**Основи об'єктно-орієнтованого програмування**

**Лабораторна робота № 2**

**Проектування та розробка програм з використанням патернів проектування**

Метою виконання лабораторної роботи №2 є вивчення патернів проектування, зокрема Команда (Command), Адаптер (Adapter), Спостерігач (Strategy), Ланцюг обов’язків (Chain of Responsilibity), Стратегія (Strategy).

Хід виконання:

Створюється загальний проект під назвою Lab2 на мові програмування C# в середовищі Visual Studio 2019 за технологією ASP NET Core MVC .NET 5.

Створений проект має 6 окремих програмних модулів.

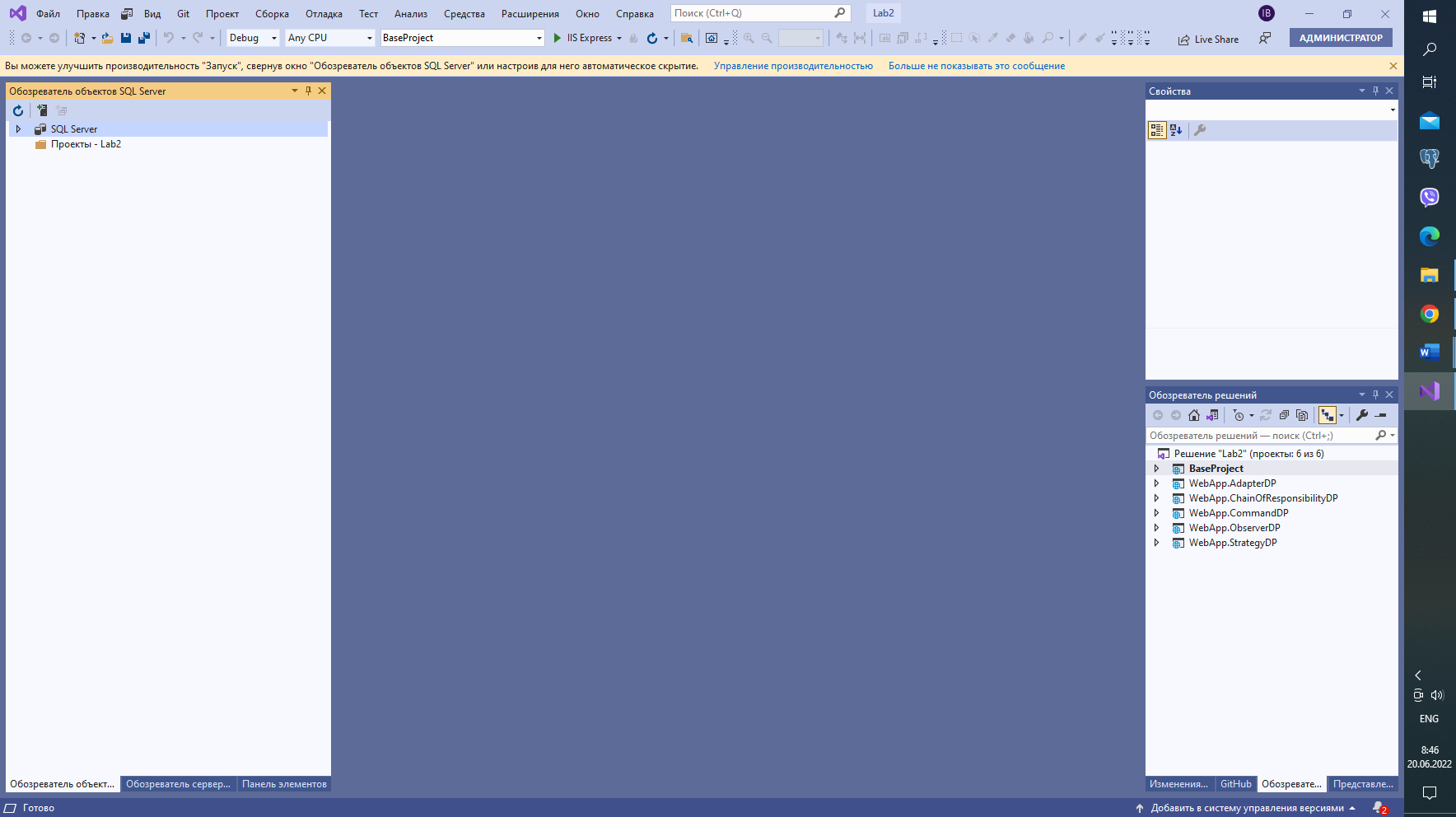


Рис 1. Скріншот загального виду проекту Lab2.

