МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

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

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

дисциплина «Разработка приложений баз данных для информационных систем»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

«Обработка HTTP запросов средствами ASP.NET Core. Сохранение состояния. Кэширование»

Вариант №29

Выполнил:

студент группы ИТП-31, Зайцев А.В.

Принял:

доцент Асенчик О.Д.

Гомель 2020

**1. Цель работы:**

Ознакомиться c методами обработкой *HTTP* средствами *ASP.NET Core*, методами сохранения состояния приложения и повышение производительности приложений путем использования разных видов кэширования.

**2. Ход работы и результаты:**

Исходный код класса *Startup.cs* с разработанными компонентами *middleware*:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Http;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection.Extensions;

using System.Globalization;

using Lab3WebApp.Data;

using Lab3WebApp.Models;

using Lab3WebApp.Services;

using Lab3WebApp.Infrastructure;

namespace Lab3WebApp

{

public class Startup

{

public IConfiguration Configuration { get; }

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

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

// For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?LinkID=398940

public void ConfigureServices(IServiceCollection services)

{

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

services.AddDbContext<car\_sharingContext>(options => options.UseSqlServer(connectionString));

services.AddMemoryCache();

services.AddScoped<ICachedCarModelsService, CachedCarModelsService>();

services.AddScoped<ICachedEmployeesService, CachedEmployeesService>();

services.AddScoped<ICachedCarsService, CachedCarsService>();

services.AddDistributedMemoryCache();

services.AddSession();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env, car\_sharingContext db)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseSession();

app.Map("/info", Info);

app.Map("/carmodels", CarModels);

app.Map("/cars", Cars);

app.Map("/employees", Employees);

app.Map("/searchform", SearchForm);

app.Run(async (context) =>

{

ICachedCarModelsService cachedCarModels = context.RequestServices.GetService<ICachedCarModelsService>();

cachedCarModels.GetCarModels("carmodels20");

ICachedEmployeesService cachedEmployeesService = context.RequestServices.GetService<ICachedEmployeesService>();

cachedEmployeesService.GetEmployees("employees20");

ICachedCarsService cachedCarsService = context.RequestServices.GetService<ICachedCarsService>();

cachedCarsService.GetCars("cars20");

User user = context.Session.Get<User>("user") ?? new User();

string htmlString = "<html>" +

"<head>" +

"<title>Форма пользователя</title>" +

"<style>" +

"div { font-size: 24; }" +

"</style>" +

"</head>" +

"<meta charset='utf-8'/>" +

"<body>" +

"<div align='center'>" +

"<form action='/'>" +

"<div>Введите логин:</div>";

htmlString += $"<div><input type='text' name='loginStr' value=" + user.Login + "></div>";

htmlString += "<div>Введите пароль:</div>";

htmlString += $"<div><input type='text' name='passwordStr' value=" + user.Password + "></div>" +

"<div><input type='submit' value='Enter/Update'></div>" +

"</form>" +

"<div><a href='/carmodels'>Table 'CarModels'</a></div>" +

"<div><a href='/employees'>Table 'Employees'</a></div>" +

"<div><a href='/cars'>Table 'Cars'</a></div>" +

"<div><a href='/searchform'>Search Form</a></div>" +

"</div>" +

"</body>" +

"</html>";

string Login = context.Request.Query["loginStr"];

string Password = context.Request.Query["passwordStr"];

if (Login != null && Password != null)

{

user.Login = Login;

user.Password = Password;

context.Session.Set<User>("user", user);

}

await context.Response.WriteAsync(htmlString);

});

}

private static void Info(IApplicationBuilder app)

{

app.Run(async (context) =>

{

string httpString = "<html>" +

"<head>" +

"<title>Информация о клиенте</title>" +

"<style>" +

"div { font-size: 24; }" +

"</style>" +

"</head>" +

"<meta charset='utf-8'/>" +

"<body align='middle'>" +

"<div> Сервер: " + context.Request.Host + "</div>" +

"<div> Путь: " + context.Request.PathBase + "</div>" +

"<div> Протокол: " + context.Request.Protocol + "</div>" +

"<div><a href='/'>Главная</a></div>" +

"</body>" +

"</html>";

await context.Response.WriteAsync(httpString);

