**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: **«РПБДИС»**

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

Выполнил: студент гр. ИТП-31

Сетко А.И.

Принял: старший преподаватель

Асенчик О.Д.

Гомель 2019

**Цель**: 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:

– 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.

– 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;

2. Реализовать сохранение состояния элементов одной формы одной страницы с использованием куки.

3. Реализовать сохранение состояния элементов одной формы одной страницы в виде одного объекта специальной структуры с использованием объекта *Session*.

4. Осуществить заполнение элементов формы при их загрузке данными ранее сохранненными в объекте *Session* и куки.

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

**Ход работы**

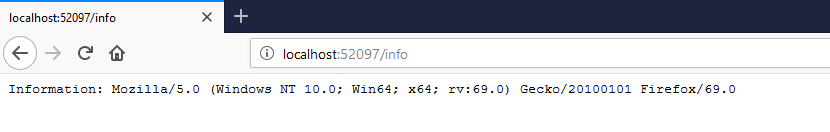
****

Рисунок 1 – Копия экрана браузера с результатом выполнения запроса с *URL* содержащим /*info*

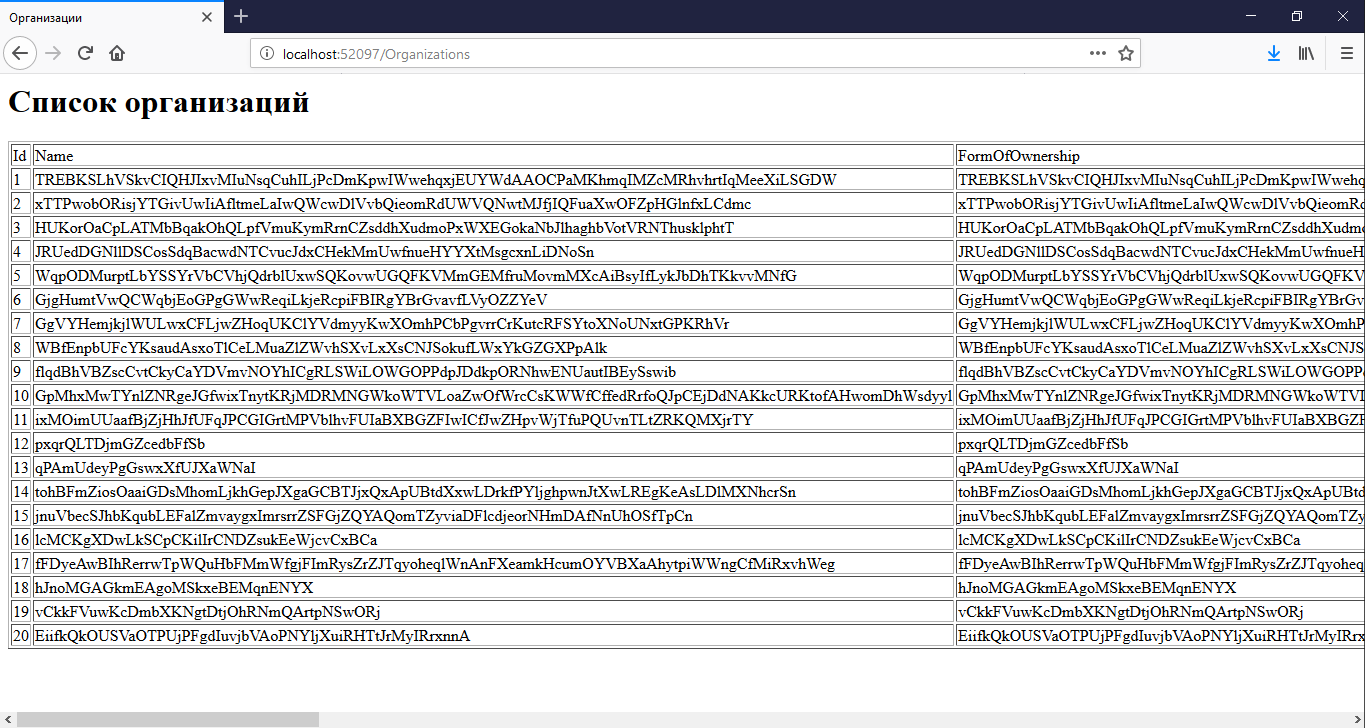


Рисунок 2 – Копия экрана браузера с результатом выполнения запроса с *URL* содержащим /*Organizations*