Кожний модуль відповідає реалізації окремого типу патерна. Програмний модуль BaseProject є базовим для виконання.

Загальні шаблони будуть реалізовані на BaseProject. Більше того, в BaseProject, який був створений з обережністю з точки зору принципу інверсії керування та інверсії залежностей.

Кожний модуль згідно технології MVC має такі папки, як Models (моделі бази даних), Controllers (контролери) та Views ( представлення).

Патерн 1- Стратегія реалізується в модулі WebApp.StrategyDP.

Лістинг класів:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using WebApp.StrategyDP.Models;

namespace WebApp.StrategyDP.Repositories

{

public interface IProductRepository

{

Task<Product> GetById(string id);

Task<List<Product>> GetAllByUserId(string userId);

Task<Product> Save(Product product);

Task Update(Product product);

Task Delete(Product product);

}

}

using BaseProject.Models;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using WebApp.StrategyDP.Models;

namespace WebApp.StrategyDP.Repositories

{

public class ProductRepositoryFromSqlServer : IProductRepository

{

private readonly AppIdentityDbContext \_context;

public ProductRepositoryFromSqlServer(AppIdentityDbContext appIdentityDbContext)

{

\_context = appIdentityDbContext;

}

public async Task Delete(Product product)

{

\_context.Products.Remove(product);

await \_context.SaveChangesAsync();

}

public async Task<List<Product>> GetAllByUserId(string userId)

{

return await \_context.Products.Where(x => x.Id == userId).ToListAsync();

}

public async Task<Product> GetById(string id)

{

return await \_context.Products.FindAsync(id);

}

public async Task<Product> Save(Product product)

{

product.Id = Guid.NewGuid().ToString();

await \_context.Products.AddAsync(product);

await \_context.SaveChangesAsync();

return product;

}

public async Task Update(Product product)

{

\_context.Products.Update(product);

await \_context.SaveChangesAsync();

}

}

}

Основна мета шаблону стратегічного проектування: цей шаблон проектування дозволяє нам змінювати поведінку об'єкта під час виконання.

У цьому сценарії ми маємо дві бази даних для використання, і ми даємо можливість користувачеві змінити, яку базу даних хоче. Я використовую дві бази даних. Одна з них — SqlServer, інша — MongoDb. Використовуючи шаблон розробки стратегії, вам не потрібно змінювати базу даних під час компіляції, ви можете легко змінити базу даних під час виконання за допомогою динамічного керування.

Вимоги:

MongoDbAtlas

SqlServer

Ядро EntityFramework

Основний ідентифікатор EntityFramework

EntityFramework Core SqlServer

Основні інструменти EntityFramework.

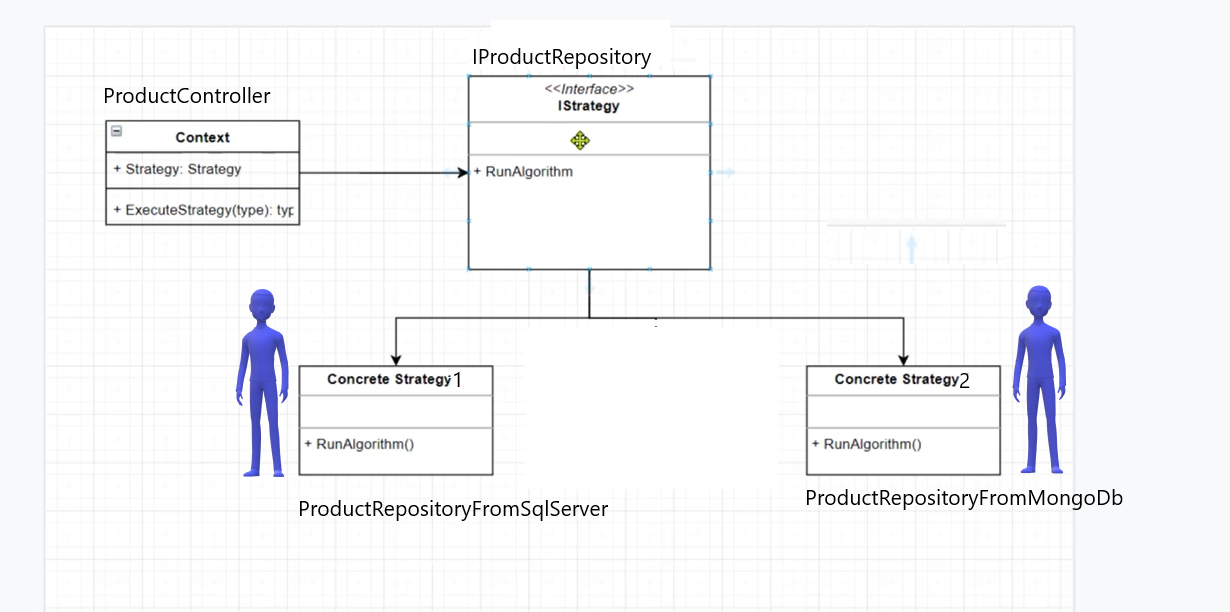


Рис. 2. Діаграма класів.

