

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

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

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №4

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

на тему: «Разработка моделей и контроллеров *ASP.NET MVC* приложения баз данных»

Выполнил: студент гр. ИТП-31

Король В. Н.

Принял: ректор

Асенчик О.Д.

Гомель 2023

Цель работы: ознакомиться с возможностями *ASP.NET Core MVC* и *Entity Framework Core* для разработки слоя доступа к данным, хранящимся в базе данных, и обработки запросов пользователя посредством контроллеров.

Задание:

Создать с использованием *ASP.NET Core MVC Web*-приложение, содержащее набор классов, моделирующих предметную область, и осуществляющих генерацию и заполнение тестовыми наборами записей базу данных. Разработать один компонент *middleware*, контроллеры и представления для выборки и отображения информации из не менее чем 3-таблиц базы данных с использованием механизма внедрение зависимостей.

Для выполнения задания необходимо создать:

- Классы, моделирующие не менее чем три таблицы базы данных согласно вашему варианту. Перечень таблиц предварительно согласовывается с преподавателем. Одна из таблиц обязательно должна находиться на стороне отношения «многие» связи с другой таблицей в схеме базы данных.

- Класс контекста данных.

- Другие классы, например, классы *View Model* и т.п. (при необходимости).

- Компонент *middleware*, вызываемый в классе *Startup*, для инициализации базы данных путем заполнения ее таблиц тестовым набором записей.

- Классы контроллеров (по одному на каждую таблицу базы данных) для обработки обращений пользователя, выборки данных из таблиц и вызова соответствующих представлений для отображения выбранных данных.

- Разработать представления для отображения данных из таблиц, выбранных контроллерами. Представления, работающими с таблицами, стоящими на стороне отношения «многие» в схеме базы данных, должны выводить вместо кодов внешних ключей смысловые значения из связанных таблиц, стоящих на стороне отношения «один».

- Используя предварительно созданный и сконфигурированный в классе *Startup* профиль кэширования, подключить кэширование вывода для страниц с использованием атрибута *ResponseCache* для соответствующих методов контроллера. Данные в кэше хранить неизменными в течение $2 * N + 240$ секунд, где N – номер вашего варианта.

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

Для проверки преподавателем следует разместить разработанный проект на *GitHub*.

Ход работы

В ходе выполнения лабораторной работы при помощи *Entity framework* были перенесены три модели которые были связаны между собой. Строка подключения к базе данных хранится в файле *appsettings.json*. Класс контекста

был внедрен в приложение при помощи *DI*. Листинг этих моделей класса контекста.

Далее были разработаны контроллеры, которые используются для передачи данных, которые хранятся в моделях в представления. Были разработаны три контроллера *AgentTypeController*, *ContractController* и *InsuranceAgentController*. Каждый контроллер используется для работы с каждой моделью. Листинг всех этих контролеров указан в приложении А.

Далее были разработаны представления выводя данных из контроллера в *HTML* страницу. Пример этих представлений указан в приложении А.

Пример представления с информацией о типах агентов указан на рисунке 1.

lab4 Типы страховых агентов Контракты Страховые агенты

Id	Название
1	Штатный работник
2	Совместитель

Рисунок 1 – Пример страницы с информацией о типах агентов

Пример представления с информацией о контрактах указан на рисунке 1.

lab4 Типы страховых агентов Контракты Страховые агенты

Id	Начало контракта	Конец контракта	Обязанности
1	12/1/2024	3/30/2021	Администратор страховых полисов
2	3/6/2017	8/19/2016	Эксперт по риску
3	7/14/2020	9/15/2033	Актюарий
4	2/20/2023	5/30/2027	Актюарий
5	6/1/2019	5/19/2016	Аналитик по страхованию
6	2/5/2027	10/13/2016	Управляющий отделом страхования
7	6/6/2018	4/9/2022	Администратор базы данных страхования
8	4/20/2027	12/31/2029	Управляющий отделом страхования
9	8/23/2014	4/16/2014	Аналитик по страхованию
10	5/28/2025	11/16/2022	Эксперт по риску

Рисунок 2 – Пример страницы с информацией о контрактах

Пример представления с информацией о страховых агентах указан на рисунке 3.

lab4 Типы страховых агентов Контракты Страховые агенты

Id	Фамилия	Имя	Отчество	Тип агента	Зарплата	Начало контракта	Конец контракта	Обязанности	Процент от сделки
1	Смирнов	Михаил	Геннадьевич	Штатный работник	44.0152	10/6/2020	6/29/2033	Эксперт по риску	0.189941349028509
2	Кузнецов	Екатерина	Андреевич	Штатный работник	0.5595	3/7/2032	8/7/2024	Аналитик по страхованию	0.07274553424585331
3	Сидоров	Екатерина	Дмитриевич	Совместитель	39.2926	10/6/2020	6/29/2033	Эксперт по риску	0.3073526914185793
4	Кузнецов	Михаил	Андреевич	Штатный работник	2.4896	3/21/2015	6/9/2023	Страховой агент	0.4285488874111939

