Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій

Кафедра програмних систем і технологій

ЗВІТ

з лабораторної роботи № 3

Тема: “Автоматизація бізнес-процесів архітектурного 3Д-дизайну”

Дисципліна «Об’єктно-орієнтоване конструювання програм»

Підготував:
студент гр. ІПЗ-23(1)
Шевченко Андрій Віталійович
Дата : 20.05.22

Перевірила:
Зубик Л.В

Київ – 2022

Тема: Патерни.

Мета роботи: ознайомитися з паттернами проектування.

Умова

Завдання 1: Продумати власний проект (3-й семестр або з ЛР 1-2) так, щоб в коді можливо було застосувати один із розглянутих вище патернів (Стратегія, Спостерігач або Декоратор). Реалізуйте його.

Завдання 2: Окремо розробити проект і застосувати в коді один із розглянутих вище патернів (Стратегія, Спостерігач або Декоратор) для наведених нижче алгоритмів.

Тема проекту: Автоматизація бізнес-процесів архітектурного 3Д-дизайну

Хід роботи:

Завдання 1:

В основі архітектурного шаблону MVC, лежать Model,View,Controller. Для початку реалізації цього шаблону потрібно створити моделі даних, які будуть взаємодіяти з контролерами, та зберігатися в базі даних.

Я створив 3 моделі даних:

- Order - замовлення клієнта

```
public class Order
{

    public int Id { get; set; }

    public string Theme { get; set; }

    public string Description { get; set; }
}
```

- Artist - 3д дизайнер

```
Ссылок: 0
public class Artist
{
    Ссылок: 0
    public int Id { get; set; }
    Ссылок: 0
    public string Name { get; set; }
    Ссылок: 0
    public string LastName { get; set; }
    Ссылок: 0
    public DateTime DateOfBirth { get; set; }

    Ссылок: 0
    public byte Experience { get; set; }
}
```

- FinalOrder - співставлення замовлення та 3д дизайнера на виконання

```
Ссылок: 0
public class FinalOrder
{
    Ссылок: 0
    public int Id { get; set; }
    Ссылок: 0
    public int OrderId { get; set; }
    Ссылок: 0
    public int ArtistId { get; set; }
}
```

Шар доступу до даних (Dal) який взаємодіє з базої данних, в моєму випадку це MSSQL, реалізований на основі Ef core. Я буду використовувати метод Code First.

```
Ссылок: 2
public class ArchitectureDesignDbContext : DbContext
{
    Ссылок: 0
    public ArchitectureDesignDbContext(DbContextOptions<ArchitectureDesignDbContext> options) : base(options)
    {
    }

    Ссылок: 0
    public DbSet<Order> Orders { get; set; }
    Ссылок: 0
    public DbSet<Artist> Artists { get; set; }
    Ссылок: 0
    public DbSet<FinalOrder> FinalOrders { get; set; }

    Ссылок: 0
    protected override void OnModelCreating(ModelBuilder builder)
    {
        base.OnModelCreating(builder);
    }
}
```

Після цього можна приступати до створення логіки застосунку, я буду

використовувати паттерн “Strategy”. Я створив інтерфейс з сигнатурою методів та сервіс який буде реалізовувати даний інтерфейс.

```
Ссылок: 0
public interface IArchitectureRepository
{
    Ссылок: 0
    IEnumerable<Order> GetOrders();
    Ссылок: 0
    IEnumerable<Artist> GetArtists();
    Ссылок: 0
    IEnumerable<FinalOrder> GetAOrder();

    Ссылок: 0
    void AddOrders(Order order);
    Ссылок: 0
    void AddArtists(Artist artist);
    Ссылок: 0
    void AddOrder(FinalOrder order);
}
```

```
Ссылок: 1
public class ArchitectureService : IArchitectureRepository
{
    private readonly ArchitectureDesignDbContext _dbContext;

    Ссылок: 0
    public ArchitectureService(ArchitectureDesignDbContext dbContext)
    {
        _dbContext = dbContext;
    }
```

```
Ссылок: 1
public void AddArtists(Artist artist)
{
    var artistDto = new Artist
    {
        Id = artist.Id,
        Name = artist.Name,
        LastName = artist.LastName,
        DateOfBirth = artist.DateOfBirth,
        Experience = artist.Experience
    };

    _dbContext.Artists.Add(artistDto);
    _dbContext.SaveChanges();
}

Ссылок: 1
public void AddOrder(FinalOrder order)
{
    var ordertDto = new FinalOrder
    {
        Id = order.Id,
        OrderId = order.OrderId,
        ArtistId = order.ArtistId
    };

    _dbContext.FinalOrders.Add(ordertDto);
    _dbContext.SaveChanges();
}
```

```
Ссылок: 1
public void AddOrders(Order order)
{
    var ordertDto = new Order
    {
        Id = order.Id,
        Theme = order.Theme,
        Description = order.Description
    };

    _dbContext.Orders.Add(ordertDto);
    _dbContext.SaveChanges();
}

Ссылок: 1
public IEnumerable<FinalOrder> GetOrder()
{
    return _dbContext.FinalOrders.ToList();
}

Ссылок: 1
public IEnumerable<Artist> GetArtists()
{
    return _dbContext.Artists.ToList();
}

Ссылок: 1
public IEnumerable<Order> GetOrders()
{
    return _dbContext.Orders.ToList();
}
```