});

}

private static void CarModels(IApplicationBuilder app)

{

app.Run(async (context) =>

{

ICachedCarModelsService cachedCarModelsService = context.RequestServices.GetService<ICachedCarModelsService>();

IEnumerable<CarModel> carModels = cachedCarModelsService.GetCarModels("carmodels20");

string httpString = "<html>" +

"<head>" +

"<title>Таблица CarModels</title>" +

"<style>" +

"div { font-size: 24; }" +

"table { font-size: 20; }" +

"</style>" +

"</head>" +

"<meta charset='utf-8'/>" +

"<body>" +

"<div align='center'>Таблица 'CarModels'</div>" +

"<div align='center'>" +

"<table border=1>" +

"<tr>" +

"<td>Название модели</td>" +

"<td>Описание</td>" +

"</tr>";

foreach (CarModel carModel in carModels)

{

httpString += "<tr>";

httpString += $"<td>{carModel.Name}</td>";

httpString += $"<td>{carModel.Description}</td>";

httpString += "</tr>";

}

httpString += "</table>";

httpString += "<div align='center'><a href='/'>Главная</a></div>";

httpString += "</body>" +

"</html>";

await context.Response.WriteAsync(httpString);

});

}

private static void Cars(IApplicationBuilder app)

{

app.Run(async (context) =>

{

ICachedCarsService cachedCarsService = context.RequestServices.GetService<ICachedCarsService>();

IEnumerable<Car> cars = cachedCarsService.GetCars("cars20");

string httpString = "<html>" +

"<head>" +

"<title>Таблица Cars</title>" +

"<style>" +

"div { font-size: 24; }" +

"table { font-size: 14; }" +

"</style>" +

"</head>" +

"<meta charset='utf-8'/>" +

"<body>" +

"<div align='center'>Таблица 'Cars'</div>" +

"<div align='center'>" +

"<table border=1>" +

"<tr>" +

"<td>Название модели</td>" +

"<td>Описание</td>" +

"<td>Регистрационный номер</td>" +

"<td>VIN-код</td>" +

"<td>Номер двигателя</td>" +

"<td>Цена</td>" +

"<td>Аренда</td>" +

"<td>Год выпуска</td>" +

"<td>Характеристики</td>" +

"<td>Дата ТО</td>" +

"<td>Спец. метка</td>" +

"<td>Метка возврата</td>" +

"<td>Имя сотрудника</td>" +

"<td>Фамилия сотрудника</td>" +

"</tr>";

foreach (Car car in cars)

{

httpString += "<tr>";

httpString += $"<td>{car.CarModel.Name}</td>";

httpString += $"<td>{car.CarModel.Description}</td>";

httpString += $"<td>{car.RegNum}</td>";

httpString += $"<td>{car.Vincode}</td>";

httpString += $"<td>{car.EngineNum}</td>";

httpString += $"<td>{car.Price}</td>";

httpString += $"<td>{car.RentalPrice}</td>";

httpString += $"<td>{car.IssueDate}</td>";

httpString += $"<td>{car.Specs}</td>";

httpString += $"<td>{car.TechnicalMaintenanceDate}</td>";

httpString += $"<td>{car.SpecMark}</td>";

httpString += $"<td>{car.ReturnMark}</td>";

httpString += $"<td>{car.Employee?.Name}</td>";

httpString += $"<td>{car.Employee?.Surname}</td>";

httpString += "</tr>";

}

httpString += "</table>";

httpString += "<div align='center'><a href='/'>Главная</a></div>";

httpString += "</body>" +

"</html>";

await context.Response.WriteAsync(httpString);

});

}

private static void Employees(IApplicationBuilder app)

{

app.Run(async (context) =>

{

ICachedEmployeesService cachedEmployeesService = context.RequestServices.GetService<ICachedEmployeesService>();