Рисунок 3 – Пример страницы с информацией о страховых агентах

Далее для более быстрого доступа ко вкладкам сайта при помощи класса *ResponseCache* было реализовано кэширование запросов. В классе *Program* был добавлен новый профиль кэш. А в контролерах был указан атрибут, в котором указывается имя этого профиля. Пример запроса с кэшированием указан на рисунке 4.

lab4/	General
ShowTable	Request URL: http://localhost:5253/Contract/ShowTable
bootstrap.min.css	Request Method: GET
site.css?v=pAGv4ietcJNk_EwsQ...	Status Code: 200 OK (from disk cache)
lab4.styles.css?v=SK8CnH-DH...	Remote Address: [::1]:5253
jquery.min.js	Referrer Policy: strict-origin-when-cross-origin
bootstrap.bundle.min.js	Response Headers
site.js?v=4q1jwFhaPaZgr8WAU...	Cache-Control: public,max-age=272
browserLink	Content-Type: text/html; charset=utf-8
aspnetcore-browser-refresh.js	Date: Sat, 28 Oct 2023 17:53:54 GMT
negotiate?requestUrl=http%3...	Server: Kestrel
connect?transport=webSocket...	X-Dns-Prefetch-Control: off

Рисунок 4 – Пример кэшированного запроса

После выполнения лабораторной работы созданные проект был добавлен в локальный *git* репозиторий а потом перенесен в *GitHub* репозиторий своего аккаунта. Чтобы ознакомиться с созданным проектом можно по ссылке [Javaro3/lab3_DB\(github.com\)](https://github.com/Javaro3/lab3_DB).

Вывод: в ходе выполнения лабораторной работы была изучена такая технология *ASP .NET Core MVC*. Было разработаны классы моделей и контекста, предназначенные для работы с данными. Классы контроллера для связи моделей с представлениями. Классы представления, предназначенные для вывода результата. Был изучен класс *ResponseCache* для хранения методов контроллеров в кэше и более быстрого доступа к данным.

ПРИЛОЖЕНИЕ А

Листинг класса *Program*

```
using lab4.Data;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

internal class Program {
    private static void Main(string[] args) {
        var builder = WebApplication.CreateBuilder(args);

        string connectionString =
builder.Configuration.GetConnectionString("MSSQL");
        builder.Services.AddDbContext<InsuranceCompanyContext>(option =>
option.UseSqlServer(connectionString));
        builder.Services.AddControllersWithViews(options => {
            options.CacheProfiles.Add("ModelCache",
                new CacheProfile() {
                    Location = ResponseCacheLocation.Any,
                    Duration = 2*16+240
                });
        });

        var app = builder.Build();

        if (!app.Environment.IsDevelopment()) {
            app.UseExceptionHandler("/Home/Error");
        }

        app.UseStaticFiles();

        app.UseRouting();

        app.UseAuthorization();

        app.MapControllerRoute(
            name: "default",
            pattern: "{controller=Home}/{action=Index}/{id?}");

        app.MapControllerRoute(
            name: "agentType",
            pattern: "{controller=AgentType}/{action=ShowTable}");

        app.MapControllerRoute(
            name: "contract",
            pattern: "{controller=Contract}/{action=ShowTable}");

        app.MapControllerRoute(
            name: "insuranceAgent",
            pattern: "{controller=InsuranceAgent}/{action=ShowTable}");

        app.Run();
    }
}
```

Листинг класса *AgentTypeController*

```
using lab4.Data;
using Microsoft.AspNetCore.Mvc;

namespace lab4.Controllers
{
    public class AgentTypeController : Controller {
        private InsuranceCompanyContext db;
    }
}
```

```

        public AgentTypeController(InsuranceCompanyContext context) {
            db = context;
        }

        [ResponseCache(CacheProfileName = "ModelCache")]
        public IActionResult ShowTable() {
            var agentTypes = db.AgentTypes.ToList();
            return View(agentTypes);
        }
    }
}

```

Листинг класса *ContractController*

```

using lab4.Data;
using Microsoft.AspNetCore.Mvc;

namespace lab4.Controllers {
    public class ContractController : Controller {
        private InsuranceCompanyContext db;

        public ContractController(InsuranceCompanyContext context) {
            db = context;
        }

        [ResponseCache(CacheProfileName = "ModelCache")]
        public IActionResult ShowTable() {
            var contracts = db.Contracts.ToList();
            return View(contracts);
        }
    }
}

```

Листинг класса *InsuranceAgentController*

```

using lab4.Data;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace lab4.Controllers {
    public class InsuranceAgentController : Controller {
        private InsuranceCompanyContext db;

        public InsuranceAgentController(InsuranceCompanyContext context) {
            db = context;
        }

        [ResponseCache(CacheProfileName = "ModelCache")]
        public IActionResult ShowTable() {
            var insuranceAgents = db.InsuranceAgents
                .Include(ia => ia.AgentTypeNavigation)
                .Include(ia => ia.ContractNavigation)
                .ToList();
            return View(insuranceAgents);
        }
    }
}

```

Листинг класса *InsuranceCompanyContext*

```

using lab4.Models;
using Microsoft.EntityFrameworkCore;

namespace lab4.Data;

public partial class InsuranceCompanyContext : DbContext