Шаблон дизайну 2 Ланцюжок обов’язків реалізується в модулі WebApp.ChainofResponsibilityDP.

Лістинг класів:

using ClosedXML.Excel;

using System;

using System.Collections.Generic;

using System.Data;

using System.IO;

using System.Linq;

using System.Threading.Tasks;

namespace WebApp.ChainOfResponsibilityDP.ChainOfResponsibilityDP

{

public class ExcellProcessHandler<T> :ProcessHandler

{

private DataTable GetTable(object o)

{

var table = new DataTable();

var type = typeof(T);

type.GetProperties().ToList().ForEach(x => table.Columns.Add(x.Name, x.PropertyType));

var list = o as List<T>;

list.ForEach(x =>

{

var values = type.GetProperties().Select(propertyInfo => propertyInfo.GetValue(x, null)).ToArray();

table.Rows.Add(values);

});

return table;

}

public override object handle(object o)

{

var wb = new XLWorkbook();

var ds = new DataSet();

ds.Tables.Add(GetTable(o));

wb.Worksheets.Add(ds);

var excellMemoryStream = new MemoryStream();

wb.SaveAs(excellMemoryStream);

return base.handle(excellMemoryStream);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace WebApp.ChainOfResponsibilityDP.ChainOfResponsibilityDP

{

public interface IProcessHandler

{

IProcessHandler SetNext(IProcessHandler processHandler);

Object handle(Object o);

}

} using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace WebApp.ChainOfResponsibilityDP.ChainOfResponsibilityDP

{

public abstract class ProcessHandler : IProcessHandler

{

private IProcessHandler nextProcessHandler;

public virtual object handle(object o)

{

if (nextProcessHandler != null)

{

return nextProcessHandler.handle(o);

}

return null;

}

public IProcessHandler SetNext(IProcessHandler processHandler)

{

nextProcessHandler=processHandler;

return nextProcessHandler;

}

}

}

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Net;

using System.Net.Mail;

using System.Net.Mime;

using System.Threading.Tasks;

namespace WebApp.ChainOfResponsibilityDP.ChainOfResponsibilityDP

{

public class SendEmailProcessHandler:ProcessHandler

{

private readonly string \_fileName;

private readonly string \_toEmail;

public SendEmailProcessHandler(string fileName, string toEmail)

{

\_fileName = fileName;

\_toEmail = toEmail;

}

public override object handle(object o)

{

var zipMemoryStream = o as MemoryStream;

zipMemoryStream.Position = 0;

var mailMessage = new MailMessage();

var spmtClient = new SmtpClient("srvm11.trww.com");

mailMessage.From = new MailAddress("deneme@kariyersistem.com");

mailMessage.To.Add(new MailAddress(\_toEmail));

mailMessage.Subject = "Zip dosyası";

mailMessage.Body = "<p>Zip Dosyası ektedir. </p>";

Attachment attachment = new Attachment(zipMemoryStream, \_fileName, MediaTypeNames.Application.Zip);

mailMessage.Attachments.Add(attachment);

mailMessage.IsBodyHtml = true;

spmtClient.Port = 587;

spmtClient.Credentials = new NetworkCredential("deneme@kariyersistem.com", "Password12+");

spmtClient.Send(mailMessage);

return base.handle(null);// There is no , next chain that's why u can set to null.

}

}

} using System;

using System.Collections.Generic;

using System.IO;

using System.IO.Compression;

using System.Linq;

using System.Threading.Tasks;

namespace WebApp.ChainOfResponsibilityDP.ChainOfResponsibilityDP

{

public class ZipFileProcessHandler<T> : ProcessHandler

{

public override object handle(object o)

{

var excelMemoryStream = o as MemoryStream;

excelMemoryStream.Position = 0;

using (var zipstream = new MemoryStream())

{

using (var archive = new ZipArchive(zipstream, ZipArchiveMode.Create,true))

{

var zipFile = archive.CreateEntry($"{typeof(T).Name}.xlsx");

using (var zipEntry = zipFile.Open())

{

excelMemoryStream.CopyTo(zipEntry);

}

}

return base.handle(zipstream);

}

}

}

}

Основний мета шаблону проектування ChainofResponsibility: це проектування шаблону дає нам можливість зв’язати весь наш процес, крім того, розділити цей процес на крок за кроком. Таким чином, коли ви хочете додати нову частину або нову поведінку до ваших програм, ви можете легко додати новий ланцюг у додаток до цього кроку, щоб змінити свій режим.