IEnumerable<Employee> employees = cachedEmployeesService.GetEmployees("employees20");

string httpString = "<html>" +

"<head>" +

"<title>Таблица Employees</title>" +

"<style>" +

"div { font-size: 24; }" +

"table { font-size: 20; }" +

"</style>" +

"</head>" +

"<meta charset='utf-8'/>" +

"<body>" +

"<div align='center'>Таблица 'Employees'</div>" +

"<div align='center'>" +

"<table border=1>" +

"<tr>" +

"<td>Должность</td>" +

"<td>Имя</td>" +

"<td>Фамилия</td>" +

"<td>Отчество</td>" +

"<td>Дата трудоустройства</td>" +

"</tr>";

foreach (Employee employee in employees)

{

httpString += "<tr>";

httpString += $"<td>{employee.Post}</td>";

httpString += $"<td>{employee.Name}</td>";

httpString += $"<td>{employee.Surname}</td>";

httpString += $"<td>{employee.Patronymic}</td>";

httpString += $"<td>{employee.EmploymentDate}</td>";

httpString += "</tr>";

}

httpString += "</table>";

httpString += "<div align='center'><a href='/'>Главная</a></div>";

httpString += "</body>" +

"</html>";

await context.Response.WriteAsync(httpString);

});

}

private static void SearchForm(IApplicationBuilder app)

{

app.Run(async (context) =>

{

ICachedCarModelsService cachedCarModelsService = context.RequestServices.GetService<ICachedCarModelsService>();

IEnumerable<CarModel> carModels = cachedCarModelsService.GetCarModels("carmodels20");

ICachedCarsService cachedCarsService = context.RequestServices.GetService<ICachedCarsService>();

IEnumerable<Car> cars = cachedCarsService.GetCars("cars20");

ICachedEmployeesService cachedEmployeesService = context.RequestServices.GetService<ICachedEmployeesService>();

IEnumerable<Employee> employees = cachedEmployeesService.GetEmployees("employees20");

string httpString = "<html>" +

"<head>" +

"<title>Форма поиска</title>" +

"<style>" +

"div { font-size: 24; }" +

"table { font-size: 20; }" +

"select {font-size: 20; width=20%; }" +

"input {font-size: 22; width=20%; }" +

"</style>" +

"</head>" +

"<meta charset='utf-8'/>" +

"<body>" +

"<div align='middle' text-align='left'>" +

"<form action='/searchform'>" +

"<div width=20%>Выберете таблицу</div>" +

"<select name='tableName'>" +

"<option>Choose table</option>" +

"<option>CarModels</option>" +

"<option>Employees</option>" +

"<option>Cars</option>" +

"</select>" +

"<input type = 'submit' value = 'Select'>";

string selectedText = context.Request.Cookies["table"] ?? context.Request.Query["tableName"];

if (context.Request.Cookies["table"] == "Choose table")

context.Response.Cookies.Delete("table");

if (selectedText != null)

{

if (selectedText != "Choose table" && selectedText != context.Request.Cookies["tableName"])

{

string querySttring = context.Request.Query["tableName"];

if (querySttring != null && querySttring != "Choose table")

{

context.Response.Cookies.Append("table", querySttring);

selectedText = querySttring;

}

}

switch (selectedText)

{

case "CarModels":

httpString += "<ul>";

foreach (CarModel carModel in carModels)

{

httpString += $"<li>{carModel.Name}</li>";

}

httpString += "</ul>";

break;

case "Employees":

httpString += "<ul>";

foreach (Employee employee in employees)

{

httpString += $"<li>{employee.Name}</li>";

}

httpString += "</ul>";

break;

case "Cars":

httpString += "<ul>";

foreach (Car car in cars)

{

httpString += $"<li>{car.Vincode}, {car.CarModel?.Name}.</li>";

}

httpString += "</ul>";

break;

}

httpString += "<div>" +

"<input type='text' name='entity'>" +

"<input type='submit' value='Input'>" +

"</div>";

string entityInput;

if ((entityInput = context.Request.Query["entity"]) != null && entityInput != "")

{

switch (selectedText)

{

case "CarModels":

CarModel carModel = carModels.FirstOrDefault(c => c.Name == entityInput);

if (carModel != null)

{

httpString += "<div>" +

"<p>" +

$"Название модели: {carModel.Name}, Описание: {carModel.Description}." +

"</p>" +

"</div>";