Після виконання попередніх пунктів, я створив 3 контролера для кожної з сутностей, відповідно кожна сутність має методи додавання та перегляду.

```
public class BaseController : Controller
{

    public IActionResult Index()
    {
        return View();
    }
}
```

```
Ссылок: 1
public class OrderController : Controller
{
    private readonly IArchitectureRepository _repository;

    Ссылок: 0
    public OrderController(IArchitectureRepository repository)
    {
        _repository = repository;
    }

    Ссылок: 0
    public IActionResult Index()
    {
        return View();
    }

    [HttpGet]
    Ссылок: 0
    public IActionResult GetAllOrders()
    {
        var result = _repository.GetOrders();
        return View(result);
    }

    [HttpGet]
    Ссылок: 0
    public IActionResult AddOrder(int Id)
    {
        ViewBag.Id = Id;
        return View();
    }

    [HttpPost]
    Ссылок: 0
    public string AddOrder(Order order)
    {
        _repository.AddOrders(order);
        return "Order," + order.Id + ",Added";
    }
}
```

```
public class ArtistController : Controller
{
    private readonly IArchitectureRepository _repository;


    public ArtistController(IArchitectureRepository repository)
    {
        _repository = repository;
    }


    public IActionResult Index()
    {
        return View();
    }

    [HttpGet]

    public IActionResult GetAllArtist()
    {
        var result = _repository.GetArtists();
        return View(result);
    }

    [HttpGet]

    public IActionResult AddArtist(int Id)
    {
        ViewBag.Id = Id;
        return View();
    }

    [HttpPost]

    public string AddArtist(Artist artist)
    {
        _repository.AddArtists(artist);
        return "Artist," + artist.Id + ",Added";
    }
}
```

```
public class FinalOrderController : Controller
{
    private readonly IArchitectureRepository _repository;


    public FinalOrderController(IArchitectureRepository repository)
    {
        _repository = repository;
    }


    public IActionResult Index()
    {
        return View();
    }

    [HttpGet]

    public IActionResult GetAllOrders()
    {
        var result = _repository.GetOrder();
        return View(result);
    }

    [HttpGet]

    public IActionResult AddArtist(int Id)
    {
        ViewBag.Id = Id;
        return View();
    }

    [HttpPost]

    public string AddOrder(FinalOrder order)
    {
        _repository.AddOrder(order);
        return "FinalOrder," + order.Id + ",Added";
    }
}
```

Відповідно код для відображення контролерів на прикладі Order:

```
<body>
<div>
    <table>
        <tr>
            <td>GetAllOrders</td>
            <td>AddOrder</td>
        </tr>
        <tr>
            <td>
                <form action="@Url.Action("GetAllOrders", "FinalOrder")" method="get">
                <input type="hidden"/>
                <button type="submit">GetAllOrders</button>
                </form>
            </td>
            <td>
                <form action="@Url.Action("AddOrder", "FinalOrder")" method="get">
                <input type="hidden" name="Id"/>
                <button type="submit">AddOrder</button>
                </form>
            </td>
        </tr>
    </table>
</div>
</body>
```

```
<body>
    <div>
    <table>
        <tr>
            <td>Id</td>
            <td>>OrderId</td>
            <td>ArtistId</td>
        </tr>
        @foreach (var value in Model)
        {
            <tr>
              <td>@value.Id</td>
              <td>@value.OrderId</td>
              <td>@value.ArtistId</td>
            </tr>
        }
    </table>
    </div>
</body>
```

```
<form method="post" action="">
    <input type="hidden" value="@ViewBag.Id" name="Id"/>
    <table>
        <tr>
            <td><p>Enter Id</p></td>
            <td><input type="text" name="Id"/></td>
        </tr>
        <tr>
            <td><p>Enter OrderId: </p></td>
            <td><input type="text" name="OrderId"/></td>
        </tr>
        <tr>
            <td><p>Enter ArtistId: </p></td>
            <td><input type="text" name="ArtistId"/></td>
        </tr>
        <tr>
            <td><input type="submit" value="Add Order"/></td>
        </tr>
    </table>
</form>
```

