**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

Выполнил: студент гр. ИТП-31

Солодков М.А

Принял: доцент

Асенчик О.Д.

Гомель 2019

**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.

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 и передавать управление системе маршрутизации MVC фреймворка;

1. Реализовать сохранение состояния элементов одной формы одной страницы с использованием куки.
2. Реализовать сохранение состояния элементов одной формы одной страницы в виде одного объекта специальной структуры с использованием объекта Session.
3. Осуществить заполнение элементов формы при их загрузке данными ранее сохранненными в объекте Session и куки.
4. С использованием средств разработчика браузера (Chrome, Firefox) продемонстрировать ускорение обработки запроса при наличии кэширования с использованием MemoryCache.

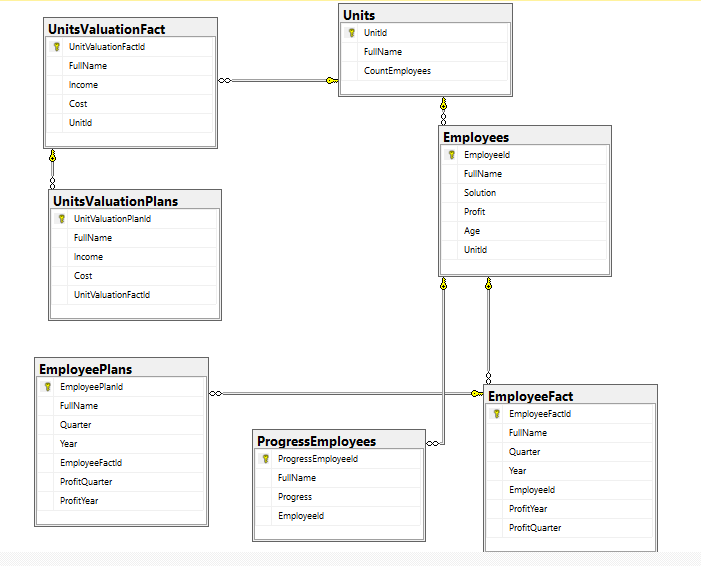


Рисунок 1 – Диаграмма базы данных

**Листинг программы Startup.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.EntityFrameworkCore;

using Company.DATA;

using Company.Services;

using Company.Models;

using Microsoft.AspNetCore.Http;

using System.Web.Mvc;

namespace Company

{

public class Startup

{

private const string LINKS = "<br><a href='info'>Информация о клиенте</a></br>" +

"<br><a href='session-employee'>Сохранение данных с помощью сесии</a></br>" +

"<br><a href='coockie-employee'>Сохранение данных с помощью куки</a></br>" +

"<br><a href='search-employee'>Поиск сотрудников по возрасту</a></br>" +

"<br><a href='table-employee'>Сотрудники</a></br>" +

"<br><a href='table-progressEmployee'>Достижения сотрудников</a></br>" +

"<br><a href='table-unit'>Подразделения компании</a></br>";

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)

{

//services.AddControllersWithViews();

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

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

//добавление кэширования

services.AddMemoryCache();

//добавление поддержки сессии

services.AddDistributedMemoryCache();

services.AddSession();

//внедрение зависимостей

services.AddTransient<CachedUnitService>();

services.AddTransient<CachedEmployeeService>();

services.AddTransient<CachedProgressEmployee>();

}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

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

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseSession();

app.Map("/info", Info); //вывод информации о клиенте

app.Map("/table-progressEmployee", TableProgressEmployee); //вывод информации о достиениях сотрудников

app.Map("/table-unit", TableUnit); //вывод информации о подразделениях компании

app.Map("/search-employee", SearchEmployee); //поиск сотрудника по возрасту

app.Map("/session-employee", SessionEmployee); //сохранение данных с помощью Session

app.Map("/coockie-employee", CoockieEmployee); //сохранение данных с помощью coockie

