**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: **«РПБДИС»**

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

Выполнил: студент гр. ИТП-31

Сетко А.И.

Принял: старший преподаватель

Асенчик О.Д.

Гомель 2019

**Цель**: 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:** cоздать 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 этого 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».

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

**Ход работы**

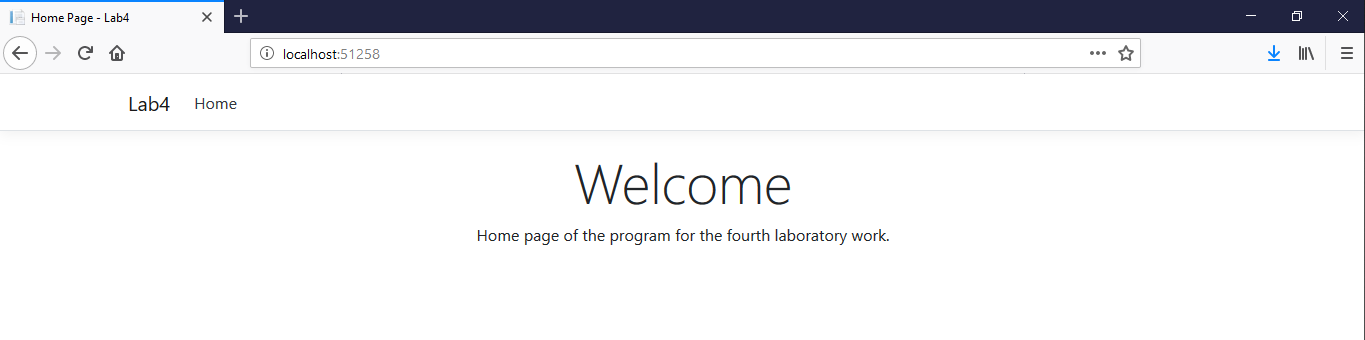
****

Рисунок 1 – Копия экрана браузера домашней страницы

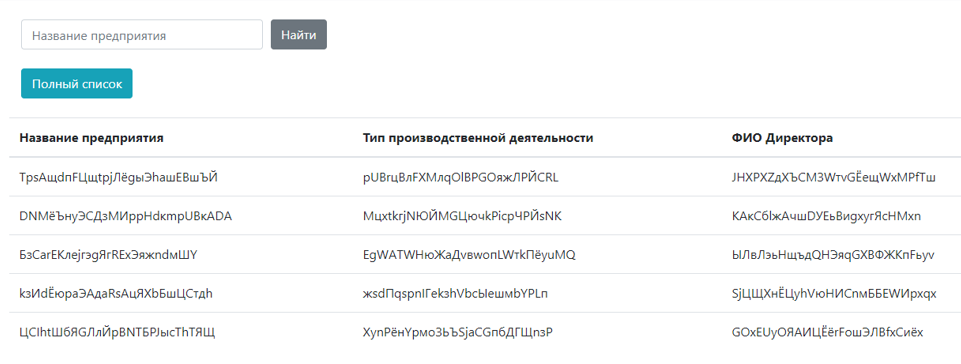


Рисунок 2 – Копия экрана браузера с результатом выполнения запроса с *URL* содержащим /*Organizations*