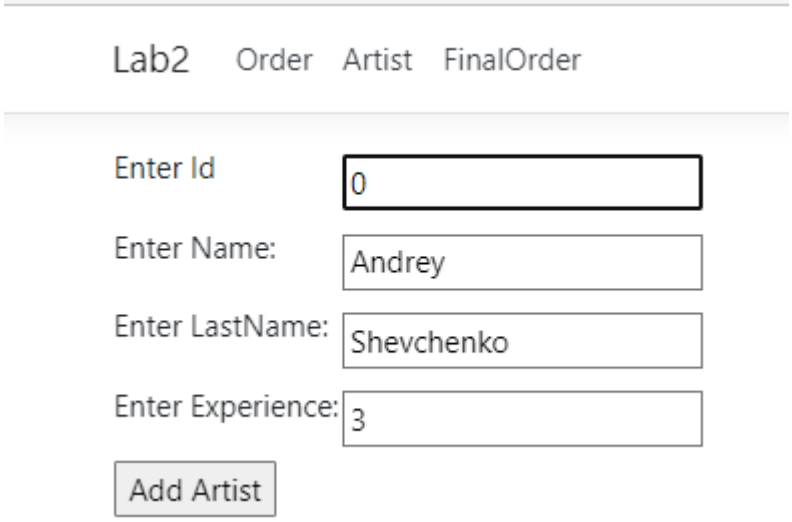
Скріншот результатів:

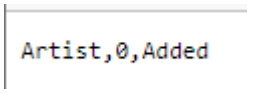
Lab2 Order Artist FinalOrder

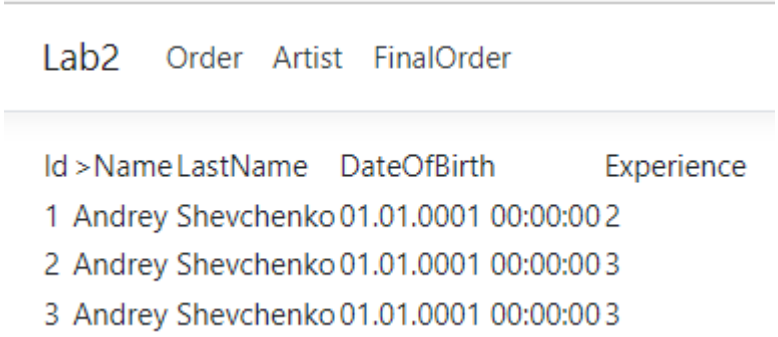
Lab2 Order Artist FinalOrder

GetAllArtist AddArtist

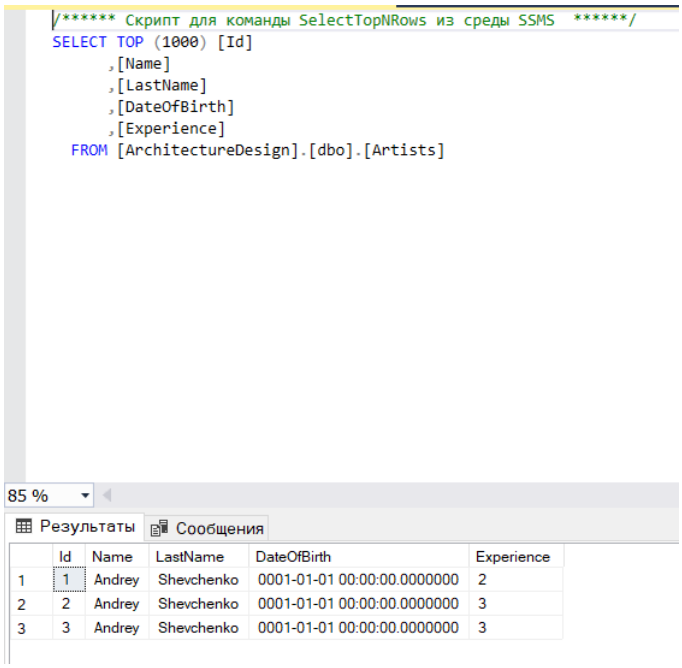
GetAllArtist | AddArtist







Перевірка БД:



```
/****** Скрипт для команды SelectTopNRows из среды SSMS  ******/
SELECT TOP (1000) [Id]
      ,[Name]
      ,[LastName]
      ,[DateOfBirth]
      ,[Experience]
  FROM [ArchitectureDesign].[dbo].[Artists]
```

85 %

Результаты | Сообщения

	Id	Name	LastName	DateOfBirth	Experience
1	1	Andrey	Shevchenko	0001-01-01 00:00:00.0000000	2
2	2	Andrey	Shevchenko	0001-01-01 00:00:00.0000000	3
3	3	Andrey	Shevchenko	0001-01-01 00:00:00.0000000	3

Завдання 2:

Сортування (символи, рядки, числа тощо);

Для реалізації цього завдання я буду використовувати паттерн, Strategy, тому що він доволі легкий в освоєнні в порівнянні з іншими, також він ідеально підходить для реалізації цього завдання.

Для початку я створив 2 інтерфейса та сервіси які будуть їх реалізовувати.

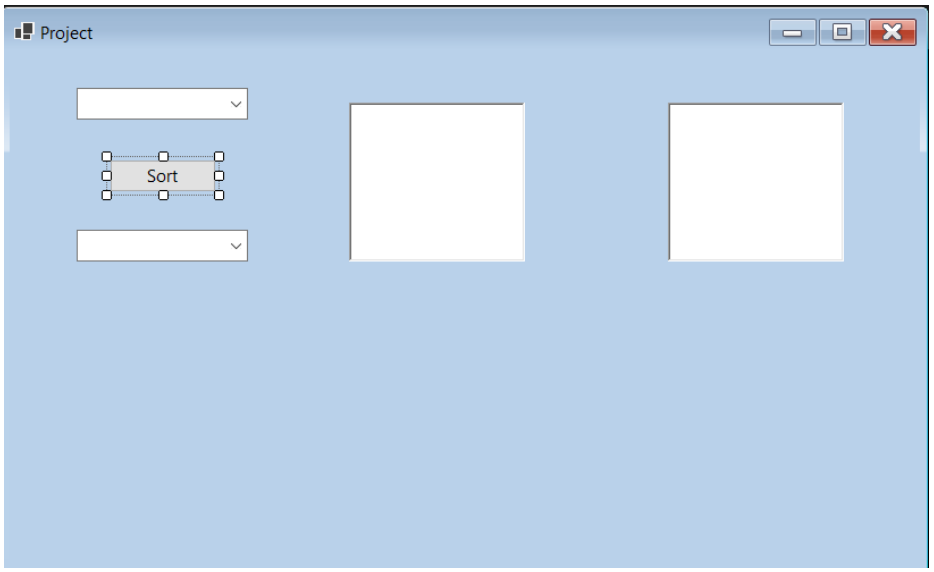
```
public interface INumbersRepository
{

    int[] SortByAscending();

    int[] SortByDescending();
}
```

```
Ссылок: 1
public interface IStringRepository
{
    Ссылок: 2
    string[] SortByAscending();
    Ссылок: 2
    string[] SortByDescending();
    Ссылок: 2
    string SortByAscii();
}
```

