**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

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

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

**ЛАБОРАТОРНАЯ РАБОТА №3**

по дисциплине: «Разработка приложений баз данных для информационных систем»

на тему: «Обработка *HTTP* запросов средствами *ASP.NET Core*. Сохранение состояния. Кэширование.»

Вариант № 27

Пекарня

Выполнил: студент гр. ИТП-31

Суковатый А.С.

Принял: преподаватель

Ястребов А.А.

Гомель 2023

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

**Задание:**

Используя ранее разработанные объектную модель для доступа к данным в заданной предметной области разработать простое *ASP*.*NET* *Core* приложение.

2.1. С использование методов *Run*, *Map* и *Use* разработать:

1. компоненты промежуточного уровня (*middleware*) и встроить их в конвейер обработки *HTTP* запроса с целью кэширования 20 записей из каждой таблицы базы данных заданной предметной области с помощью встроенного инструмента кэширования ­­­­­­­­­­– объекта *IMemoryCache*. Данные в кэше хранить неизменными в течение 2\**N+*240 секунд, где *N* – номер вашего варианта.

2. собственную систему маршрутизации входящих запросов:

• если *URL* адрес входящего запроса содержит \*info* – выводить в выходной поток для отображения браузером информацию о клиенте и выходить из конвейера обработки запроса;

• если *URL* адрес входящего запроса содержит \*table* (где *table* – имя таблицы из базы данных) – выводить в выходной поток для отображения браузером с использование метода *Response*.*WriteAsync* кэшированную информацию из соответствующей таблицы базы данных и выходить из конвейера обработки запроса;

• если *URL* адрес входящего запроса содержит \*searchform1* или \*searchform2* – выводить в выходной поток для отображения браузером с использование метода *Response*.*WriteAsync* формы для поиска информации из базы данных и выходить из конвейера обработки запроса;

форма должна содержать, как минимум: одно поле, одного поле со списком, один список, одну кнопку;

• в противном случае (*URL* адрес входящего запроса не содержит перечисленных выше элементов) – продолжать обрабатывать другие компоненты конвейера обработки запросов;

2.2. Реализовать сохранение состояния элементов одной формы одной страницы с использованием куки (\*searchform1*).

2.3. Реализовать сохранение состояния элементов одной формы одной страницы в виде одного объекта специальной структуры с использованием объекта *Session* (\*searchform2*).

2.4. Осуществить заполнение элементов формы при их загрузке данными ранее сохранненными в объекте *Session* и куки (\*searchform1*, \*searchform2*).

2.5. С использованием средств разработчика браузера (*Chrome*, *Firefox*) продемонстрировать ускорение обработки запроса при наличии кэширования с использованием *MemoryCache*.

2.6. Разместить выполненный проект на *github*.

**Ход работы**

Для выполнения лабораторной работы в пустой проект *ASP NET* были перенесены объектная модель базы данных из прошлой лабораторной работы, использующая *EF*, и слой доступа к ней.

Для кеширования таблиц базы данных был создан класс *CachedProductsService*, в котором находятся методы занесения и изъятия данных из кэша. После этого были написаны обработчики для запросов с использованием метода *Map*.

После создания всех классов была разработана система маршрутизации входящих запросов, для удобства была создана стартовая страница, представленная на рисунке 1.

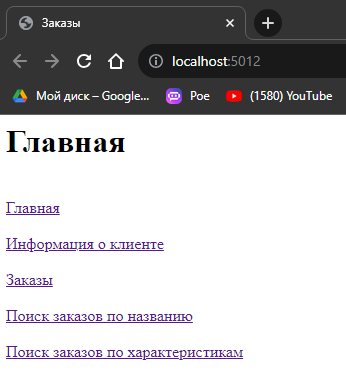


Рисунок 1 – Стартовая страница

На странице «Информация о клиенте» представлена следующая информация:

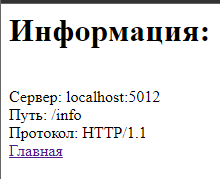


Рисунок 2 –Страница с информацией о клиенте

На странице «Заказы» представлена информация о 20-ти первых заказах:

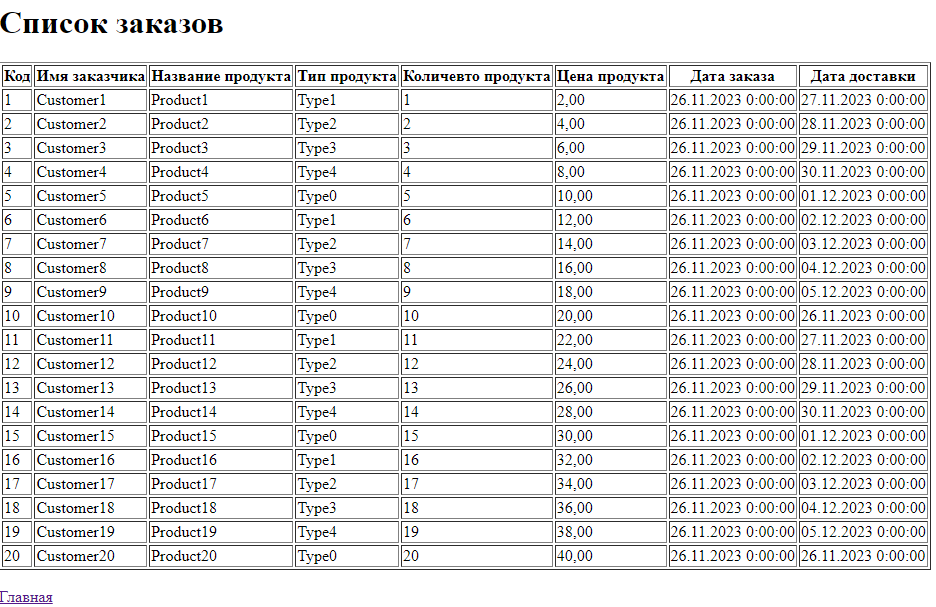


Рисунок 3 – Страница «Заказы»

Далее были реализованы страницы, которые реализуют поля для фильтрации данных в таблице «Заказы». На рисунках 4-5 изображена страница, предоставляющая возможность поиска изделий по названию и запоминает в *Session* значения, введенные в форме.

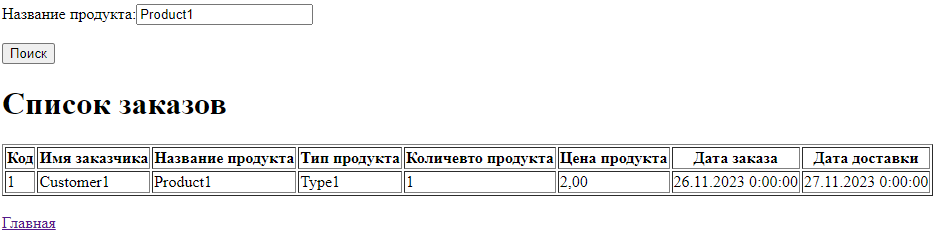


Рисунок 4 – Результат поиска заказа по названию продукта

При повторном посещении страницы в форме должно сохранится название изделия, которое мы искали.