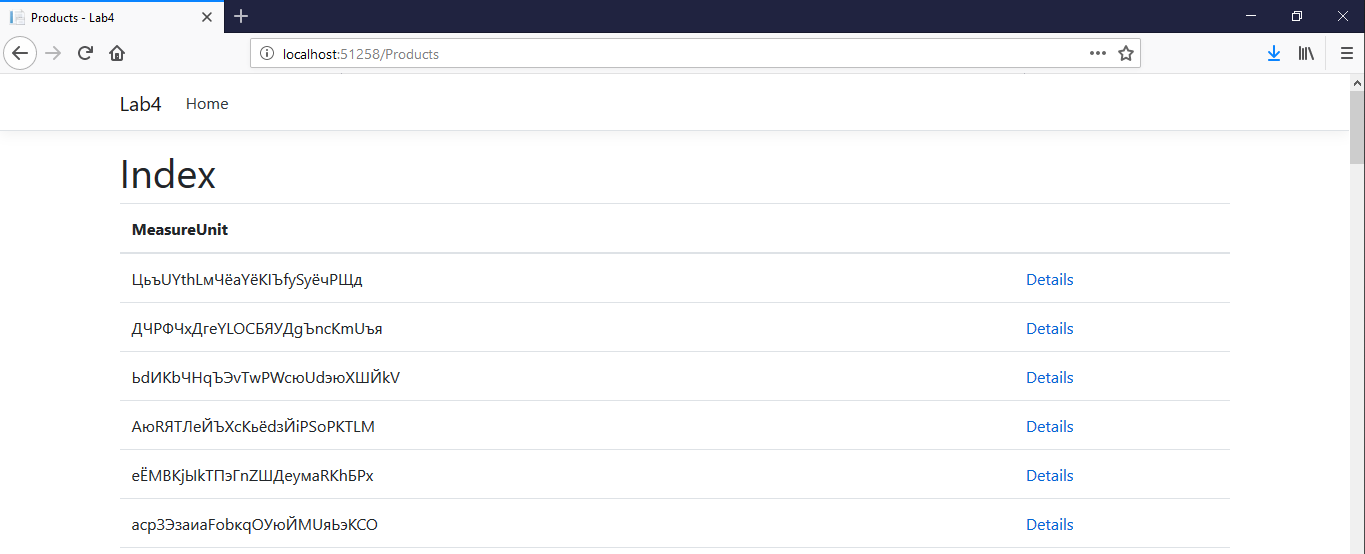


Рисунок 3 – Копия экрана браузера с результатом выполнения запроса с *URL* содержащим /*Products*

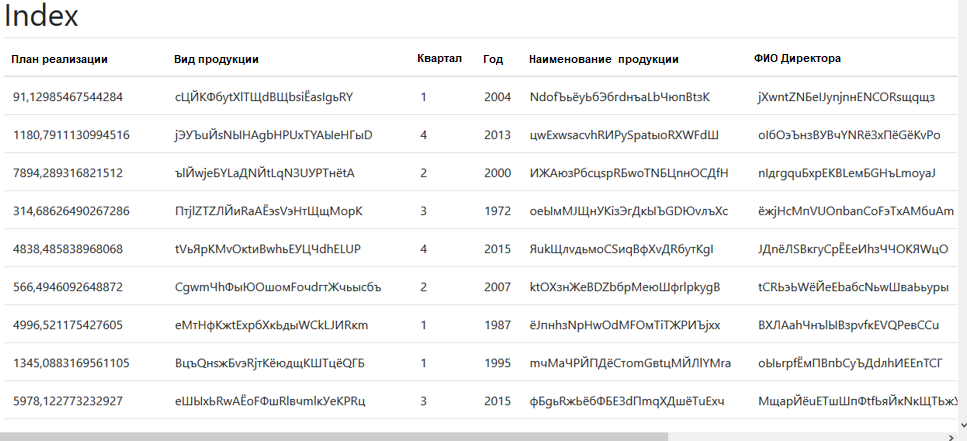


Рисунок 4 – Копия экрана браузера с результатом выполнения запроса с *URL* содержащим /*Realisations*

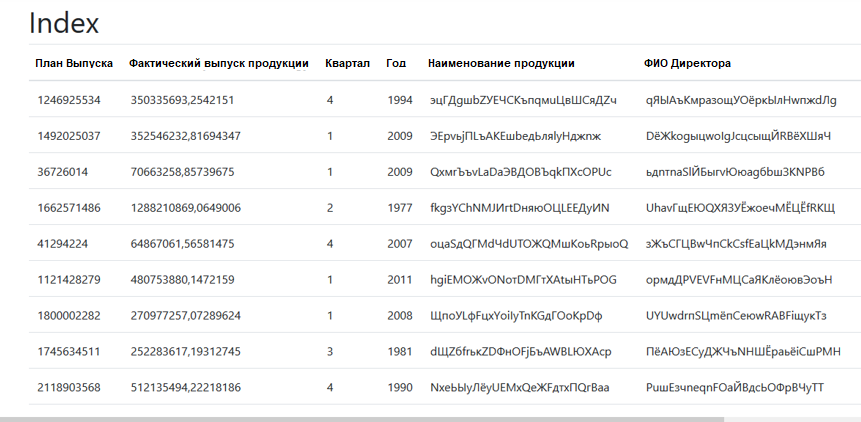


Рисунок 5 – Копия экрана браузера с результатом выполнения запроса с *URL* содержащим /*Outputs*