}

break;

case "Employees":

Employee employee = employees.FirstOrDefault(e => e.Name == entityInput);

if (employee != null)

{

httpString += "<div>" +

"<p>" +

$"Имя: {employee.Name}, Фамилия {employee.Surname}, Отчество: {employee.Patronymic}, " +

$"Должность: {employee.Post}, дата трудоустройства: {employee.EmploymentDate}." +

"</p>" +

"</div>";

}

break;

case "Cars":

int id;

if (int.TryParse(entityInput, out id))

{

Car car = cars.FirstOrDefault(c => c.CarId == id);

httpString += "<div><p>";

httpString += $"Название модели : {car.CarModel?.Name},";

httpString += $"Описание: {car.CarModel?.Description},";

httpString += $"Регистрационный номер: {car.RegNum},";

httpString += $"VIN-код: {car.Vincode},";

httpString += $"Номер двигателя: {car.EngineNum},";

httpString += $"Цена: {car.Price},";

httpString += $"Аренда: {car.RentalPrice},";

httpString += $"Год выпуска: {car.IssueDate},";

httpString += $"Характеристики: {car.Specs},";

httpString += $"Дата ТО: {car.TechnicalMaintenanceDate},";

httpString += $"Спец. метка: {car.SpecMark},";

httpString += $"Метка возврата: {car.ReturnMark},";

httpString += $"Имя сотрудника: {car.Employee?.Name},";

httpString += $"Фамилия сотрудника: {car.Employee?.Surname},";

httpString += "</p>" +

"</div>";

}

break;

}

}

}

httpString += "</form>" +

"<div><a href='/searchform'>Очистить</a></div>" +

"<div><a href='/'>Главная</a></div>" +

"</div>" +

"</body>" +

"</html>";

await context.Response.WriteAsync(httpString);

});

}

}

}

Исходные классы классов (методов) для создания объектов, сохраняемых в куки и *Session*, и задание значений элементов формы:

Класс *User.cs*.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace Lab3WebApp.Models

{

public class User

{

public string Login { get; set; }

public string Password { get; set; }

}

}

Класс *Car.cs*.

using Microsoft.EntityFrameworkCore.Migrations.Operations;

using System;

using System.Collections.Generic;

namespace Lab3WebApp.Models

{

public partial class Car

{

public Car()

{

}

public int CarId { get; set; }

public int? CarModelId { get; set; }

public int? RegNum { get; set; }

public string Vincode { get; set; }

public int? EngineNum { get; set; }

public decimal? Price { get; set; }

public decimal? RentalPrice { get; set; }

public DateTime? IssueDate { get; set; }

public string Specs { get; set; }

public DateTime? TechnicalMaintenanceDate { get; set; }

public bool? SpecMark { get; set; }

public bool? ReturnMark { get; set; }

public int? EmployeeId { get; set; }

public virtual CarModel CarModel { get; set; }

public virtual Employee Employee { get; set; }

public override string ToString()

{

return $"VIN-код: {Vincode}, Цена: {Price.ToString()},Стоимость аренды {RentalPrice.ToString()}";

}

}

}

Класс *CarModel.cs*.

using System;

using System.Collections.Generic;

namespace Lab3WebApp.Models

{

public partial class CarModel

{

public CarModel()

{

Cars = new HashSet<Car>();

}

public int CarModelId { get; set; }

public string Name { get; set; }

public string Description { get; set; }

public virtual ICollection<Car> Cars { get; set; }

public override string ToString()

{

return $"Название: {Name}, Описание: {Description}";

}

}

}

Класс *Employee.cs*.

using System;

using System.Collections.Generic;

namespace Lab3WebApp.Models

{

public partial class Employee

{

public Employee()

{

Cars = new HashSet<Car>();

}

public int EmployeeId { get; set; }

public string Post { get; set; }

public string Name { get; set; }

public string Surname { get; set; }

public string Patronymic { get; set; }

public DateTime? EmploymentDate { get; set; }

public virtual ICollection<Car> Cars { get; set; }

public override string ToString()