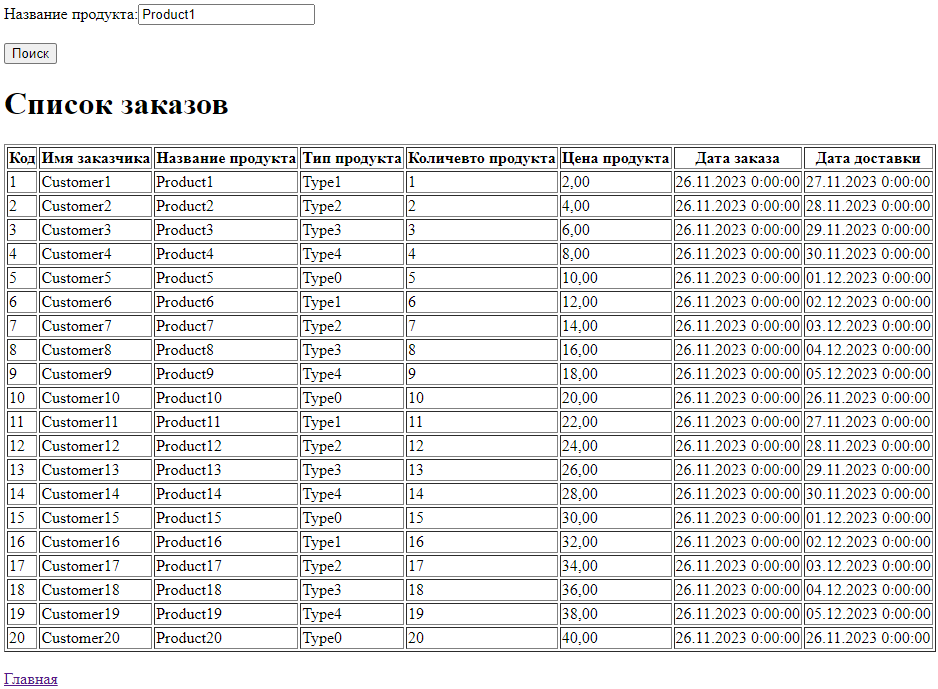


Рисунок 5 – Результат сохранения значения поля формы в *Session*

На рисунках 6-7 изображена страница, предоставляющая возможность фильтрации изделий по бюджету (поиск изделий с ценой ниже либо равной заданной) и цвету изделия. На данной странице реализовано сохранение в *Сookies* значений, введенных в форме.

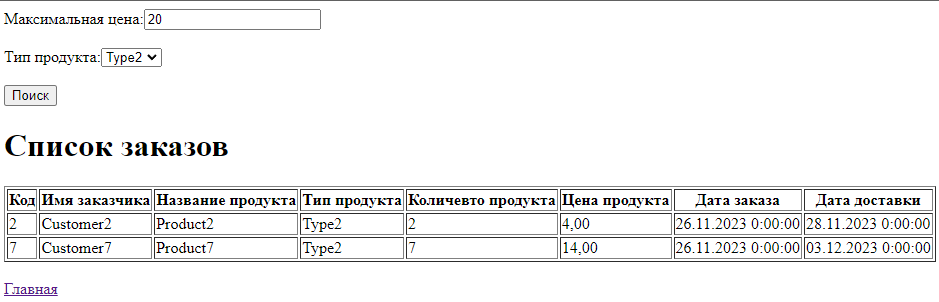


Рисунок 6 – Пример фильтрации изделий по цене и цвету

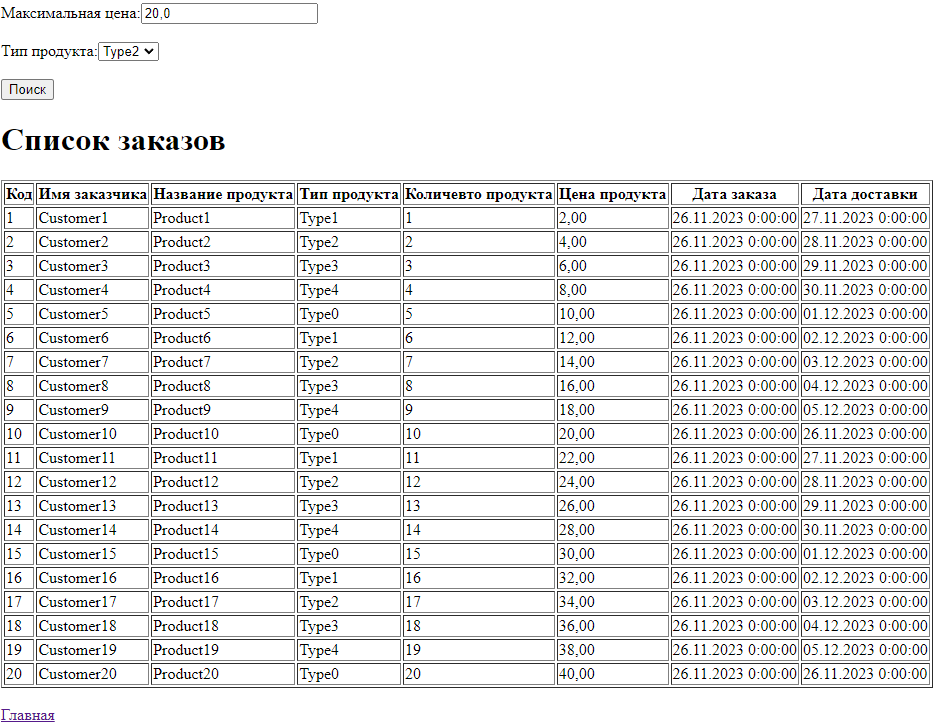


Рисунок 7 – Результат сохранения значений полей формы в *Сookies*

Пример хранения данных в *Session* и *Cookies* указаны на рисунке 8.