**Код классов**

**Класс Startup:**

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

public void ConfigureServices(IServiceCollection services)

{

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

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

services.AddControllersWithViews();

}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

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

}

app.UseInitializerMiddleware();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Organizations}/{action=Index}/{id?}");

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Products}/{action=Index}/{id?}");

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Outputs}/{action=Index}/{id?}");

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Realisations}/{action=Index}/{id?}");

});

}

}

**Класс RationingDbContext:**

public partial class RationingDbContext : DbContext

{

public RationingDbContext()

{

}

public RationingDbContext(DbContextOptions<RationingDbContext> options)

: base(options)

{

}

public virtual DbSet<Realisation> Realisations { get; set; }

public virtual DbSet<Organization> Organizations { get; set; }

public virtual DbSet<Output> Outputs { get; set; }

public virtual DbSet<Product> Products { get; set; }

}

**Класс Organization:**

public partial class Organization

{

public Organization()

{

NormalConsumptionRate = new HashSet<NormalConsumptionRate>();

ProductionsAndConsumption = new HashSet<ProductionAndConsumption>();

}

public int Id { get; set; }

public string Name { get; set; }

public string FormOfOwnership { get; set; }

public string ActivityType { get; set; }

public string HeadName { get; set; }

public virtual ICollection<Realisation> Realisation { get; set; }

public virtual ICollection<Output> Output{ get; set; }

}

**Класс Product:**

public partial class Product

{

public Product()

{

Realisation = new HashSet<Realisation>();

Output = new HashSet<Output>();

}

public int Id { get; set; }

public string MeasureUnit { get; set; }

public virtual ICollection<Realisation> Realisation { get; set; }

public virtual ICollection<Output> Output{ get; set; }

}

**Класс Realisation:**

public partial class Realisation

{

public int Id { get; set; }

public int RealisationPlan { get; set; }

public double RealisationFact { get; set; }

public short Quarter { get; set; }

public short Year { get; set; }

public int OrganizationId { get; set; }

public int ProductId { get; set; }

public virtual Organization Organization { get; set; }

public virtual Product Product { get; set; }

}

**Класс Output:**

public partial class Output

{

public int Id { get; set; }

public int OutputPlan { get; set; }

public double OutputFact { get; set; }

public short Quarter { get; set; }

public short Year { get; set; }

public int OrganizationId { get; set; }

public int ProductId { get; set; }

public virtual Organization Organization { get; set; }

public virtual Product Product { get; set; }

}

**Класс DbInitializer:**

public class DbInitializer

{

public static void Initialize(RationingDbContext db)

{

db.Database.EnsureCreated();

if (db.Organizations.Any())

{

return;

}

int onSideOne = 100;

int onSideMany = 1000;

Random random = new Random();

for (int i = 0; i < onSideOne; i++)

{

db.Organizations.Add(new Tables.Organization

{

Name = CreateRandomString(random),

FormOfOwnership = CreateRandomString(random),

ActivityType = CreateRandomString(random),

HeadName = CreateRandomString(random),

});

db.Products.Add(new Tables.Product

{

MeasureUnit = CreateRandomString(random)

});

}

db.SaveChanges();

for (int i = 0; i < onSideMany; i++)

