**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

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

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 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. ИТИ-31

Мусафиров А. Ю.

Принял: доцент

Асенчик О. Д.

Гомель 2024

**Цель работы:** ознакомиться c методами обработки *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 282 секунд.
   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;

1. Реализовать сохранение состояния элементов одной формы одной страницы с использованием куки (*\searchform1*).
2. Реализовать сохранение состояния элементов одной формы одной страницы в виде одного объекта специальной структуры с использованием объекта Session (*\searchform2*).
3. Осуществить заполнение элементов формы при их загрузке данными ранее сохранненными в объекте Session и куки (*\searchform1, \searchform2* ).
4. С использованием средств разработчика браузера (*Chrome, Firefox*) продемонстрировать ускорение обработки запроса при наличии кэширования с использованием *MemoryCache.*
5. Разместить выполненный проект на *github.*

**Ход работы**

Для выполнения задания было создано веб-приложение *ASP.NET Core,* и с помощью метода *Run* была выведена на экран стартовая страница. Если не выбран один *URL* маршрутов, то на сайте будет выводиться надпись «Маршрут не найден». Запуск веб-приложения изображен на рисунке 1.

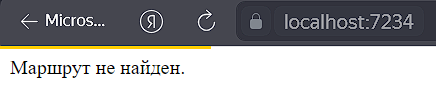


Рисунок 1 – Главная страница приложения

При помощи метода *Map* были осуществлены переходы на уже созданные отдельные страницы приложения по следующим маршрутам: */searchform1, /searchform2, /info, /table.*

После этого был разработан сервис для кэширования данных с использованием *MemoryCache,* код которого помещён в приложение А. Во время разработки сервиса кэширования была осуществлена разработка всех страниц веб-приложения. Первой была разработана страница с выводом информации, которая выглядит, как показано на рисунке 2.

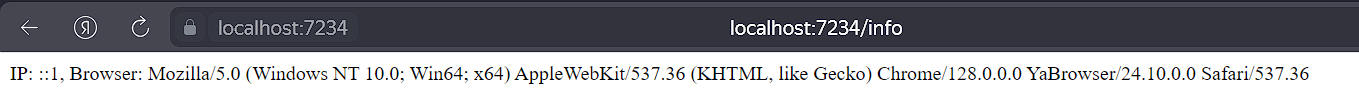


Рисунок 2 – Вывод информации

Страница по маршруту */table*, изображённая на рисунке 3,представляет собой выборку данных из базы данных. Для обращения к определённой таблице необходимо указать корректный маршрут к ней, к примеру /table/Books. Пример работы показан на рисунках 3 − 8 для каждой таблицы.

