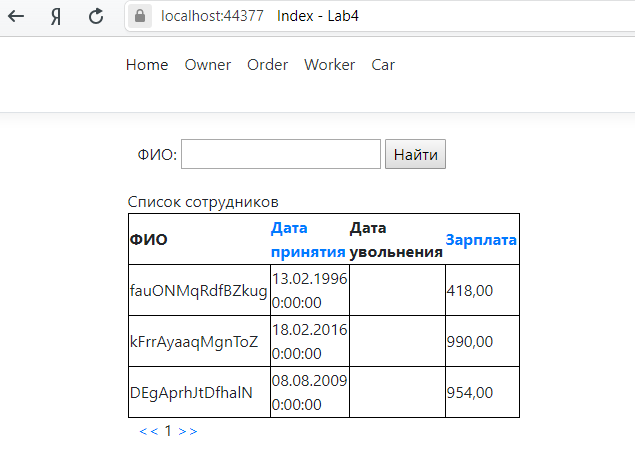
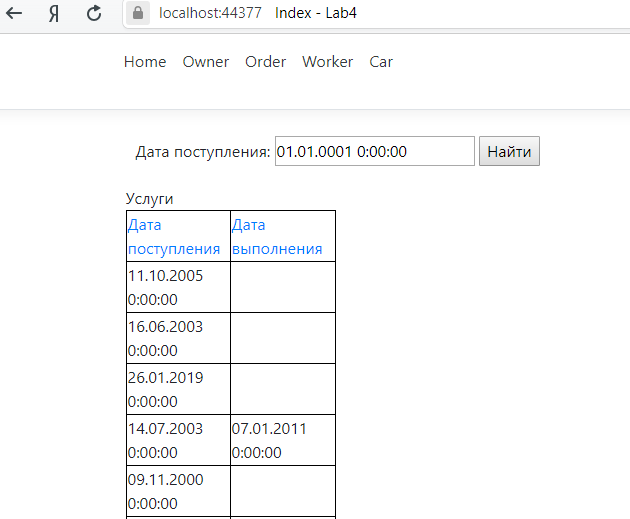
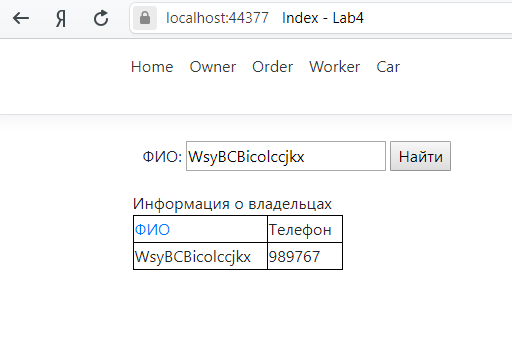
}

</table>

</div>

</div>

Копии экранов браузера, демонстрирующие отображение представлений (Views).



**Вывод**. Создали web приложение использующее фильтры и контроллеры для управления отображаемыми данными.