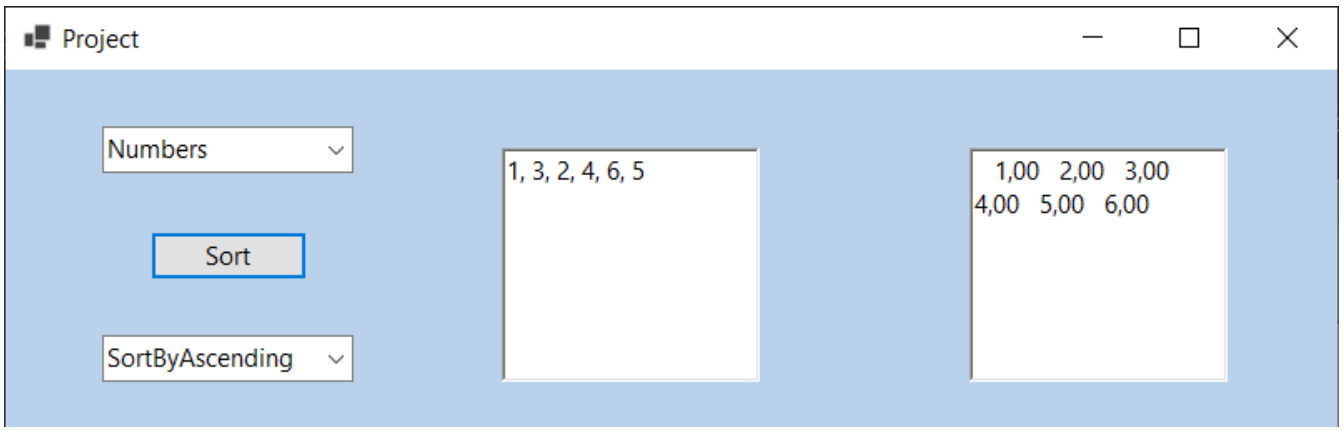
```
public class NumberService : INumbersRepository
{
    int[] numbers = { 1, 3, 2, 4, 6, 5 };


    public int[] SortByAscending()
    {
        Array.Sort(numbers);
        return numbers;
    }


    public int[] SortByDescending()
    {
        Array.Sort(numbers);
        Array.Reverse(numbers);
        return numbers;
    }
}
```

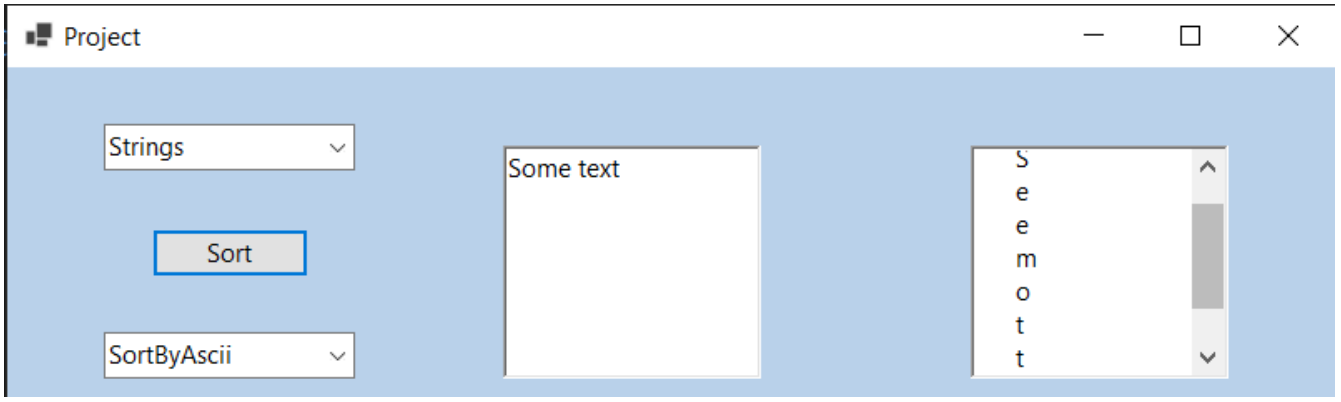
```
Ссылок: 2
public class StringService : IStringRepository
{
    string[] people = { "Tom", "Sam", "Andrey"};

    Ссылок: 2
    public string[] SortByAscending()
    {
        Array.Sort(people);
        return people;
    }

    Ссылок: 2
    public string SortByAscii()
    {
        string text = "Some text";
        char[] chars = text.ToCharArray();
        Array.Sort(chars);
        text = new string(chars);
        return text;
    }

    Ссылок: 2
    public string[] SortByDescending()
    {
        Array.Sort(people);
        Array.Reverse(people);
        return people;
    }
}
```

Так як це десктопний застосунок на основі winforms, я розробив базовий інтерфейс для сортування чисел та строк.



Кінцевий результат:





Висновки:

Під час виконання цією лабораторної роботи я ознайомився з патернами проектування, які входять в Gof, та допомагають створювати більш гнучкі та великі системи. Для першого завдання я розробив веб-застосунок за допомогою Asp net Core Mvc згідно своєї теми. Застосував ORM Ef core для роботи з БД, та паттерн Strategy. Для 2 завдання я розробив десктопний застосунок за допомогою Winforms.