//вывод записей из таблицы Employee с использованием кэширования

app.Run( async (context) =>

{

CachedEmployeeService cachedEmployeeService = context.RequestServices.GetService<CachedEmployeeService>();

IEnumerable<Employee> employees = cachedEmployeeService.GetEmployee("aPEsRJrLUsIwOiMBbj");

string HtmlString = "<HTML><HEAD><TITLE>Сотрудники</TITLE></HEAD>" +

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

"<BODY><H1>Список Сотрудников</H1>" +

"<TABLE BORDER=1>";

HtmlString += "<TR>";

HtmlString += "<TD>Id сотрудника</TD>";

HtmlString += "<TD>ФИО сотрудника</TD>";

HtmlString += "<TD>Зароботная плата</TD>";

HtmlString += "<TD>Затраты</TD>";

HtmlString += "<TD>Возраст</TD>";

HtmlString += "</TR>";

foreach (var employee in employees)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + employee.EmployeeId + "</TD>";

HtmlString += "<TD>" + employee.FullName + "</TD>";

HtmlString += "<TD>" + employee.Solution + "</TD>";

HtmlString += "<TD>" + employee.Profit + "</TD>";

HtmlString += "<TD>" + employee.Age.ToString() + "</TD>";

HtmlString += "</TR>";

}

HtmlString += "</TABLE>";

HtmlString += LINKS;

HtmlString += "</TABLE></HTML>";

await context.Response.WriteAsync(HtmlString);

});

}

private static void Info(IApplicationBuilder app)

{

app.Run(async (context) =>

{

string HtmlString = "<HTML><HEAD>" +

"<TITLE>Информация о сотруднике</TITLE></HEAD>" +

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

"<BODY><H1>Информация</H1><P>Headers запроса: " + context.Response.Headers.ToString() + "</P>" +

"<P>Статус: " + context.Response.StatusCode + "</P>" +

"<P>IP адресс: " + context.Request.HttpContext.Connection.RemoteIpAddress + "</P>" +

"</BODY></HTML>";

HtmlString += LINKS;

await context.Response.WriteAsync(HtmlString);

});

}

private static void TableProgressEmployee(IApplicationBuilder app)

{

app.Run(async (context) =>

{

CachedProgressEmployee cachedProgressEmployee = context.RequestServices.GetService<CachedProgressEmployee>();

IEnumerable<ProgressEmployee> progressEmployees = cachedProgressEmployee.GetEmployee("jSpBdMSFHvWTdJwB");

string HtmlString = "<HTML><HEAD><TITLE>Достижения сотрудников</TITLE></HEAD>" +

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

"<BODY><H1>Достижения сотрудников</H1>" +

"<TABLE BORDER=1>";

HtmlString += "<TR>";

HtmlString += "<TD>Код достижения</TD>";

HtmlString += "<TD>ФИО сотрудника</TD>";

HtmlString += "<TD>Достижение</TD>";

HtmlString += "<TD>Код сотрудника</TD>";

HtmlString += "</TR>";

foreach (var progressEmployee in progressEmployees)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + progressEmployee.ProgressEmployeeId + "</TD>";

HtmlString += "<TD>" + progressEmployee.FullName + "</TD>";

HtmlString += "<TD>" + progressEmployee.Progress + "</TD>";

HtmlString += "<TD>" + progressEmployee.EmployeeId + "</TD>";

HtmlString += "</TR>";

}

HtmlString += "</TABLE>";

HtmlString += LINKS;

HtmlString += "</TABLE></HTML>";

await context.Response.WriteAsync(HtmlString);

});

}

private static void TableUnit(IApplicationBuilder app)

{

app.Run(async (context) =>

{

CachedUnitService cachedUnitService = context.RequestServices.GetService<CachedUnitService>();