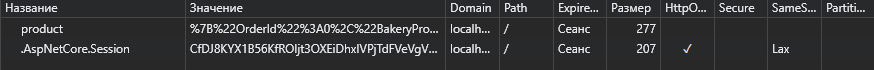


Рисунок 8 – Пример хранения данных в *Session* и *Cookies*

На рисунках 9-10 представлена разница в производительности при наличии кэширования.

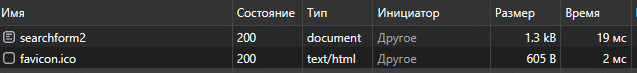


Рисунок 9 – Время загрузки до кэширования

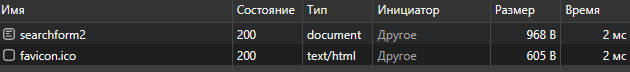


Рисунок 10 – Время загрузки после кэширования

Исходя из разницы в скорости загрузки на первом и втором рисунках (12 раз), мы опытным путём убедились в пользе использования кэширования при разработке веб-приложений.

Ссылка на *GIT*: *https://github.com/UnitiCS/Lab3*

**Вывод**: в ходе выполнения лабораторной работы была освоена технология *ASP .NET* для создания веб-приложений. Изучен интерфейс *IMemoryCache* для кэширования данных, освоены методы обработки запросов с использованием класса *HttpContext* и способы сохранения информации во временных хранилищах с применением технологий *Sessions* и *Cookies*.

**ПРИЛОЖНИЕ А**

Листинг класса *CachedOrdertsService*

using Lab3.Models;

using Microsoft.Extensions.Caching.Memory;

using Microsoft.EntityFrameworkCore.Metadata.Internal;

using Lab3.Data;

namespace Lab3.Services

{

public class CachedOrdersService : ICachedOrdersService

{

private readonly BakeryDBContext \_dbContext;

private readonly IMemoryCache \_memoryCache;

private readonly int \_savingTime;

public CachedOrdersService(BakeryDBContext dbContext, IMemoryCache memoryCache)

{

\_dbContext = dbContext;

\_memoryCache = memoryCache;

\_savingTime = 2 \* 27 + 240;

}

// получение списка материалов из базы

public IEnumerable<Order> GetOrders(int rowsNumber = 20)

{

return \_dbContext.Orders.Take(rowsNumber).ToList();

}

// добавление списка материалов в кэш

public void AddOrders(string cacheKey, int rowsNumber = 20)

{

if (!\_memoryCache.TryGetValue(cacheKey, out IEnumerable<Order> cachedOrders))

{

cachedOrders = \_dbContext.Orders.Take(rowsNumber).ToList();

if (cachedOrders != null)

{

\_memoryCache.Set(cacheKey, cachedOrders, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(\_savingTime)

});

}

Console.WriteLine("Таблица занесена в кеш");

}

else

{

Console.WriteLine("Таблица уже находится в кеше");

}

}

// получение списка матреиалов из кэша или из базы, если нет в кэше

public IEnumerable<Order> GetOrdersFromCache(string cacheKey, int rowsNumber = 20)

{

IEnumerable<Order> orders;

if (!\_memoryCache.TryGetValue(cacheKey, out orders))

{

orders = \_dbContext.Orders.Take(rowsNumber).ToList();

if (orders != null)

{

\_memoryCache.Set(cacheKey, orders,

new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromSeconds(\_savingTime)));

}

}

return orders;

}

//Получение списка уникальных цветов изделий

public IEnumerable<string> GetTypes(IEnumerable<Order> selectedOrders)

{

IEnumerable<string> colors = selectedOrders.Select(o => o.ProductType).Distinct().ToList();

return colors;

}

}

}

Листинг класса *Program*

using Lab3.Models;

using Lab3.Services;

using Lab3.Data;

using Lab3.Infrastructure;

using Microsoft.EntityFrameworkCore;

using Newtonsoft.Json;

namespace FuelStationT

{

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

var services = builder.Services;

// внедрение зависимости для доступа к БД с использованием EF

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

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

// добавление кэширования

services.AddMemoryCache();

// добавление поддержки сессии

services.AddDistributedMemoryCache();

services.AddSession();

// внедрение зависимости CachedMaterialsService

services.AddScoped<ICachedOrdersService, CachedOrdersService>();

builder.Services.AddDistributedMemoryCache();

builder.Services.AddSession();

var app = builder.Build();