У цьому сценарії ми маємо 3 кроки. Це означає, що наша програма має 3 ланцюжки:

Відмінний процес

Процес Zip

Процес електронної пошти

Вимоги:

Закритий XML

SqlServer

Ядро EntityFramework

Основний ідентифікатор EntityFramework

EntityFramework Core SqlServer

Основні інструменти EntityFramework

Шаблон дизайну ланцюжка відповідальності

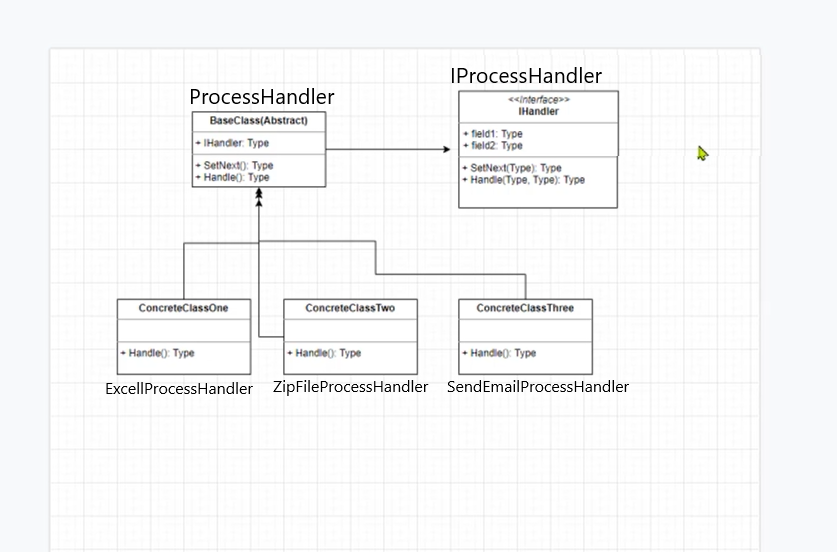


Рис. 3. Діаграма класів.

Шаблон проектування 3 - Команда реалізується в модулі WebApp.CommandDP.

Лістинг класів:

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace WebApp.CommandDP.Commands

{

public class CreateExcellTableActionCommand<T> : ITableButtonActionCommand

{

private readonly ExcellFile<T> \_excellFile;

public CreateExcellTableActionCommand(ExcellFile<T> excellFile)

{

\_excellFile = excellFile;

}

public IActionResult Execute()

{

var excellMemoryStream = \_excellFile.Create();

return new FileContentResult(excellMemoryStream.ToArray(), \_excellFile.FileType)

{

FileDownloadName = \_excellFile.FileName

};

}

}

}

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace WebApp.CommandDP.Commands

{

public class CreatePdfTableActionCommand<T> : ITableButtonActionCommand

{

private readonly PdfFile<T> \_pdfFile;

public CreatePdfTableActionCommand(PdfFile<T> pdfFile)

{

\_pdfFile = pdfFile;

}

public IActionResult Execute()

{

var pdfMemoryStream = \_pdfFile.Create();

return new FileContentResult(pdfMemoryStream.ToArray(), \_pdfFile.FileType)

{

FileDownloadName = \_pdfFile.FileName

};

}

}

}

using ClosedXML.Excel;

using System;

using System.Collections.Generic;

using System.Data;

using System.IO;

using System.Linq;

using System.Threading.Tasks;

namespace WebApp.CommandDP.Commands

{

public class ExcellFile<T>

{

public readonly List<T> \_list;

public string FileName => $"{typeof(T).Name}.xlsx";

public string FileType => "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet";

public ExcellFile(List<T> list)

{

\_list = list;

}

public MemoryStream Create()

{

var wb = new XLWorkbook();

var ds = new DataSet();

ds.Tables.Add(GetTable());

wb.Worksheets.Add(ds);

var excellMemory = new MemoryStream();

wb.SaveAs(excellMemory);

return excellMemory;

}

private DataTable GetTable ()

{

var tablo = new DataTable();

var type = typeof(T);

type.GetProperties().ToList().ForEach(x => tablo.Columns.Add(x.Name, x.PropertyType));