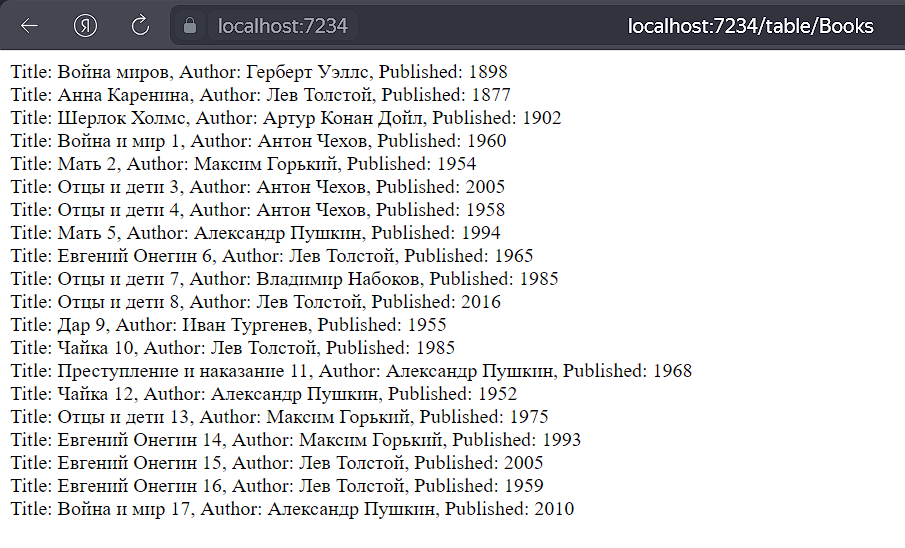


Рисунок 3 – Вывод информации из таблицы *Books*

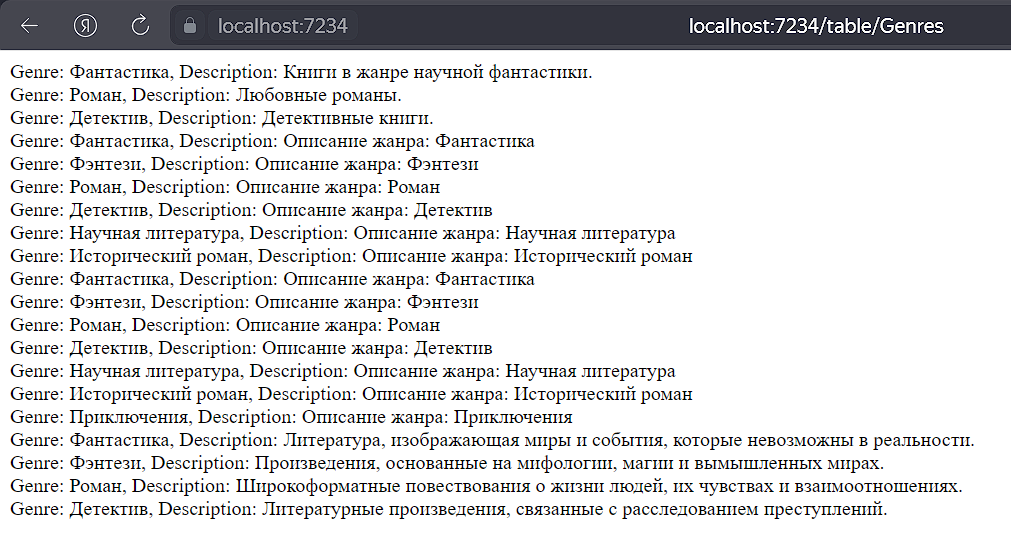


Рисунок 4 – Вывод информации из таблицы *Genres*

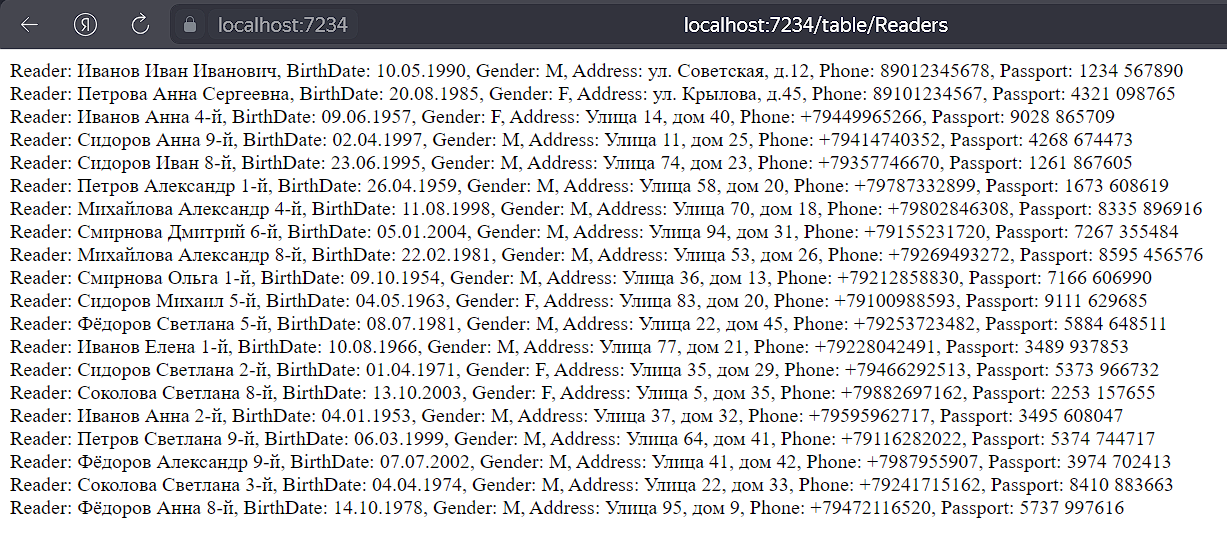


Рисунок 5 – Вывод информации из таблицы *Readers*



Рисунок 6 – Вывод информации из таблицы *Publishers*

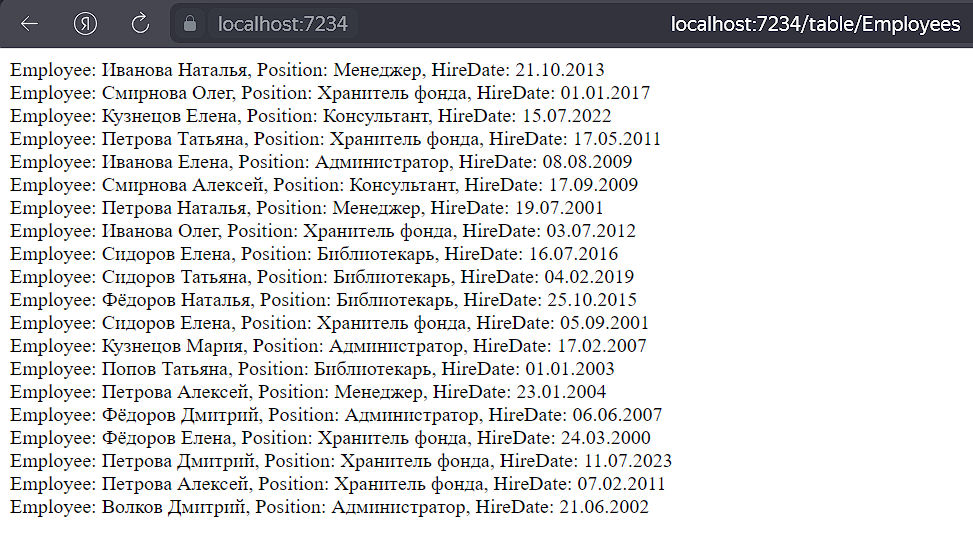