{

int quarter = random.Next(1, 5);

int year = random.Next(1970, 2019);

int organizationId = i % onSideOne + 1;

int productId = i % onSideOne + 1;

db.Realisations.Add(new Tables.Realisation

{

RealisationPlan = random.Next(),

RealisationFact = random.NextDouble() \* random.Next(),

Quarter = (short) quarter,

Year = (short) year,

OrganizationId = organizationId,

ProductId = productId

});

db.Outputs.Add(new Tables.Output

{

OutputPlan = random.Next(),

OutputFact = random.NextDouble() \* random.Next(),

Quarter = (short)quarter,

Year = (short)year,

OrganizationId = organizationId,

ProductId = productId

});

}

db.SaveChanges();

}

private static string CreateRandomString(Random random)

{

char[] letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZАБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯабвгдеёжзиёклмнопрстуфхцчшщъыьэюяabcdefghijklmnopqrstuvwxyz".ToCharArray();

string word = "";

for (int j = 5; j <= 30; j++)

{

int letter\_num = random.Next(0, letters.Length - 1);

word += letters[letter\_num];

}

return word;

}

}

**Класс InitializerMiddleware:**

public class InitializerMiddleware

{

private readonly RequestDelegate \_next;

public InitializerMiddleware(RequestDelegate next)

{

\_next = next;

}

public Task Invoke(HttpContext httpContext, RationingDbContext db)

{

DbInitializer.Initialize(db);

return \_next(httpContext);

}

}

**Класс MiddlewareExtensions:**

public static class MiddlewareExtensions

{

public static IApplicationBuilder UseInitializerMiddleware(this IApplicationBuilder builder)

{

return builder.UseMiddleware<InitializerMiddleware>();

}

}

**Класс HomeController:**

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

public IActionResult Index()

{

return View();

}

public IActionResult Privacy()

{

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

**Класс OrganizationsController:**

public class OrganizationsController : Controller

{

private readonly RationingDbContext \_context;

public OrganizationsController(RationingDbContext context)

{

\_context = context;

}

// GET: Organizations

public async Task<IActionResult> Index()

{

return View(await \_context.Organizations.ToListAsync());

}

// GET: Organizations/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null)

{

return NotFound();

}

var organization = await \_context.Organizations

.FirstOrDefaultAsync(m => m.Id == id);

if (organization == null)

{

return NotFound();

}

return View(organization);

}

}

**Класс ProductsController:**

public class ProductsController : Controller

{

private readonly RationingDbContext \_context;

public ProductsController(RationingDbContext context)

{

\_context = context;

}

// GET: Products

public async Task<IActionResult> Index()

{

return View(await \_context.Products.ToListAsync());

}

// GET: Products/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null)

{

return NotFound();

}

var product = await \_context.Products

.FirstOrDefaultAsync(m => m.Id == id);

if (product == null)

{

return NotFound();

}

return View(product);

}

}

**Класс RealisationsController:**

public class RealisationsController : Controller

{

private readonly RationingDbContext \_context;

public RealisationsController(RationingDbContext context)

{

\_context = context;

}

// GET: Realisations

public async Task<IActionResult> Index()

{

var rationingDbContext = \_context.Realisations.Include(n => n.Organization).Include(n => n.Product);

return View(await rationingDbContext.ToListAsync());

}

// GET: Realisations/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null)

{

return NotFound();

}

var realisation = await \_context.realisation

.Include(n => n.Organization)

.Include(n => n.Product)

.FirstOrDefaultAsync(m => m.Id == id);

if (realisation == null)

{

return NotFound();

}

return View(realisation);

}

}

**Класс OutputsController:**

public class OutputsController : Controller

{

private readonly RationingDbContext \_context;

public OutputsController(RationingDbContext context)

{

\_context = context;

}

// GET: Outputs

public async Task<IActionResult> Index()

{

var rationingDbContext = \_context.Outputs.Include(p => p.Organization).Include(p => p.Product);

return View(await rationingDbContext.ToListAsync());

}

// GET: Outputs/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null)

{

return NotFound();

}

var output = await \_context.Outputs

.Include(p => p.Organization)

.Include(p => p.Product)

.FirstOrDefaultAsync(m => m.Id == id);

if (output == null)

{

return NotFound();

}

return View(output);

}

}

**Вывод**: в результате выполнения лабораторной работы были ознакомлены 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.