{

return $"Должность: {Post}, Имя: {Name}, Фамилия: {Surname}, Отчество: {Patronymic}, Дата трудоустройства: {EmploymentDate.ToString()}";

}

}

}

Класс *SessionExtensions.cs*.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using System.Text.Json;

namespace Lab3WebApp.Infrastructure

{

public static class SessionExtensions

{

public static void Set<T>(this ISession session, string key, T value)

{

session.SetString(key, JsonSerializer.Serialize<T>(value));

}

public static T Get<T>(this ISession session, string key)

{

var value = session.GetString(key);

return value == null ? default(T) : JsonSerializer.Deserialize<T>(value);

}

}

}

Класс *CachedCarModelsService.cs*.

using Lab3WebApp.Data;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Caching.Memory;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Lab3WebApp.Models;

namespace Lab3WebApp.Services

{

public class CachedCarModelsService : ICachedCarModelsService

{

private car\_sharingContext db;

private IMemoryCache cache;

public CachedCarModelsService(car\_sharingContext context, IMemoryCache memoryCache)

{

db = context;

cache = memoryCache;

}

public void AddCarModels(string cacheKey, int rowsCount = 20)

{

IEnumerable<CarModel> carModels = null;

carModels = db.CarModels.Take(rowsCount).ToList();

if (carModels != null)

{

cache.Set(cacheKey, carModels, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(298)

});

}

}

public IEnumerable<CarModel> GetCarModels(int rowsCount = 20)

{

return db.CarModels.Take(rowsCount).ToList();

}

public IEnumerable<CarModel> GetCarModels(string cacheKey, int rowCount = 20)

{

IEnumerable<CarModel> carModels = null;

if (!cache.TryGetValue(cacheKey, out carModels))

{

carModels = db.CarModels.Take(rowCount).ToList();

if (carModels != null)

{

cache.Set(cacheKey, carModels, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(298)

});

}

}

return carModels;

}

}

}

Класс *CachedCarsService.cs*.

using Lab3WebApp.Data;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Caching.Memory;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Lab3WebApp.Models;

namespace Lab3WebApp.Services

{

public class CachedCarsService : ICachedCarsService

{

private car\_sharingContext db;

private IMemoryCache cache;

public CachedCarsService(car\_sharingContext context, IMemoryCache memoryCache)

{

db = context;

cache = memoryCache;

}

public void AddCars(string cacheKey, int rowsCount = 20)

{

IEnumerable<Car> cars = null;

cars = db.Cars.Include(c => c.CarModel).Include(c => c.Employee).Take(rowsCount).ToList();

if (cars != null)

{

cache.Set(cacheKey, cars, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(298)

});

}

}

public IEnumerable<Car> GetCars(int rowsCount = 20)

{

return db.Cars.Include(c => c.CarModel).Include(c => c.Employee).Take(rowsCount).ToList();

}

public IEnumerable<Car> GetCars(string cacheKey, int rowsCount = 20)

{

IEnumerable<Car> cars = null;

if (!cache.TryGetValue(cacheKey, out cars))

{

cars = db.Cars.Include(c => c.CarModel).Include(c => c.Employee).Take(rowsCount).ToList();

if (cars != null)

{

cache.Set(cacheKey, cars, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(298)

});

}

}

return cars;

}

}

}

Класс *CachedEmployeesService.cs*.

using Lab3WebApp.Data;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Caching.Memory;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Lab3WebApp.Models;

namespace Lab3WebApp.Services

{

public class CachedEmployeesService : ICachedEmployeesService

{

private car\_sharingContext db;

private IMemoryCache cache;

public CachedEmployeesService(car\_sharingContext context, IMemoryCache memoryCache)

{

db = context;

cache = memoryCache;

}

public void AddEmployees(string cacheKey, int rowsCount = 20)

{

IEnumerable<Employee> employees = null;

employees = db.Employees.Take(rowsCount).ToList();

if (employees != null)

{

cache.Set(cacheKey, employees, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(298)

});

}

}

public IEnumerable<Employee> GetEmployees(int rowsCount = 20)

{

return db.Employees.Take(rowsCount).ToList();