\_list.ForEach(

x =>

{

var values = type.GetProperties().Select(propertyInfo => propertyInfo.GetValue(x, null)).ToArray();

tablo.Rows.Add(values);

});

return tablo;

}

}

}

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace WebApp.CommandDP.Commands

{

public class FileCreateInvoker

{

private ITableButtonActionCommand \_tableButtonActionCommand;

private List<ITableButtonActionCommand> tables = new List<ITableButtonActionCommand>();

public void SetCommand(ITableButtonActionCommand tableButtonActionCommand)

{

\_tableButtonActionCommand = tableButtonActionCommand;

}

public void AddCommand(ITableButtonActionCommand tableButtonActionCommand)

{

tables.Add(tableButtonActionCommand);

}

public IActionResult CreateFile()

{

return \_tableButtonActionCommand.Execute();

}

public List<IActionResult> CreateFiles()

{

return tables.Select(x => x.Execute()).ToList();

}

}

} using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace WebApp.CommandDP.Commands

{

public interface ITableButtonActionCommand

{

IActionResult Execute();

}

}

using DinkToPdf;

using DinkToPdf.Contracts;

using Microsoft.AspNetCore.Http;

using Microsoft.Extensions.DependencyInjection;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WebApp.CommandDP.Commands

{

public class PdfFile<T>

{

public readonly List<T> \_list;

public readonly HttpContext \_context;

public PdfFile(List<T> list, HttpContext context)

{

\_list = list;

\_context = context;

}

public string FileName => $"{typeof(T).Name}.pdf";

public string FileType => "application/octet-stream";

public MemoryStream Create()

{

var type = typeof(T);

var sb = new StringBuilder();

sb.Append($@"<html>

<head></head>

<body>

<div class ='text-center'><h1>{type.Name} tablo</h1></div>

<table class = 'table table-striped' align = 'center'>");

sb.Append("<tr>");

type.GetProperties().ToList().ForEach(x =>

{

sb.Append($"<th>{x.Name}</th>");

});

sb.Append("</tr>");

\_list.ForEach(x =>

{

var values = type.GetProperties().Select(propertyInfo => propertyInfo.GetValue(x, null)).ToList();

sb.Append("<tr>");

values.ForEach(values =>

{

sb.Append($"<td>{values}</td>");

});

sb.Append("</tr>");

});

sb.Append("</table></body></html>");

var doc = new HtmlToPdfDocument()

{

GlobalSettings = {

ColorMode = ColorMode.Color,

Orientation = Orientation.Portrait,

PaperSize = PaperKind.A4,

},

Objects = {

new ObjectSettings() {

PagesCount = true,

HtmlContent =sb.ToString(),

WebSettings = { DefaultEncoding = "utf-8" , UserStyleSheet=Path.Combine(Directory.GetCurrentDirectory(),"wwwroot/lib/bootstrap/dist/css/bootstrap.css)")},

HeaderSettings = { FontSize = 9, Right = "Page [page] of [toPage]", Line = true, Spacing = 2.812 }

}

}

};

var converter = \_context.RequestServices.GetRequiredService<IConverter>();

MemoryStream pdfMemoryStream = new MemoryStream();

return new(converter.Convert(doc));

}

}

}

Основний мета шаблону команди проекту: цей шаблон проектування дає нам можливість поєднати всі наші методи. Якщо ви хочете змінити, ви можете легко дістатися до свого інвікера команди, крім того, це дає шанс на втрато пов'язаних класів.

У цьому сценарії ми маємо 3 кроки. Це означає, що наша програма має 3 команди:

Відмінний процес

Процес PDF

Процес Excell&Pdf + Zip

Вимоги:

DinktoPDF

Закритий XML

SqlServer

Ядро EntityFramework

Основний ідентифікатор EntityFramework

EntityFramework Core SqlServer

Основні інструменти EntityFramework

Шаблон проектування команд

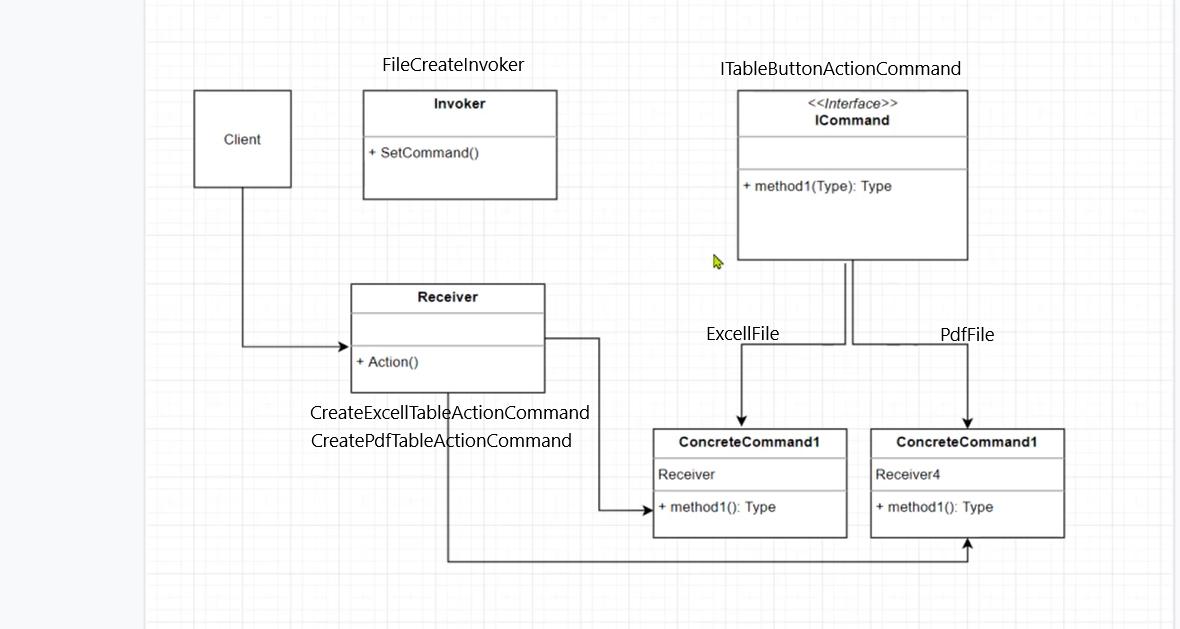


Рис. 4. Діаграма класів.

Шаблон дизайну 4 – Cпостерігач реалізується в модулі WebApp.ObserverDP.

Лістинг класів:

using BaseProject.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace WebApp.ObserverDP.Observer

{

public interface IUserObserver

{

void CreateUser(AppUser appUser);

}

}

using BaseProject.Models;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Logging;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace WebApp.ObserverDP.Observer

{

public class UserObserverCreateDiscount : IUserObserver

{

private readonly IServiceProvider \_serviceProvider;

public UserObserverCreateDiscount(IServiceProvider serviceProvider)

{

\_serviceProvider = serviceProvider;

}

public void CreateUser(AppUser appUser)

{

var logger = \_serviceProvider.GetRequiredService<ILogger<UserObserverCreateDiscount>>();

using var scope = \_serviceProvider.CreateScope();

var context = scope.ServiceProvider.GetRequiredService<AppIdentityDbContext>();

context.Discounts.Add(new Models.Discount { Rate = 30, UserId = appUser.Id });

context.SaveChanges();

logger.LogInformation("Discount row created");

}

}

}

using BaseProject.Models;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Logging;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Mail;

using System.Threading.Tasks;

namespace WebApp.ObserverDP.Observer

{

public class UserObserverSendEmail : IUserObserver

{

private readonly IServiceProvider \_serviceProvider;

public UserObserverSendEmail(IServiceProvider serviceProvider)

{

\_serviceProvider = serviceProvider;

}

public void CreateUser(AppUser appUser)

{

var logger = \_serviceProvider.GetRequiredService<ILogger<UserObserverSendEmail>>();

var mailMessage = new MailMessage();

var smtpClient = new SmtpClient("Sunucu bilgisi");

mailMessage.From = new MailAddress("Gönderilecek E-mail girilecek");

mailMessage.To.Add(new MailAddress(appUser.Email));

mailMessage.Subject = "Wellcome to hasanwebsite.com.tr.";

mailMessage.Body = "<p>General rule of this website ; ..... </p>";

mailMessage.IsBodyHtml = true;

smtpClient.Port = 587;

smtpClient.Credentials = new NetworkCredential("UserName", "Password");

smtpClient.Send(mailMessage);

logger.LogInformation($"E-mail has been sent to user : { appUser.UserName}");

using BaseProject.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace WebApp.ObserverDP.Observer

{

public class UserObserverSubject

{

private readonly List<IUserObserver> \_userObservers;

public UserObserverSubject()

{

\_userObservers = new List<IUserObserver>();

}

public void RegisterObserver(IUserObserver userObserver)

{

\_userObservers.Add(userObserver);

}

public void RemoveObserver(IUserObserver userObserver)