// добавляем поддержку сессий

app.UseSession();

//Запоминание в Сookies значений, введенных в форме

app.Map("/searchform1", appBuilder =>

{

appBuilder.Run(async context =>

{

var order = new Order();

ICachedOrdersService cachedOrders = context.RequestServices.GetService<ICachedOrdersService>();

IEnumerable<Order> orders = cachedOrders.GetOrdersFromCache("orders20");

IEnumerable<string> productTypes = cachedOrders.GetTypes(orders);

if (context.Request.Method == "POST")

{

order.Price = decimal.Parse(context.Request.Form["priceLimit"]);

order.ProductType = context.Request.Form["type"];

context.Response.Cookies.Append("product", JsonConvert.SerializeObject(order));

if (order.ProductType != "all")

{

orders = orders.Where(o => o.Price <= order.Price && o.ProductType == order.ProductType);

}

else

{

orders = orders.Where(p => p.Price <= order.Price);

}

}

else if (context.Request.Cookies.ContainsKey("product"))

{

order = JsonConvert.DeserializeObject<Order>(context.Request.Cookies["product"]);

}

string htmlString = "<html><head><title>Заказы</title></head>" +

"<meta http-equiv='Content-Type' content='text/html; charset=utf-8'/>" +

"<body>" +

"<form method='post' action='/searchform1'>" +

"<label>Максимальная цена:</label>" +

$"<input type='text' name='priceLimit' value='{order.Price}' placeholder='Максимальная цена'><br><br>" +

"<label>Тип продукта:</label>" +

"<select name='type'>" +

"<option value='all'>Все</option>";

foreach (var type in productTypes)

{

htmlString += $"<option value='{type}' {(type == order.ProductType ? "selected" : "")}>{type}</option>";

}

htmlString += "</select><br><br>" +

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

"</form>";

htmlString += "<h1>Список заказов</h1>" +

"<table border='1'>";

htmlString += "<TR>";

htmlString += "<TH>Код</TH>";

htmlString += "<TH>Имя заказчика</TH>";

htmlString += "<TH>Название продукта</TH>";

htmlString += "<TH>Тип продукта</TH>";

htmlString += "<TH>Количевто продукта</TH>";

htmlString += "<TH>Цена продукта</TH>";

htmlString += "<TH>Дата заказа</TH>";

htmlString += "<TH>Дата доставки</TH>";

htmlString += "</TR>";

foreach (var o in orders)

{

htmlString += "<tr>" +

$"<td>{o.OrderId}</td>" +

$"<td>{o.CustomerName}</td>" +

$"<td>{o.ProductName}</td>" +

$"<td>{o.ProductType}</td>" +

$"<td>{o.Quantity}</td>" +

$"<td>{o.Price}</td>" +

$"<td>{o.OrderDate}</td>" +

$"<td>{o.DeliveryDate}</td>" +

"</tr>";

}

htmlString += "</table><br><a href='/'>Главная</a></br></body></html>";

await context.Response.WriteAsync(htmlString);

});

});

//Запоминание в Session значений, введенных в форме

app.Map("/searchform2", appBuilder =>

{

appBuilder.Run(async context =>

{

Order order = context.Session.Get<Order>("product") ?? new Order();

ICachedOrdersService cachedProducts = context.RequestServices.GetService<ICachedOrdersService>();

IEnumerable<Order> products = cachedProducts.GetOrdersFromCache("orders20");

if (context.Request.Method == "POST")

{

order.ProductName = context.Request.Form["ProductName"];

context.Session.Set("product", order);

products = products.Where(p => p.ProductName == order.ProductName);

}

string htmlString = "<html><head><title>Заказы</title></head>" +

"<meta http-equiv='Content-Type' content='text/html; charset=utf-8'/>" +

"<body>" +

"<form method='post' action='/searchform2'>" +

"<label>Название продукта:</label>" +

$"<input type='text' name='ProductName' value='{order.ProductName}'><br><br>" +

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

"</form>" +

"<h1>Список заказов</h1>" +

"<table border='1'>";

htmlString += "<TR>";

htmlString += "<TH>Код</TH>";

htmlString += "<TH>Имя заказчика</TH>";

htmlString += "<TH>Название продукта</TH>";

htmlString += "<TH>Тип продукта</TH>";

htmlString += "<TH>Количевто продукта</TH>";

htmlString += "<TH>Цена продукта</TH>";

htmlString += "<TH>Дата заказа</TH>";

htmlString += "<TH>Дата доставки</TH>";

htmlString += "</TR>";

foreach (var o in products)