Рисунок 7 – Вывод информации из таблицы *Employees*

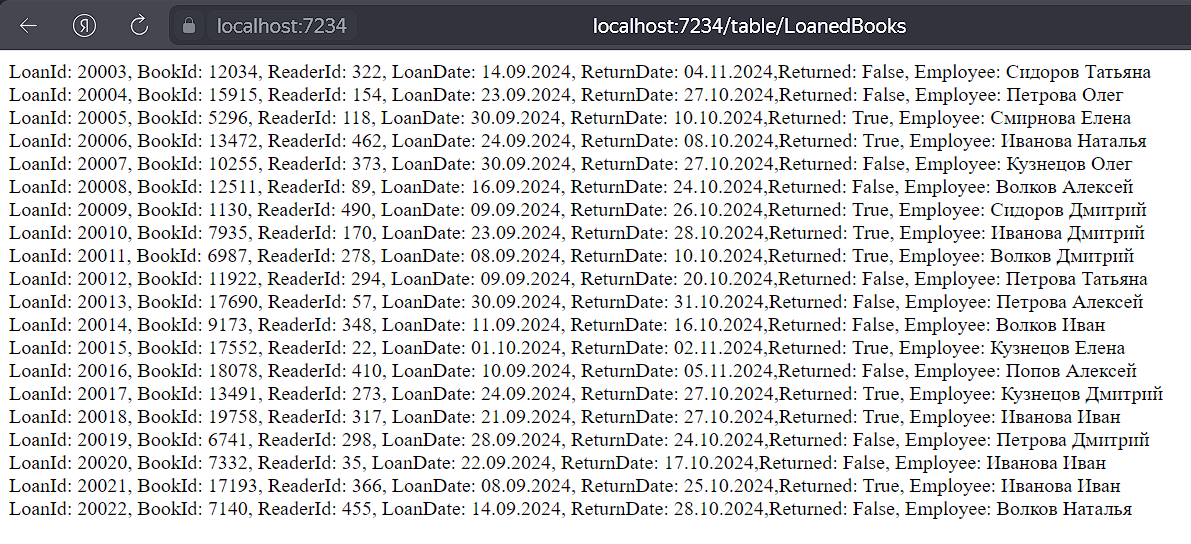


Рисунок 8 – Вывод информации из таблицы *LoanedBooks*

При переходе по маршрутам */searchform1* и */searchform2,* на экран выводится форма. При вводе данных и нажатия кнопки «Поиск» введённые значения сохраняются в куки и извлекаются из них при повторной попытке открыть страницу с формой. Результат работы показан на рисунках 9 − 11