{

\_userObservers.Remove(userObserver);

}

public void NotifyObserver(AppUser appUser)

{

\_userObservers.ForEach(x =>

{

x.CreateUser(appUser);

});

}

}

} using BaseProject.Models;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Logging;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace WebApp.ObserverDP.Observer

{

public class UserObserverWriteToConsole : IUserObserver

{

private readonly IServiceProvider \_serviceProvider;

public UserObserverWriteToConsole(IServiceProvider serviceProvider)

{

\_serviceProvider = serviceProvider;

}

public void CreateUser(AppUser appUser)

{

var logger = \_serviceProvider.GetRequiredService<ILogger<UserObserverWriteToConsole>>();

logger.LogInformation($"user created : Id = {appUser.Id}");

}

}

}

Основне проектування мета - шаблон Observer: Цей шаблон проектування дає нам змогу спостерігати за об’єктами з інших об’єктів, які пов’язані з ними. Observer DP надає можливість бездоганно пов’язаних класів із точки зору модулів високого рівня до модулів низького рівня. Більше того, це забезпечує світло написання кодів пролити на принципі SOLiD.

У цьому сценарії ми маємо 3 кроки. Коли новий користувач реєструє нашу програму:

Увійти в консоль;

Створити знижку для нового користувача;

Автоматично надсилати електронний лист. " Вітаємо, сайт www.default.com "

Вимоги:

SqlServer

Ядро EntityFramework

Основний ідентифікатор EntityFramework

EntityFramework Core SqlServer

Основні інструменти EntityFramework

Шаблон дизайну спостерігача

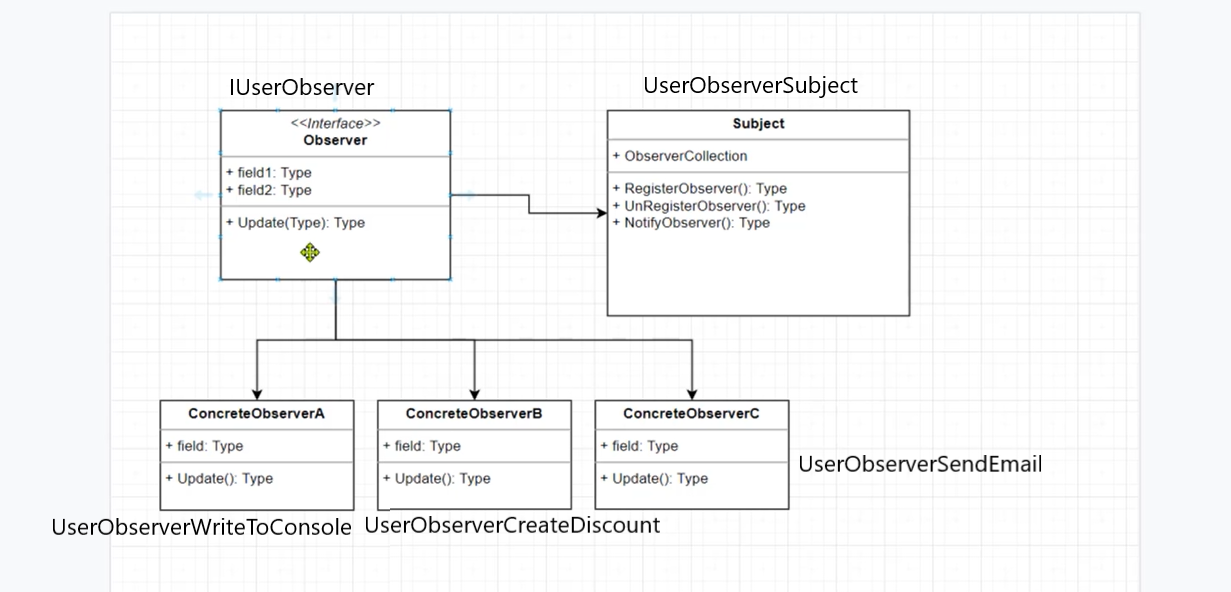


Рис. 5. Діаграма класів.

Шаблон дизайну 5 Aдаптер - реалізується в модулі WebApp.AdapterDP.