```

```

{
    public InsuranceCompanyContext()
    {
    }

    public InsuranceCompanyContext(DbContextOptions<InsuranceCompanyContext>
options)
        : base(options)
    {
    }

    public virtual DbSet<AgentType> AgentTypes { get; set; }

    public virtual DbSet<Contract> Contracts { get; set; }

    public virtual DbSet<InsuranceAgent> InsuranceAgents { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder) {
        modelBuilder.Entity<InsuranceAgent>(entity => {
            entity.HasOne(d => d.AgentTypeNavigation).WithMany(p =>
p.InsuranceAgents)
                .HasForeignKey(d => d.AgentType)
                .OnDelete(DeleteBehavior.ClientSetNull);

            entity.HasOne(d => d.ContractNavigation).WithMany(p =>
p.InsuranceAgents)
                .HasForeignKey(d => d.Contract)
                .OnDelete(DeleteBehavior.ClientSetNull);
        });

        modelBuilder.OnModelCreatingPartial(modelBuilder);
    }

    partial void OnModelCreatingPartial(ModelBuilder modelBuilder);
}

```

Листинг класса *AgentType*

```

using System;
using System.Collections.Generic;

namespace lab4.Models;

public partial class AgentType
{
    public int Id { get; set; }

    public string Type { get; set; } = null!;

    public virtual ICollection<InsuranceAgent> InsuranceAgents { get; set; } = new
List<InsuranceAgent>();
}

```

Листинг класса *Contract*

```

using System;
using System.Collections.Generic;

namespace lab4.Models;

public partial class Contract
{
    public int Id { get; set; }

    public string Responsibilities { get; set; } = null!;
}

```

```

        public DateTime StartDeadline { get; set; }

        public DateTime EndDeadline { get; set; }

        public virtual ICollection<InsuranceAgent> InsuranceAgents { get; set; } = new
        List<InsuranceAgent>();
    }

```

Листинг класса *InsuranceAgent*

```

namespace lab4.Models;

public partial class InsuranceAgent
{
    public int Id { get; set; }

    public string Name { get; set; } = null!;

    public string Surname { get; set; } = null!;

    public string MiddleName { get; set; } = null!;

    public int AgentType { get; set; }

    public decimal Salary { get; set; }

    public int Contract { get; set; }

    public double TransactionPercent { get; set; }

    public virtual AgentType AgentTypeNavigation { get; set; } = null!;

    public virtual Contract ContractNavigation { get; set; } = null!;
}

```

Листинг представления *ShowTable*

```

@{
    ViewData["Title"] = "Типы агентов";
    @model List<AgentType>;
}

<div class="text-center">
    <table class="table">
        <tr>
            <th>Id</th>
            <th>Название</th>
        </tr>
        @foreach (var agentType in Model) {
            <tr>
                <td>
                    @agentType.Id
                </td>
                <td>
                    @agentType.Type
                </td>
            </tr>
        }
    </table>
</div>

```

Листинг представления *ShowTable*

```

@{
    ViewData["Title"] = "Типы агентов";
    @model List<Contract>;
}

```



```

}

<div class="text-center">
  <table class="table">
    <tr>
      <th>Id</th>
      <th>Начало контракта</th>
      <th>Конец контракта</th>
      <th>Обязанности</th>
    </tr>
    @foreach(var contract in Model) {
      <tr>
        <td>
          @contract.Id
        </td>
        <td>
          @contract.StartDeadline.ToShortDateString()
        </td>
        <td>
          @contract.EndDeadline.ToShortDateString()
        </td>
        <td>
          @contract.Responsibilities
        </td>
      </tr>
    }
  </table>
</div>

```

Листинг представления *ShowTable*

```

@{
  ViewData["Title"] = "Типы агентов";
  @model List<InsuranceAgent>;
}

<div class="text-center">
  <table class="table">
    <tr>
      <th>Id</th>
      <th>Фамилия</th>
      <th>Имя</th>
      <th>Отчество</th>
      <th>Тип агента</th>
      <th>Зарплата</th>
      <th>Начало контракта</th>
      <th>Конец контракта</th>
      <th>Обязанности</th>
      <th>Процент от сделки</th>
    </tr>
    @foreach(var insuranceAgent in Model) {
      <tr>
        <td>
          @insuranceAgent.Id
        </td>
        <td>
          @insuranceAgent.Surname
        </td>
        <td>
          @insuranceAgent.Name
        </td>
        <td>
          @insuranceAgent.MiddleName
        </td>
        <td>
          @insuranceAgent.AgentTypeNavigation.Type

```

```

        </td>
        <td>
            @insuranceAgent.Salary
        </td>
        <td>

@insuranceAgent.ContractNavigation.StartDeadline.ToShortDateString()
        </td>
        <td>

@insuranceAgent.ContractNavigation.EndDeadline.ToShortDateString()
        </td>
        <td>
            @insuranceAgent.ContractNavigation.Responsibilities
        </td>
        <td>
            @insuranceAgent.TransactionPercent
        </td>
    </tr>
}
</table>
</div>

```