}

public IEnumerable<Employee> GetEmployees(string cacheKey, int rowsCount = 20)

{

IEnumerable<Employee> employees = null;

if (!cache.TryGetValue(cacheKey, out employees))

{

employees = db.Employees.Take(rowsCount).ToList();

if (employees != null)

{

cache.Set(cacheKey, employees, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(298)

});

}

}

return employees;

}

}

}

Класс *ICachedCarModelsService.cs*.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Lab3WebApp.Models;

namespace Lab3WebApp.Services

{

public interface ICachedCarModelsService

{

public IEnumerable<CarModel> GetCarModels(int rowCount = 20);

public void AddCarModels(string cacheKey, int rowCount = 20);

public IEnumerable<CarModel> GetCarModels(string cacheKey, int rowCount = 20);

}

}

Класс *ICachedCarsService.cs*.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Lab3WebApp.Models;

namespace Lab3WebApp.Services

{

public interface ICachedCarsService

{

public IEnumerable<Car> GetCars(int rowCount = 20);

public void AddCars(string cacheKey, int rowCount = 20);

public IEnumerable<Car> GetCars(string cacheKey, int rowCount = 20);

}

}

Класс *ICachedEmployeesService.cs*.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Lab3WebApp.Models;

namespace Lab3WebApp.Services

{

public interface ICachedEmployeesService

{

public IEnumerable<Employee> GetEmployees(int rowCount = 20);

public void AddEmployees(string cacheKey, int rowCount = 20);

public IEnumerable<Employee> GetEmployees(string cacheKey, int rowCount = 20);

}

}

Копии экранов браузера, демонстрирующие результаты выполнение запросов с различными URL:

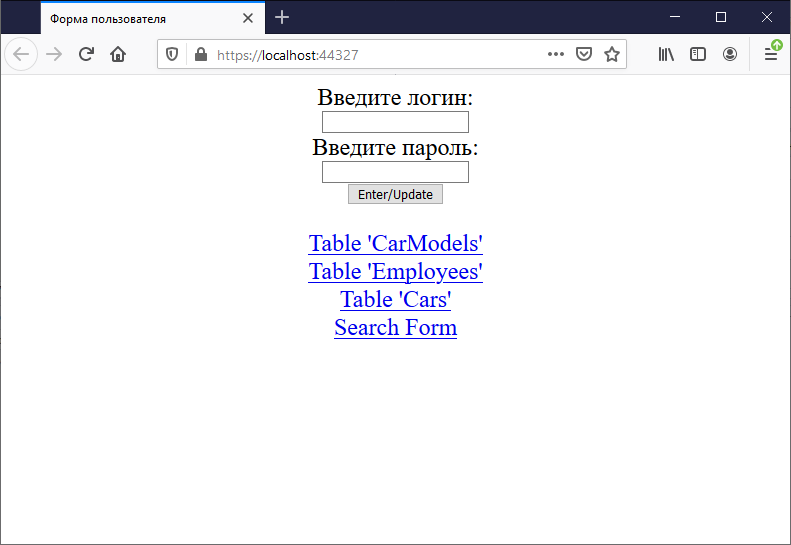


Рисунок 1 – Главная страница с реализацией *Sessions*

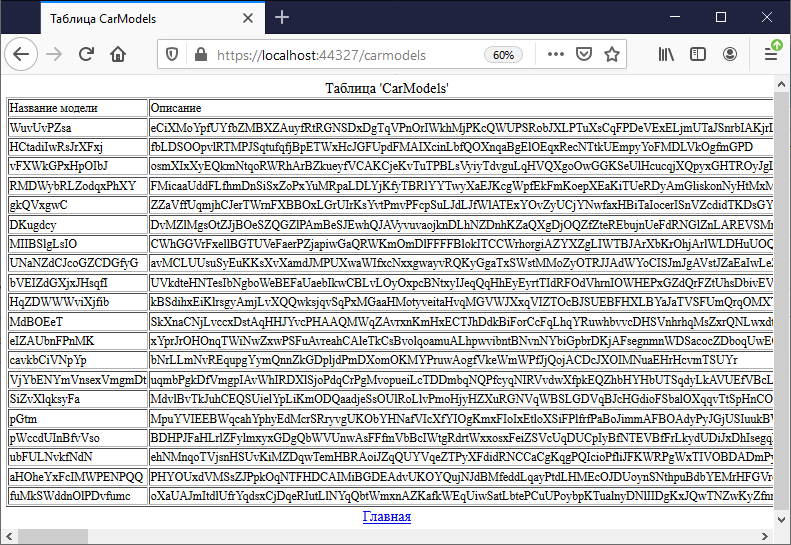


Рисунок 2 – Таблица «*CarModels*»