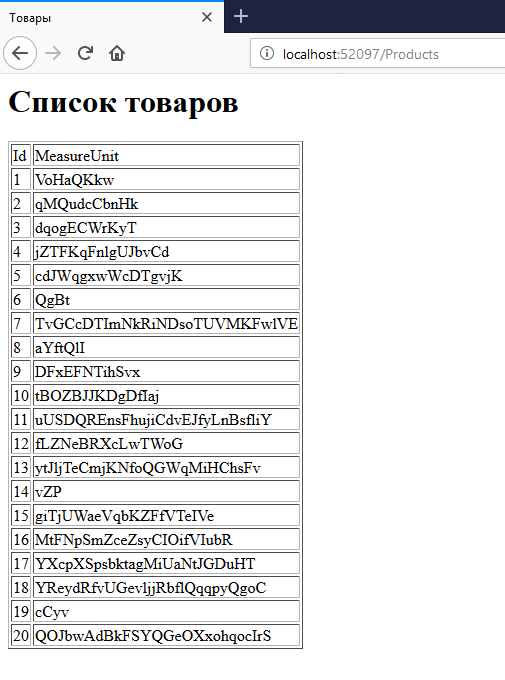


Рисунок 3 – Копия экрана браузера с результатом выполнения запроса с *URL* содержащим /*Products*

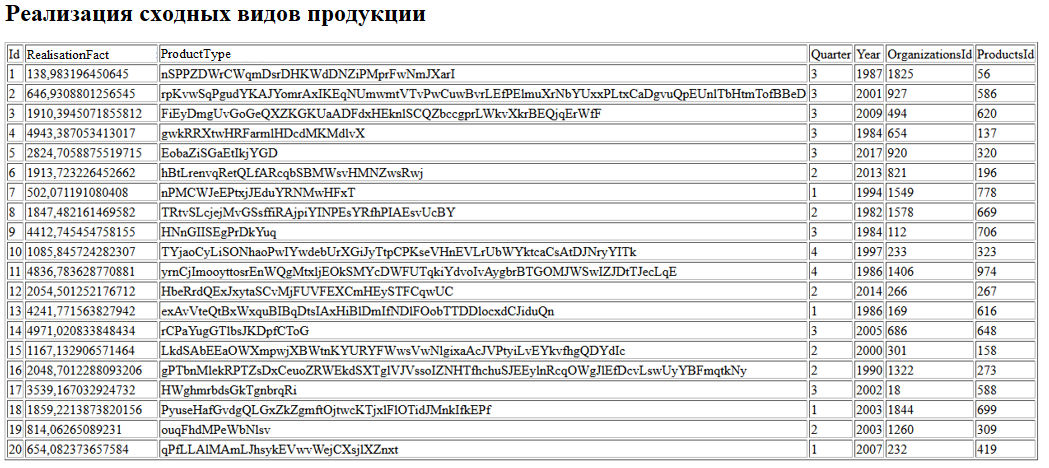


Рисунок 4 – Копия экрана браузера с результатом выполнения запроса с *URL* содержащим /*Realisations*



Рисунок 5 – Копия экрана браузера с результатом выполнения запроса с *URL* содержащим /*Outputs*

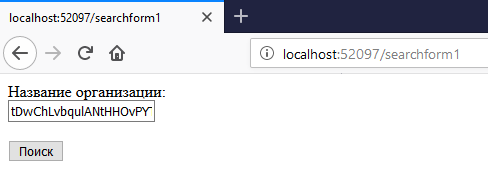
****

Рисунок 6 – Копия экрана браузера с результатом выполнения запроса с *URL* содержащим /*searchform1*