{

htmlString += "<tr>" +

$"<td>{o.OrderId}</td>" +

$"<td>{o.CustomerName}</td>" +

$"<td>{o.ProductName}</td>" +

$"<td>{o.ProductType}</td>" +

$"<td>{o.Quantity}</td>" +

$"<td>{o.Price}</td>" +

$"<td>{o.OrderDate}</td>" +

$"<td>{o.DeliveryDate}</td>" +

"</tr>";

}

htmlString += "</table><br><a href='/'>Главная</a></br></body></html>";

await context.Response.WriteAsync(htmlString);

});

});

// Вывод информации о клиенте

app.Map("/info", (appBuilder) =>

{

appBuilder.Run(async (context) =>

{

// Формирование строки для вывода

string htmlString = "<HTML><HEAD><TITLE>Информация</TITLE></HEAD>" +

"<META http-equiv='Content-Type' content='text/html; charset=utf-8'/>" +

"<BODY><H1>Информация:</H1>"

+ "<BR> Сервер: " + context.Request.Host

+ "<BR> Путь: " + context.Request.PathBase

+ "<BR> Протокол: " + context.Request.Protocol

+ "<BR><A href='/'>Главная</A></BODY></HTML>";

// Вывод данных

await context.Response.WriteAsync(htmlString);

});

});

// Вывод кэшированной информации из таблицы базы данных

app.Map("/orders", (appBuilder) =>

{

appBuilder.Run(async (context) =>

{

//обращение к сервису

ICachedOrdersService cachedOrdersService = context.RequestServices.GetService<ICachedOrdersService>();

IEnumerable<Order> orders = cachedOrdersService.GetOrdersFromCache("orders20");

string htmlString = "<HTML><HEAD><TITLE>Заказы</TITLE></HEAD>" +

"<META http-equiv='Content-Type' content='text/html; charset=utf-8'/>" +

"<BODY><H1>Список заказов</H1>" +

"<TABLE BORDER=1>";

htmlString += "<TR>";

htmlString += "<TH>Код</TH>";

htmlString += "<TH>Имя заказчика</TH>";

htmlString += "<TH>Название продукта</TH>";

htmlString += "<TH>Тип продукта</TH>";

htmlString += "<TH>Количевто продукта</TH>";

htmlString += "<TH>Цена продукта</TH>";

htmlString += "<TH>Дата заказа</TH>";

htmlString += "<TH>Дата доставки</TH>";

htmlString += "</TR>";

foreach (var o in orders)

{

htmlString += "<tr>" +

$"<td>{o.OrderId}</td>" +

$"<td>{o.CustomerName}</td>" +

$"<td>{o.ProductName}</td>" +

$"<td>{o.ProductType}</td>" +

$"<td>{o.Quantity}</td>" +

$"<td>{o.Price}</td>" +

$"<td>{o.OrderDate}</td>" +

$"<td>{o.DeliveryDate}</td>" +

"</tr>";

}

htmlString += "</TABLE>";

htmlString += "<BR><A href='/'>Главная</A></BR>";

htmlString += "</BODY></HTML>";

// Вывод данных

await context.Response.WriteAsync(htmlString);

});

});

// Стартовая страница и кэширование данных таблицы на web-сервере

app.Run((context) =>

{

//обращение к сервису

ICachedOrdersService cachedMaterials = context.RequestServices.GetService<ICachedOrdersService>();

cachedMaterials.AddOrders("orders20");

string htmlString = "<HTML><HEAD><TITLE>Материалы</TITLE></HEAD>" +

"<META http-equiv='Content-Type' content='text/html; charset=utf-8'/>" +

"<BODY><H1>Главная</H1>"

+ "<BR><A href='/'>Главная</A></BR>"

+ "<BR><A href='/info'>Информация о клиенте</A></BR>"

+ "<BR><A href='/orders'>Заказы</A></BR>"

+ "<BR><A href='/searchform2'>Поиск заказов по названию</A></BR>"

+ "<BR><A href='/searchform1'>Поиск заказов по характеристикам</A></BR>"

+ "</BODY></HTML>";

return context.Response.WriteAsync(htmlString);

});

app.Run();

}

}

}