IEnumerable<Unit> units = cachedUnitService.GetUnit("dDAogoAEU");

string HtmlString = "<HTML><HEAD><TITLE>Подразделения компании</TITLE></HEAD>" +

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

"<BODY><H1>Подразделения компании</H1>" +

"<TABLE BORDER=1>";

HtmlString += "<TR>";

HtmlString += "<TD>Код подразделения</TD>";

HtmlString += "<TD>Наименование подразделения</TD>";

HtmlString += "<TD>Количество сотрудников</TD>";

HtmlString += "</TR>";

foreach (var unit in units)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + unit.UnitId + "</TD>";

HtmlString += "<TD>" + unit.FullName + "</TD>";

HtmlString += "<TD>" + unit.CountEmployees + "</TD>";

HtmlString += "</TR>";

}

HtmlString += "</TABLE>";

HtmlString += LINKS;

HtmlString += "</TABLE></HTML>";

await context.Response.WriteAsync(HtmlString);

});

}

private static void SearchEmployee(IApplicationBuilder app)

{

app.Run(async (context) =>

{

string fullName = null;

fullName = context.Request.Query["FullName"];

string HtmlString = "<HTML><HEAD><TITLE>Подразделения компании</TITLE></HEAD>" +

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

"<body><form action = '/search-employee'>" +

"ФИО:<br><input type = 'text' name = 'fullName' value = "+ fullName + ">" + "<br>" +

"<br><input type = 'submit' value = 'Submit'></form></br>";

if (fullName != null)

{

CachedEmployeeService cachedEmployeeService = context.RequestServices.GetService<CachedEmployeeService>();

var employees = cachedEmployeeService.ReadEmployee(fullName);

HtmlString = "<HTML><HEAD><TITLE>Сотрудники компании</TITLE></HEAD>" +

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

"<BODY><H1>Сотрудник</H1>" +

"<TABLE BORDER=1>";

HtmlString += "<TR>";

HtmlString += "<TD>Код сотрудника</TD>";

HtmlString += "<TD>ФИО сотрудника</TD>";

HtmlString += "<TD>ЗП сотрудника</TD>";

HtmlString += "<TD>Затраты компании на сотрудника</TD>";

HtmlString += "<TD>Возраст сотрудника</TD>";

HtmlString += "</TR>";

foreach (Employee employee in employees)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + employee.EmployeeId + "</TD>";

HtmlString += "<TD>" + employee.FullName + "</TD>";

HtmlString += "<TD>" + employee.Solution + "</TD>";

HtmlString += "<TD>" + employee.Profit + "</TD>";

HtmlString += "<TD>" + employee.Age + "</TD>";

HtmlString += "</TR>";

}

HtmlString += "</TABLE>";

HtmlString += LINKS;

HtmlString += "</TABLE></HTML>" + "<br>";

}

await context.Response.WriteAsync(HtmlString);

});

}

//запоминание в Session значений, введенных в форме

private static void SessionEmployee(IApplicationBuilder app)

{

app.Run(async (context) =>

{

string fullName = "";

if (context.Session.Keys.Contains("fullName"))

{

fullName = context.Session.GetString("fullName");

}

string strResponse = "<HTML><HEAD>" +

"<TITLE>Пользователь Session</TITLE></HEAD>" +

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

"<BODY><FORM action ='/session-employee' / >" +

"ФИО:<BR><INPUT type = 'text' name = 'fullName' value = " + fullName + ">" +

"<BR><BR><INPUT type ='submit' value='Сохранить в Session'><INPUT type ='submit' value='Показать'></FORM>";

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

//strResponse += LINKS + "</BODY></HTML>";

if (context.Request.Query.ContainsKey("fullName") && context.Request.Query["fullName"] != "")

{

context.Session.SetString("fullName", context.Request.Query["fullName"]);

}

await context.Response.WriteAsync(strResponse);

});

}

private static void CoockieEmployee(IApplicationBuilder app)