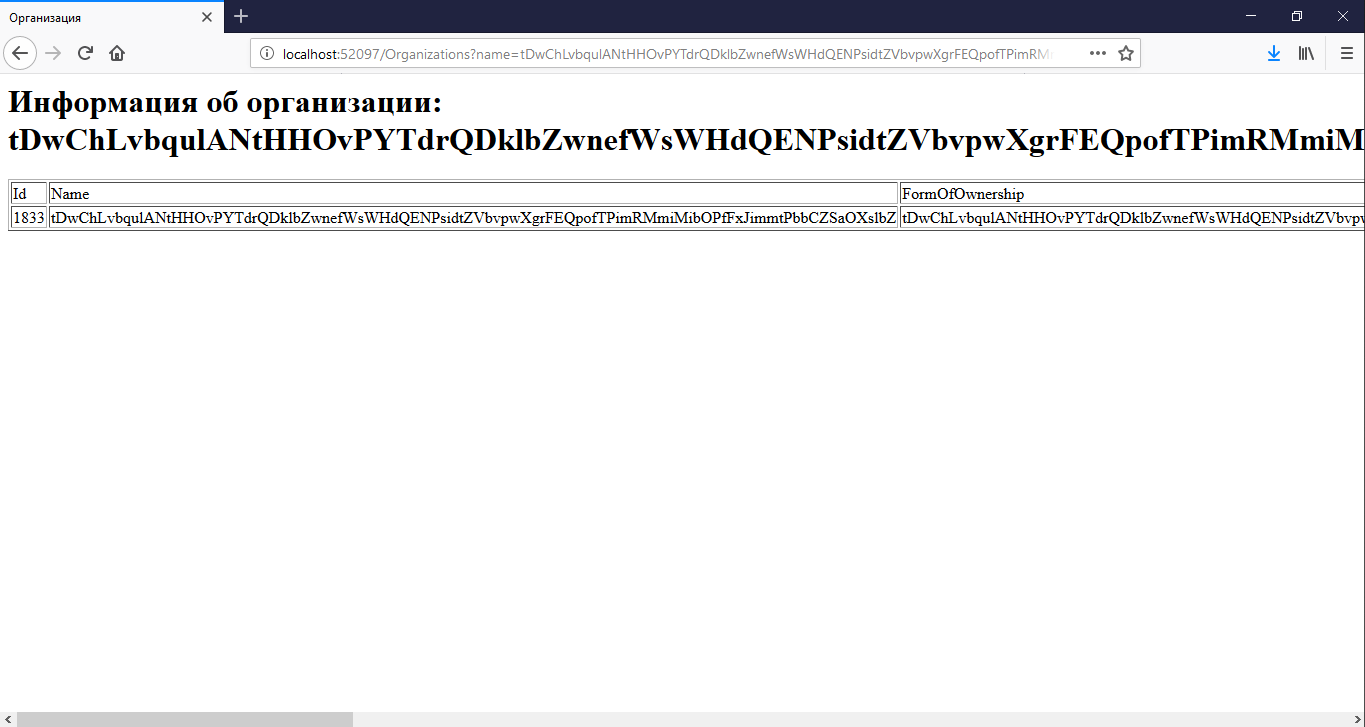


Рисунок 7 – Копия экрана браузера с результатом выполнения поиска с помощью первой формы

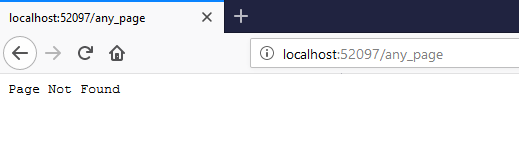


Рисунок 8 – Копия экрана браузера с результатом выполнения запроса с любым другим *URL*

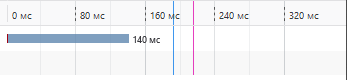


Рисунок 9 – Время обработки страницы без кэширования

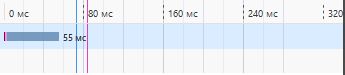


Рисунок 12 – Время обработки страницы с использованием кэширования

**Код классов**

**Класс Startup:**

public class Startup

{

public void ConfigureServices(IServiceCollection services)

{

string connectionString = new ConfigurationBuilder()

.SetBasePath(Directory.GetCurrentDirectory())

.AddJsonFile("appsettings.json")

.Build()

.GetConnectionString("DefaultConnection");

services.AddDbContext<RationingDbContext>(

options => options.UseSqlServer(connectionString));

services.AddMemoryCache();

services.AddRouting();

services.AddDistributedMemoryCache();

services.AddSession();

services.AddTransient<CacheService>();