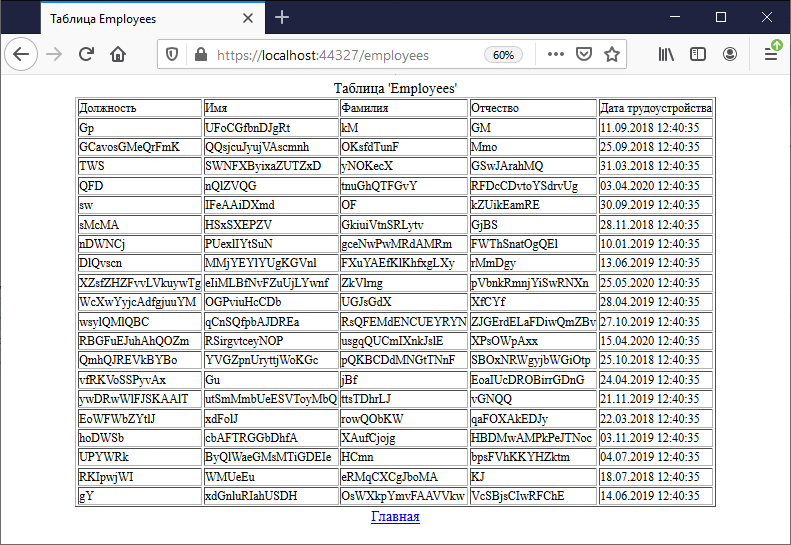


Рисунок 3 – Таблица «*Employees*»

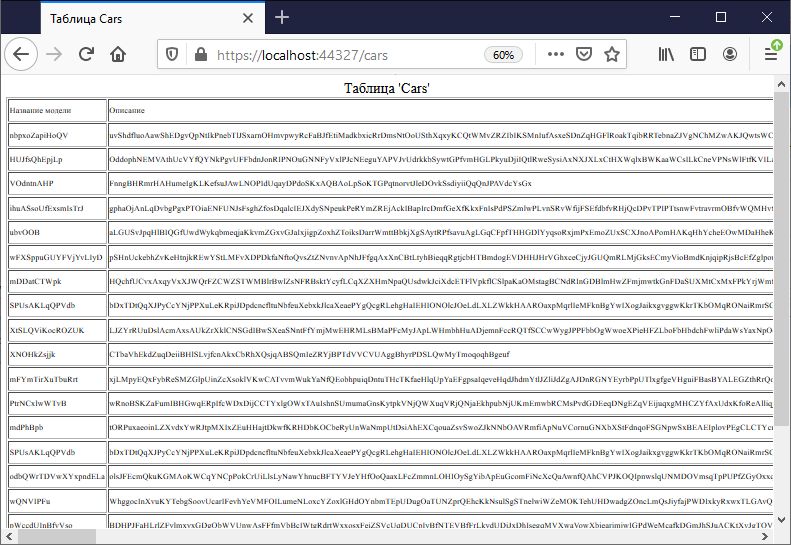


Рисунок 4 – Таблица «*Cars*»

На рисунках 4-6 представлена форма с реализацией *coockies*.