{

app.Run(async (context) =>

{

string fullName;

context.Request.Cookies.TryGetValue("fullName", out fullName);

string strResponse = "<HTML><HEAD>" +

"<TITLE>Пользователь Coockie</TITLE></HEAD>" +

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

"<BODY><FORM action ='/coockie-employee' / >" +

"ФИО:<BR><INPUT type = 'text' name = 'fullName' value = " + fullName + ">" +

"<BR><BR><INPUT type ='submit' value='Сохранить в Coockie'><INPUT type ='submit' value='Показать'></FORM>";

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

if (context.Request.Query.ContainsKey("fullName"))

{

context.Response.Cookies.Append("fullName", context.Request.Query["fullName"]);

}

await context.Response.WriteAsync(strResponse);

});

}

}

}**Листинг программы ChachedEmployeeService.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text.Json;

using System.Threading.Tasks;

using System.Web.Mvc;

using Company.DATA;

using Company.Models;

using Microsoft.AspNetCore.Http;

using Microsoft.Extensions.Caching.Memory;

namespace Company.Services

{

public class CachedEmployeeService

{

private CompanyContext db;

private IMemoryCache cache;

private int rowsNumber;

public CachedEmployeeService(CompanyContext context, IMemoryCache memoryCache)

{

db = context;

cache = memoryCache;

rowsNumber = 20;

}

public IEnumerable<Employee> GetEmployee()

{

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

}

public void AddEmployee(string cacheKey)

{

IEnumerable<Employee> employee = db.Employees.Take(rowsNumber);

cache.Set(cacheKey, employee, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromMinutes(5)

});

}

public IEnumerable<Employee> GetEmployee(string cacheKey)

{

IEnumerable<Employee> employees = null;

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

{

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

if (employees != null)

{

cache.Set(cacheKey, employees,

new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(5)));

}

}

return employees;

}

public IEnumerable<Employee> ReadEmployee(string fullName)

{

return db.Employees.Where(employee => employee.FullName.Equals(fullName));

}

}

}

**Листинг программы CachedProgressEmployee.cs:**

using Company.DATA;

using Company.Models;

using Microsoft.Extensions.Caching.Memory;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace Company.Services

{

public class CachedProgressEmployee

{

private CompanyContext db;

private IMemoryCache cache;

private int rowsNumber;

public CachedProgressEmployee(CompanyContext context, IMemoryCache memoryCache)

{

db = context;

cache = memoryCache;

rowsNumber = 20;

}

public IEnumerable<ProgressEmployee> GetEmployee()

{

return db.ProgressEmployees.Take(rowsNumber).ToList();

}

public void AddEmployee(string cacheKey)

{

IEnumerable<ProgressEmployee> employee = db.ProgressEmployees.Take(rowsNumber);

cache.Set(cacheKey, employee, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromMinutes(5)

});

}

public IEnumerable<ProgressEmployee> GetEmployee(string cacheKey)

{

IEnumerable<ProgressEmployee> employees = null;

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

{

employees = db.ProgressEmployees.Take(rowsNumber).ToList();

if (employees != null)

{

cache.Set(cacheKey, employees,

new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(5)));

}

}

return employees;

}

}

}

**Листинг программы CachedUnitService.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Company.DATA;

using Company.Models;

using Microsoft.Extensions.Caching.Memory;

namespace Company.Services

{

public class CachedUnitService

{

private CompanyContext db;

private IMemoryCache cache;

private int rowsNumber;

public CachedUnitService(CompanyContext context, IMemoryCache memoryCache)

{

db = context;

cache = memoryCache;

rowsNumber = 20;

}

public IEnumerable<Unit> GetUnit()

{

return db.Units.Take(rowsNumber).ToList();

}

public void AddUnit(string cacheKey)

{

IEnumerable<Unit> unit = db.Units.Take(rowsNumber);

cache.Set(cacheKey, unit, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromMinutes(5)

});

}

public IEnumerable<Unit> GetUnit(string cacheKey)