}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseSession();

app.UseRouting();

app.Map("/info", app => app.Run(context => context.Response.WriteAsync("Information: "

+ context.Request.Headers["User-Agent"])));

app.Map("/Organizations", app => app.Run(context => GetOrganizations(context)));

app.Map("/Products", app => app.Run(context => GetProducts(context)));

app.Map("/Realisations",

app => app.Run(context => GetRealisations(context)));

app.Map("/Outputs",

app => app.Run(context => GetOutputs(context)));

app.Map("/ProductTypes",

app => app.Run(context => GetProductTypes(context)));

app.Map("/searchform1", app => app.Run(context => OpenFirstSearchForm(context)));

app.Map("/organization\_info\_cons\_prods",

app => app.Run(context => GetConsAndProdsAboutOrganizations(context)));

app.Run(async (context) =>

{

await context.Response.WriteAsync("Page Not Found");

});

}

private string GetHtmlForm(string link, string name)

{

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

"<body><form action =" + link + ">" +

"Название организации:<br><input type = 'text' name = 'name' value = " + name + ">" +

"<br><br><input type = 'submit' value = 'Поиск' ></form></body></html>";

}

private Task OpenFirstSearchForm(HttpContext context)

{

string name = "";

if (context.Request.Cookies.ContainsKey("organization\_name"))

{

name = context.Request.Cookies["organization\_name"];

}

return context.Response.WriteAsync(GetHtmlForm("/Organizations", name));

}

private string GetHtmlTable<T>(string title, string tableName, IEnumerable<T> list)

{

StringBuilder builder = new StringBuilder();

builder.Append("<HTML><HEAD><TITLE>");

builder.Append(title);

builder.Append("</TITLE></HEAD><META http-equiv='Content-Type' ");

builder.Append("content='text/html; charset=utf-8'/><BODY><H1>");

builder.Append(tableName);

builder.Append("</H1><TABLE BORDER=1><TR>");

Type type = typeof(T);

List<PropertyInfo> properties = new List<PropertyInfo>();

foreach (PropertyInfo property in type.GetProperties())

{

if (property.PropertyType.IsPrimitive

|| property.PropertyType.Equals(typeof(string)))

{

builder.Append("<TD>");

builder.Append(property.Name);

builder.Append("</TD>");

properties.Add(property);

}

}

builder.Append("</TR>");

foreach (T item in list)

{

builder.Append("<TR>");

foreach (PropertyInfo property in properties)

{

builder.Append("<TD>");

builder.Append(property.GetValue(item));

builder.Append("</TD>");

}

builder.Append("</TR>");

}

builder.Append("</TABLE></BODY></HTML>");

return builder.ToString();

}

private Task GetOrganizations(HttpContext context)

{

CacheService cached = context.RequestServices.GetService<CacheService>();

string name = context.Request.Query["name"];

string title, tableName;

IEnumerable<Organization> organizations;

if (name == null || name.Equals(""))

{

title = "Организации";

tableName = "Список организаций";

organizations = cached.GetOrganizations("Organizations20");

}

else

{

title = "Организация";

tableName = "Информация об организации: " + name;

organizations = cached.GetOrganizations("OrganizationInfo" + name, name);

context.Response.Cookies.Append("organization\_name", name);

}

return context.Response.WriteAsync(GetHtmlTable(title, tableName, organizations));

}

private Task GetOutputsandRealisationsAboutOrganizations(HttpContext context)

{

string name = context.Request.Query["name"];

if (name != null && !name.Equals(""))

context.Session.SetString("organization\_name", name);

else

return context.Response.WriteAsync("");

CacheService cached = context.RequestServices.GetService<CacheService>();

IEnumerable<OutputsRealisationsOfOrganizationView> infos

= cached.GetInfoAboutOrganization("OrganizationInfo" + name, name);

return context.Response.WriteAsync(GetHtmlTable("Информация об организации",

"Выпуск и реализация по организации: " + name, infos));

}

private Task GetProducts(HttpContext context)

{

CacheService cached = context.RequestServices.GetService<CacheService>();