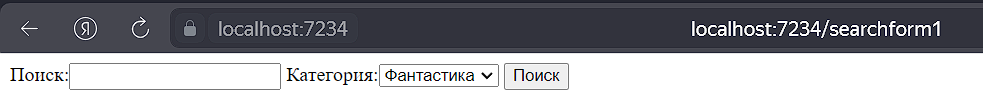


Рисунок 9 – Пустая форма поиска

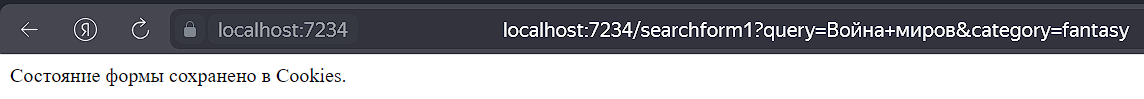


Рисунок 10 – Форма после введённых данных

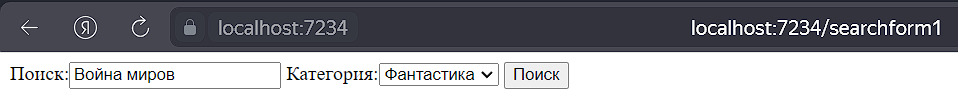


Рисунок 11 – Форма при повторной попытке её открытия

Обе формы сохраняют данные после заполнения. Данные первой формы сохраняются при помощи куки. Данные второй формы хранятся в сессии. При повторном обращении к формам, они автоматически заполняются сохранёнными данными. Код приложения помещён в приложение А. Создан репозиторий на *GitHub*, с ссылкой [*https://github.com/lefas13/CustomDB/tree/lab3*](https://github.com/lefas13/CustomDB/tree/lab3)

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

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Листинг программы**

**Program*.*cs**

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Http;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Caching.Memory;

using Microsoft.Extensions.DependencyInjection;

using RPBDIS\_LibraryDB\_lab3.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddDbContext<LibraryDbContext>(options =>

options.UseSqlServer("Server=DESKTOP-2GTDQ2V\\SQLEXPRESS;Database=LibraryDB;Trusted\_Connection=True;TrustServerCertificate=True;"));

builder.Services.AddMemoryCache();

builder.Services.AddDistributedMemoryCache();

builder.Services.AddSession();

var app = builder.Build();

app.UseRouting(); // Добавляем для активации системы маршрутизации

app.Use(async (context, next) =>

{

Console.WriteLine("Setting response encoding to UTF-8.");

context.Response.ContentType = "text/html; charset=utf-8";

await next();

});

// Middleware для кэширования данных

app.Use(async (context, next) =>

{

Console.WriteLine("Entering caching middleware.");

var dbContext = context.RequestServices.GetRequiredService<LibraryDbContext>();

var cache = context.RequestServices.GetRequiredService<IMemoryCache>();

var tables = new List<string> { "Genres", "Readers", "Books", "Publishers", "LoanedBooks", "Employees" };

var cacheDuration = TimeSpan.FromSeconds(282);

foreach (var table in tables)

{

if (!cache.TryGetValue(table, out List<object> cachedData))

{

Console.WriteLine($"Caching data for table: {table}");

cachedData = FetchTableData(table, dbContext);

cache.Set(table, cachedData.Take(20).ToList(), cacheDuration);

}

else

{

Console.WriteLine($"Data for table {table} is already cached.");

}

}

await next();

});

List<object> FetchTableData(string tableName, LibraryDbContext dbContext)

{

Console.WriteLine($"Fetching data from table: {tableName}");

switch (tableName)

{

case "Genres":

return dbContext.Genres.Take(20).ToList<object>();

case "Readers":

return dbContext.Readers.Take(20).ToList<object>();

case "Books":

return dbContext.Books.Take(20).ToList<object>();

case "Publishers":

return dbContext.Publishers.Take(20).ToList<object>();

case "LoanedBooks":

return dbContext.LoanedBooks.Take(20).ToList<object>();

case "Employees":

return dbContext.Employees.Take(20).ToList<object>();

default:

Console.WriteLine("Table not found in DbContext.");

return new List<object>();

}

}

app.UseSession();

app.UseEndpoints(endpoints =>

{

endpoints.MapGet("/info", async context =>

{

Console.WriteLine("Processing /info route.");

var clientInfo = $"IP: {context.Connection.RemoteIpAddress}, Browser: {context.Request.Headers["User-Agent"]}";

await context.Response.WriteAsync(clientInfo);

});

endpoints.MapGet("/table/{tableName}", async context =>

{

var tableName = context.Request.RouteValues["tableName"].ToString();