{

IEnumerable<Unit> unit = null;

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

{

unit = db.Units.Take(rowsNumber).ToList();

if (unit != null)

{

cache.Set(cacheKey, unit,

new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(5)));

}

}

return unit;

}

}

}

**Листинг метода с сохранением данных в сессии:**

private static void SessionEmployee(IApplicationBuilder app)

{

app.Run(async (context) =>

{

string fullName = "";

if (context.Session.Keys.Contains("fullName"))

{

fullName = context.Session.GetString("fullName");

}

string strResponse = "<HTML><HEAD>" +

"<TITLE>Пользователь Session</TITLE></HEAD>" +

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

"<BODY><FORM action ='/session-employee' / >" +

"ФИО:<BR><INPUT type = 'text' name = 'fullName' value = " + fullName + ">" +

"<BR><BR><INPUT type ='submit' value='Сохранить в Session'><INPUT type ='submit' value='Показать'></FORM>";

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

//strResponse += LINKS + "</BODY></HTML>";

if (context.Request.Query.ContainsKey("fullName") && context.Request.Query["fullName"] != "")

{

context.Session.SetString("fullName", context.Request.Query["fullName"]);

}

await context.Response.WriteAsync(strResponse);

});

}

**Листинг метода с сохранением данных в Coockie:**

private static void CoockieEmployee(IApplicationBuilder app)

{

app.Run(async (context) =>

{

string fullName;

context.Request.Cookies.TryGetValue("fullName", out fullName);

string strResponse = "<HTML><HEAD>" +

"<TITLE>Пользователь Coockie</TITLE></HEAD>" +

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

"<BODY><FORM action ='/coockie-employee' / >" +

"ФИО:<BR><INPUT type = 'text' name = 'fullName' value = " + fullName + ">" +

"<BR><BR><INPUT type ='submit' value='Сохранить в Coockie'><INPUT type ='submit' value='Показать'></FORM>";

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

if (context.Request.Query.ContainsKey("fullName"))

{

context.Response.Cookies.Append("fullName", context.Request.Query["fullName"]);

}

await context.Response.WriteAsync(strResponse);

});

}

**Результат выполнения:**



Рисунок 2 – Получение кэшированных записей из таблицы Employees



Рисунок 3 – Получение записей из таблицы Posts



Рисунок 4 – Получение записей из таблицы Orders

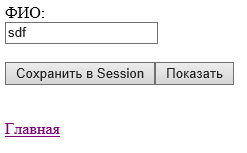
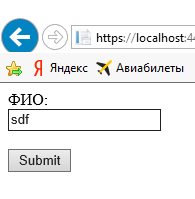