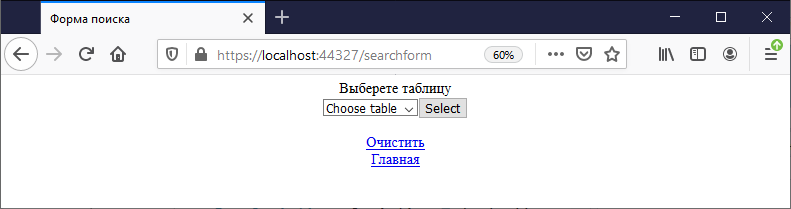


Рисунок 4 – Форма поиска

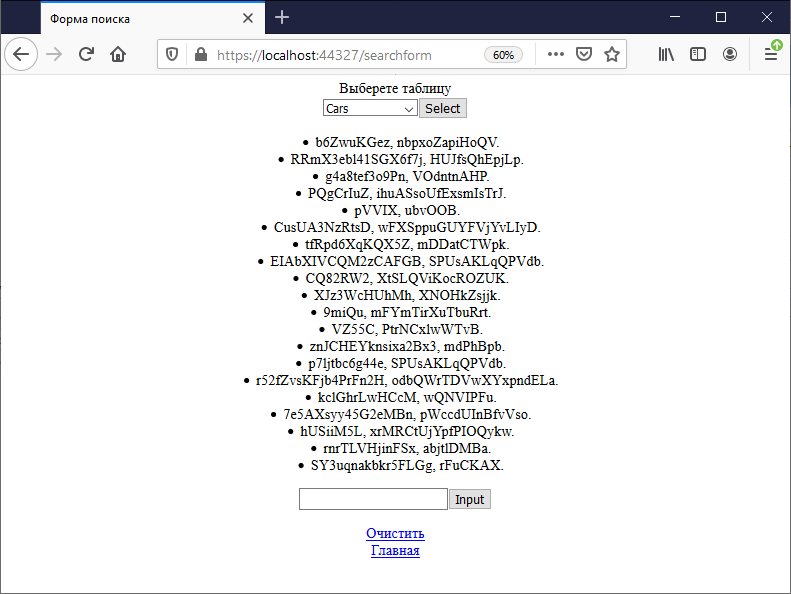


Рисунок 5 – Форма поиска после выбора таблицы

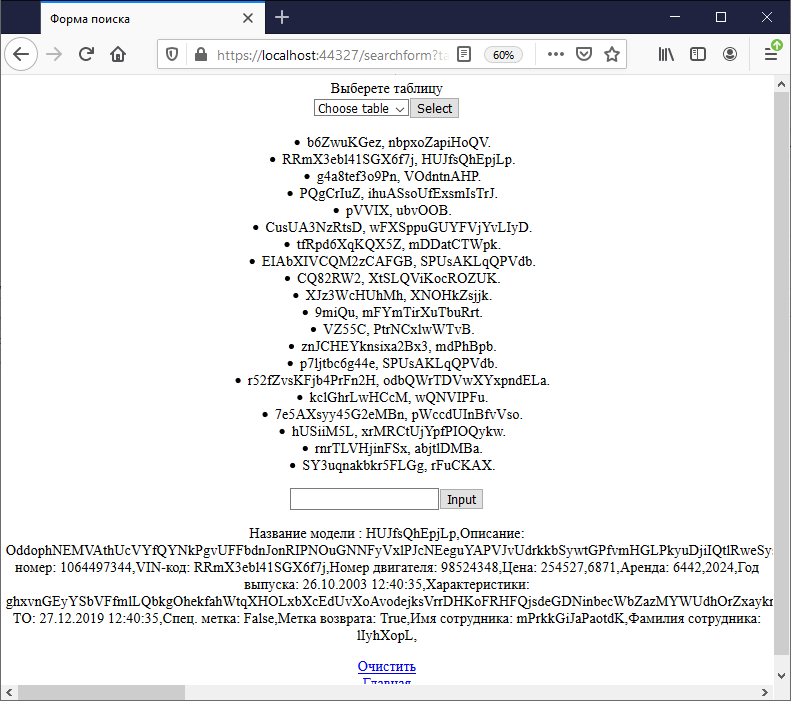


Рисунок 6 – Форма поиска с выводом записи по имени

**3. Вывод:** в результате выполнения данной лабораторной работы были созданы методы обработки *HTTP* средствами *ASP.NET Core*, методы сохранения состояния приложения и повышение производительности приложений путем использования разных видов кэширования.