Console.WriteLine($"Processing /table/{tableName} route.");

var cache = context.RequestServices.GetRequiredService<IMemoryCache>();

if (cache.TryGetValue(tableName, out List<object> cachedData))

{

Console.WriteLine($"Found cached data for table: {tableName}");

foreach (var item in cachedData)

{

if (item is Book book)

{

await context.Response.WriteAsync($"Title: {book.Title}, Author: {book.Author}, " +

$"Published: {book.PublishYear}<br>");

}

else if (item is Genre genre)

{

await context.Response.WriteAsync($"Genre: {genre.Name}, Description: {genre.Description}<br>");

}

else if (item is Reader reader)

{

await context.Response.WriteAsync($"Reader: {reader.FullName}, BirthDate: {reader.BirthDate}, " +

$"Gender: {reader.Gender}, Address: {reader.Address}, Phone: {reader.Phone}, Passport: {reader.Passport} <br>");

}

else if (item is Employee employee)

{

await context.Response.WriteAsync($"Employee: {employee.FullName}, Position: {employee.Position}, " +

$"HireDate: {employee.HireDate}<br>");

}

else if (item is LoanedBook loanedBook)

{

await context.Response.WriteAsync($"LoanId: {loanedBook.LoanId}, BookId: {loanedBook.BookId}, " +

$"ReaderId: {loanedBook.ReaderId}, LoanDate: {loanedBook.LoanDate}, ReturnDate: {loanedBook.ReturnDate}," +

$"Returned: {loanedBook.Returned}, Employee: {loanedBook.Employee}<br>");

}

else if (item is Publisher publisher)

{

await context.Response.WriteAsync($"Title: {publisher.Name}, Author: {publisher.City}, " +

$"Published: {publisher.Address}<br>");

}

}

}

else

{

Console.WriteLine($"No cached data found for table: {tableName}");

await context.Response.WriteAsync("Данные не найдены.");

}

});

endpoints.MapGet("/searchform1", async context =>

{

Console.WriteLine("Processing /searchform1 route.");

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

{

context.Response.Cookies.Append("searchform1\_query", context.Request.Query["query"]);

context.Response.Cookies.Append("searchform1\_category", context.Request.Query["category"]);

await context.Response.WriteAsync("Состояние формы сохранено в Cookies.");

}

else

{

var query = context.Request.Cookies["searchform1\_query"] ?? "";

var category = context.Request.Cookies["searchform1\_category"] ?? "";

Console.WriteLine("Restoring form state from Cookies.");

string formHtml = @$"

<form method='get' action='/searchform1'>

<label>Поиск:</label><input type='text' name='query' value='{query}' />

<label>Категория:</label><select name='category'>

<option value='fantasy' {(category == "fantasy" ? "selected" : "")}>Фантастика</option>

<option value='detective' {(category == "detective" ? "selected" : "")}>Детектив</option>

<option value='science' {(category == "science" ? "selected" : "")}>Наука</option>

<option value='history' {(category == "history" ? "selected" : "")}>История</option>

</select>

<input type='submit' value='Поиск' />

</form>";

await context.Response.WriteAsync(formHtml);

}

});

endpoints.MapGet("/searchform2", async context =>

{

Console.WriteLine("Processing /searchform2 route.");

await context.Session.LoadAsync();

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

{

context.Session.SetString("searchform2\_query", context.Request.Query["query"]);

context.Session.SetString("searchform2\_category", context.Request.Query["category"]);

await context.Response.WriteAsync("Состояние формы сохранено в Session.");

}

else

{

var query = context.Session.GetString("searchform2\_query") ?? "";

var category = context.Session.GetString("searchform2\_category") ?? "";

Console.WriteLine("Restoring form state from Session.");

string formHtml = @$"

<form method='get' action='/searchform1'>

<label>Поиск:</label><input type='text' name='query' value='{query}' />

<label>Категория:</label><select name='category'>

<option value='fantasy' {(category == "fantasy" ? "selected" : "")}>Фантастика</option>

<option value='detective' {(category == "detective" ? "selected" : "")}>Детектив</option>

<option value='science' {(category == "science" ? "selected" : "")}>Наука</option>

<option value='history' {(category == "history" ? "selected" : "")}>История</option>

</select>

<input type='submit' value='Поиск' />

</form>";

await context.Response.WriteAsync(formHtml);

}

});

});

app.Run(async context =>

{

Console.WriteLine("Route not found.");

await context.Response.WriteAsync("Маршрут не найден.");

});

app.Run();