Рисунок 5 – Сохранение данных с помощью сессии



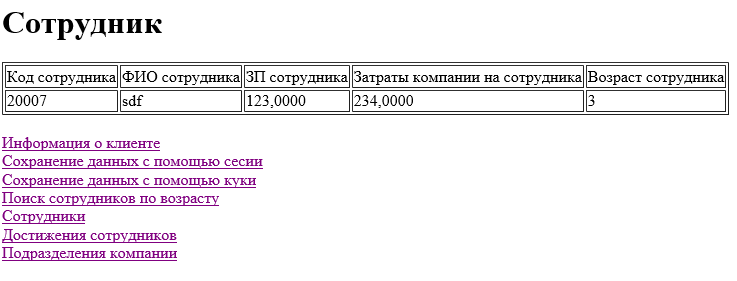


Рисунок 6 – Поиск сотрудника по ФИО

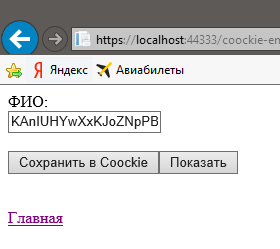
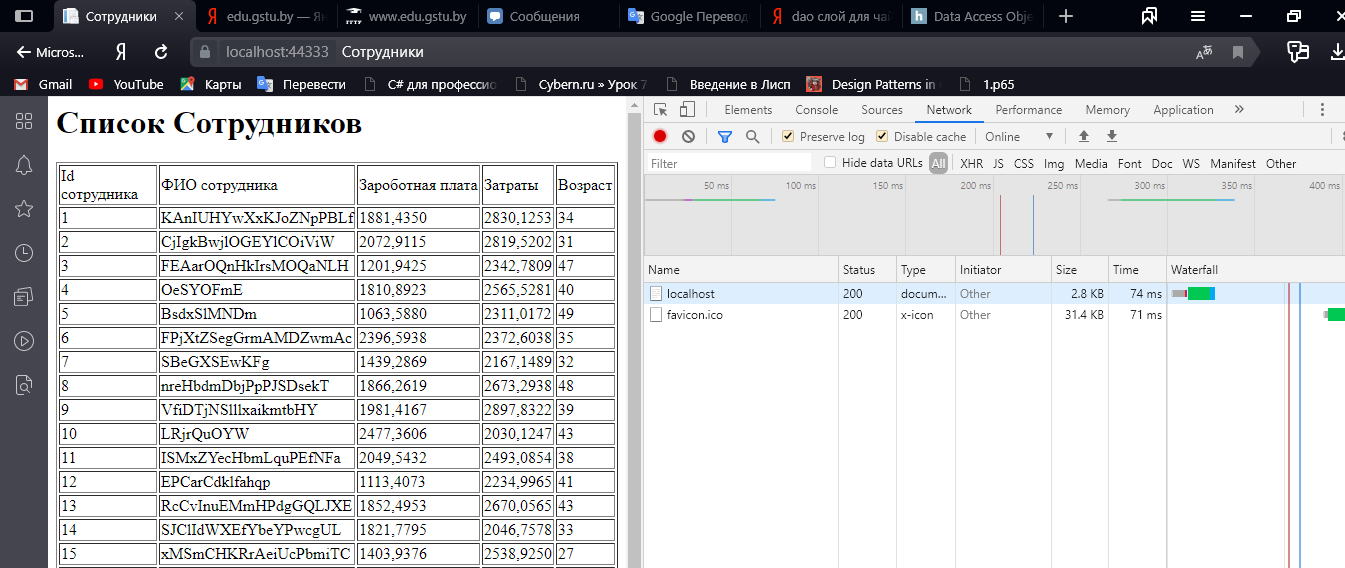


Рисунок 7 – Сохранение данных с помощью куки



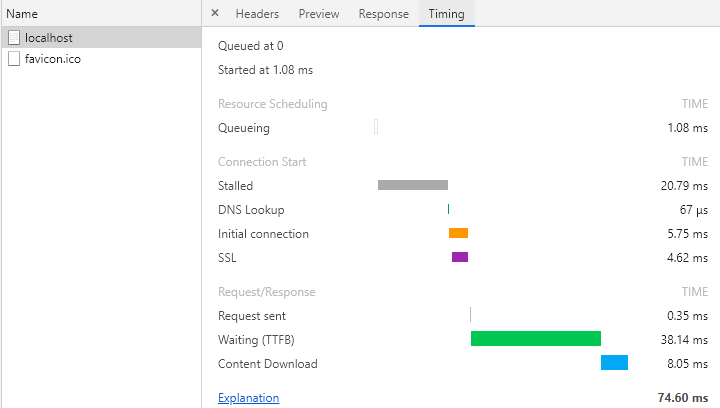


Рисунок 8 – Демонстрировать ускоренной обработки запроса при наличии кэширования с использованием MemoryCache

**Вывод:** в ходе выполнения лабораторной работы были получены навыки работы с методами обработкой HTTP средствами ASP.NET Core, методами сохранения состояния приложения и повышение производительности приложений путем использования разных видов кэширования.