IEnumerable<Product> products = cached.GetProducts("Products20");

return context.Response.WriteAsync(GetHtmlTable("Товары", "Список товаров", products));

}

private Task GetRealisations(HttpContext context)

{

CacheService cached = context.RequestServices.GetService<CacheService>();

IEnumerable<Realisation> rates = cached.GetRealisations("Realisations20");

return context.Response.WriteAsync(GetHtmlTable("Реализация", "Реализация сходных видов продукции", realisations));

}

private Task GetOutputs(HttpContext context)

{

CacheService cached = context.RequestServices.GetService<CacheService>();

IEnumerable<Output> prodAndConses

= cached.GetOutputs("Outputs20");

return context.Response.WriteAsync(GetHtmlTable("Выпуск",

"План и фактический выпуск продукции", outputs));

}

}

**Класс CacheService:**

public class CacheService

{

private const int RowsNumber = 20;

private const int CacheTimeSeconds = 2 \* 2 + 240;

private readonly RationingDbContext db;

private readonly IMemoryCache cache;

public CacheService(RationingDbContext context, IMemoryCache memoryCache)

{

db = context;

cache = memoryCache;

}

public IEnumerable<Organization> GetOrganizations(string cacheKey)

{

if (!cache.TryGetValue(cacheKey, out IEnumerable<Organization> organizations))

{

organizations = db.Organizations.Take(RowsNumber).ToList();

if (organizations != null)

{

cache.Set(cacheKey, organizations,

new MemoryCacheEntryOptions()

.SetAbsoluteExpiration(TimeSpan.FromSeconds(CacheTimeSeconds)));

}

}

return organizations;

}

public IEnumerable<Organization> GetOrganizations(string cacheKey, string name)

{

if (!cache.TryGetValue(cacheKey, out IEnumerable<Organization> organizations))

{

organizations = db.Organizations.Where(item => item.Name == name).ToList();

if (organizations != null)

{

cache.Set(cacheKey, organizations,

new MemoryCacheEntryOptions()

.SetAbsoluteExpiration(TimeSpan.FromSeconds(CacheTimeSeconds)));

}

}

return organizations;

}

public IEnumerable<Product> GetProducts(string cacheKey)

{

if (!cache.TryGetValue(cacheKey, out IEnumerable<Product> products))

{

products = db.Products.Take(RowsNumber).ToList();

if (products != null)

{

cache.Set(cacheKey, products,

new MemoryCacheEntryOptions()

.SetAbsoluteExpiration(TimeSpan.FromSeconds(CacheTimeSeconds)));

}

}

return products;

}

public IEnumerable<Realisaton> GetRealisations(string cacheKey)

{

if (!cache.TryGetValue(cacheKey,

out IEnumerable<Realisation> realisations))

{

realisations = db.Realisations.Take(RowsNumber).ToList();

if (realisations != null)

{

cache.Set(cacheKey, realisations,

new MemoryCacheEntryOptions()

.SetAbsoluteExpiration(TimeSpan.FromSeconds(CacheTimeSeconds)));

}

}

return realisations;

}

public IEnumerable<Output> GetOutputs(string cacheKey)

{

if (!cache.TryGetValue(cacheKey,

out IEnumerable<Output> outputs))

{

outputs = db.Outputs.Take(RowsNumber).ToList();

if (outputs != null)

{

cache.Set(cacheKey, outputss,

new MemoryCacheEntryOptions()

.SetAbsoluteExpiration(TimeSpan.FromSeconds(CacheTimeSeconds)));

}

}

return outputs;

}

public IEnumerable<OutputsRealisationsOfOrganizationView>

GetInfoAboutOrganization(string cacheKey, string name)

{

if (!cache.TryGetValue(cacheKey,

out IEnumerable<OutputsRealisationsOfOrganizationView> infos))

{

infos = db.OutputsRealisationsOfOrganizations

.Where(elem => elem.Name.Equals(name)).ToList();

if (infos != null)

{

cache.Set(cacheKey, infos,

new MemoryCacheEntryOptions()

.SetAbsoluteExpiration(TimeSpan.FromSeconds(CacheTimeSeconds)));

}

}

return infos;

}

}

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