Лістинг класів:

using LazZiya.ImageResize;

using System;

using System.Collections.Generic;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Threading.Tasks;

namespace WebApp.AdapterDP.Services

{

public class AdvanceImageProcess : IAdvanceImageProcess

{

public void AddWaterMarkImage(Stream stream, string text, string filePath, Color color, Color outlineColor)

{

using (var img = Image.FromStream(stream))

{

var tOps = new TextWatermarkOptions

{

// Change text color and opacity

// Text opacity range depends on Color's alpha channel (0 - 255)

TextColor = color,

// Add text outline

// Outline color opacity range depends on Color's alpha channel (0 - 255)

OutlineColor = outlineColor

};

img.AddTextWatermark(text, tOps)

.SaveAs(filePath);

}

}

}

}

using System;

using System.Collections.Generic;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Threading.Tasks;

namespace WebApp.AdapterDP.Services

{

public interface IAdvanceImageProcess

{

//void AddWaterMark(string text, string fileName, Stream imageStream); These codes written manually,

void AddWaterMarkImage(Stream stream, string text, string filePath, Color color, Color outlineColor);

}

}

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Threading.Tasks;

namespace WebApp.AdapterDP.Services

{

public interface IImageProcess

{

void AddWaterMark(string text, string fileName, Stream imageStream);

}

}

using System;

using System.Collections.Generic;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Threading.Tasks;

namespace WebApp.AdapterDP.Services

{

public class ImageProcess : IImageProcess

{

public void AddWaterMark(string text, string fileName, Stream imageStream)

{

using var img = Image.FromStream(imageStream);

using var graphic = Graphics.FromImage(img);

var font = new Font(FontFamily.GenericMonospace, 40, FontStyle.Bold, GraphicsUnit.Pixel);

var textSize = graphic.MeasureString(text, font);

var color = Color.FromArgb(128, 255, 255, 255);

var brush = new SolidBrush(color);

var position = new Point(img.Width - ((int)textSize.Width + 30), img.Height - ((int)textSize.Height + 30));

graphic.DrawString(text, font, brush, position);

img.Save("wwwroot/watermarks/" + fileName);

img.Dispose();

graphic.Dispose();

}

}

} using System;

using System.Collections.Generic;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Threading.Tasks;

namespace WebApp.AdapterDP.Services

{

public class ImageProcessAdapter : IImageProcess

{

private readonly IAdvanceImageProcess \_advanceImageProcess;

public ImageProcessAdapter(IAdvanceImageProcess advanceImageProcess)

{

\_advanceImageProcess = advanceImageProcess;

}

public void AddWaterMark(string text, string fileName, Stream imageStream)

{

\_advanceImageProcess.AddWaterMarkImage(imageStream, text, $"wwwroot/watermarks/{fileName}", Color.FromArgb(128, 255, 255, 255), Color.FromArgb(0, 355, 355, 355));

}

}

}

Основний мета-шаблон дизайну адаптера: дизайн адаптера шаблону дуже розширений із точки зору використання; він дуже корисний. Ця важка програма була розгорнута з цим шаблоном, ви можете змінити будь-який інтерфейс без торкання загальної структури. Загальний приклад для AdapterDp — електрична розетка. .Коли ви не можете використовувати цей самий вхідний сокет, ви можете додати новий, який дає тип обміну вашого сокету на відповідний.

У цій сценарії у нас є 2 кроки. Наша програма поставить водяний знак на наше зображення, яке ми збережемо.

Водяний знак до зображення з кодом Manuel;

Водяний знак до зображення за допомогою інструмента Thirdpart;

Використовуйте адаптер для цього процесу

Вимоги:

DrawingCommon

Третя частина зміни розміру зображення

Ядро EntityFramework

Основний ідентифікатор EntityFramework

EntityFramework Core SqlServer

Основні інструменти EntityFramework

Шаблон дизайну Adepter

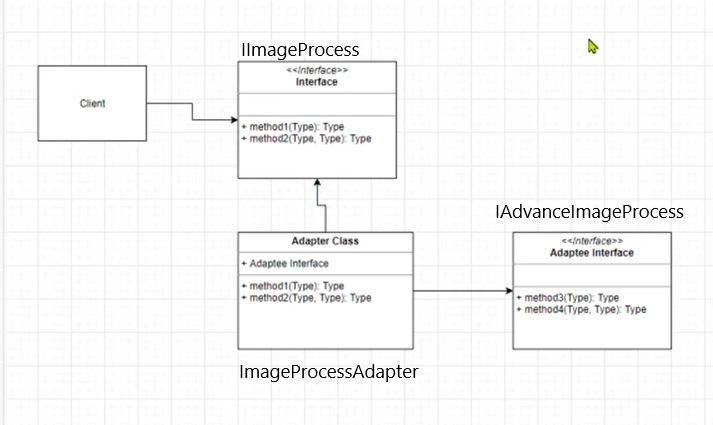


Рис. 6. Діаграма класів.