Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №5

з дисципліни <<Технології розробки програмного забезпечення>>

Тема <<Шаблони «ADAPTER», «BUILDER», «COMMAND», «CHAIN OF RESPONSIBILITY», «PROTOTYPE»>>

Виконав:
студент ІА-23
Содолиснький Вадим

Перевірив:
Мягкий М. Ю.

Київ 2024

Тема: Шаблони «ADAPTER», «BUILDER», «COMMAND», «CHAIN OF RESPONSIBILITY», «PROTOTYPE».

Хід роботи

Варіант №13

Office communicator

Мережевий комунікатор для офісу повинен нагадувати функціонал програми Skype з можливостями голосового / відео / конференц-зв'язку, відправки текстових повідомлень і файлів (можливо, оффлайн), веденням організованого списку груп / контактів.

Патерн Builder

BUILDER є шаблоном, який дозволяє поступово створювати складні об'єкти з багатьма параметрами. Він розділяє процес створення об'єкта на кілька кроків, забезпечуючи можливість збирати об'єкти з різними конфігураціями. ей шаблон особливо корисний, коли необхідно створити об'єкт із багатьма обов'язковими та необов'язковими параметрами.

Патерн Command

COMMAND інкапсулює запит як об'єкт, дозволяючи зберігати, чергувати або скасовувати виконання команд. Цей шаблон дозволяє передавати операції як параметри, забезпечуючи незалежність від клієнтської логіки.

Патерн Chain of Responsibility

CHAIN OF RESPONSIBILITY дозволяє організувати ланцюг обробників, через який проходить запит. Цей шаблон корисний, коли обробка запиту може залежати від кількох об’єктів, наприклад, у реалізації системи обробки подій, розподіленої маршрутизації або налаштування прав доступу.

Патерн Prototype

PROTOTYPE дозволяє створювати нові об'єкти шляхом копіювання існуючих, а не створювати їх з нуля. Це забезпечує ефективність у ситуаціях, коли створення об'єкта є ресурсомістким процесом. Цей шаблон часто використовується для роботи з об'єктами, що мають багато властивостей або складну структуру.

Патерн Adapter

ADAPTER використовується для приведення інтерфейсу одного класу у відповідність до вимог іншого інтерфейсу. Це дозволяє об'єктам із несумісними інтерфейсами працювати разом.

Застосування патерну Adapter

Переваги

Сумісність з новими або сторонніми бібліотеками без зміни коду клієнта.
Гнучкість — дозволяє використовувати класи з несумісними інтерфейсами.
Повторне використання коду — адаптер можна використовувати для різних цілей.

Проблема

Оскільки в мене клієнт-серверна архітектура, то мені потрібно реалізувати взаємодію клієнта з сервером. Я робив це за допомогою класу HttpClient, який надсилає повідомлення у форматі json. Але оскільки мені потрібно надсилати дані у вигляді класів, то потрібно перетворювати об’єкти у рядок json.

Рішення

В цьому випадку я використав патерн адаптер для перетворення інформації, яку потрібно надіслати у формат json. Потім ці дані прочитає сервер і перетворить їх у потрібний клас.

```
public static class JsonRequestConvert
{
    3 references
    public static StringContent ConvertToJsonRequest(object obj)
    {
        return new StringContent(
            System.Text.Json.JsonSerializer.Serialize(obj),
            System.Text.Encoding.UTF8,
            "application/json"
        );
    }
}
```

Рис. 1 – Клас, який приймає об’кт та повертає StringContent для надсилання даних до сервера

Клас JsonRequestConvert, який містить метод ConvertToJsonRequest(), який використовується для перетворення об'єкта у формат JSON і створення об'єкта StringContent.

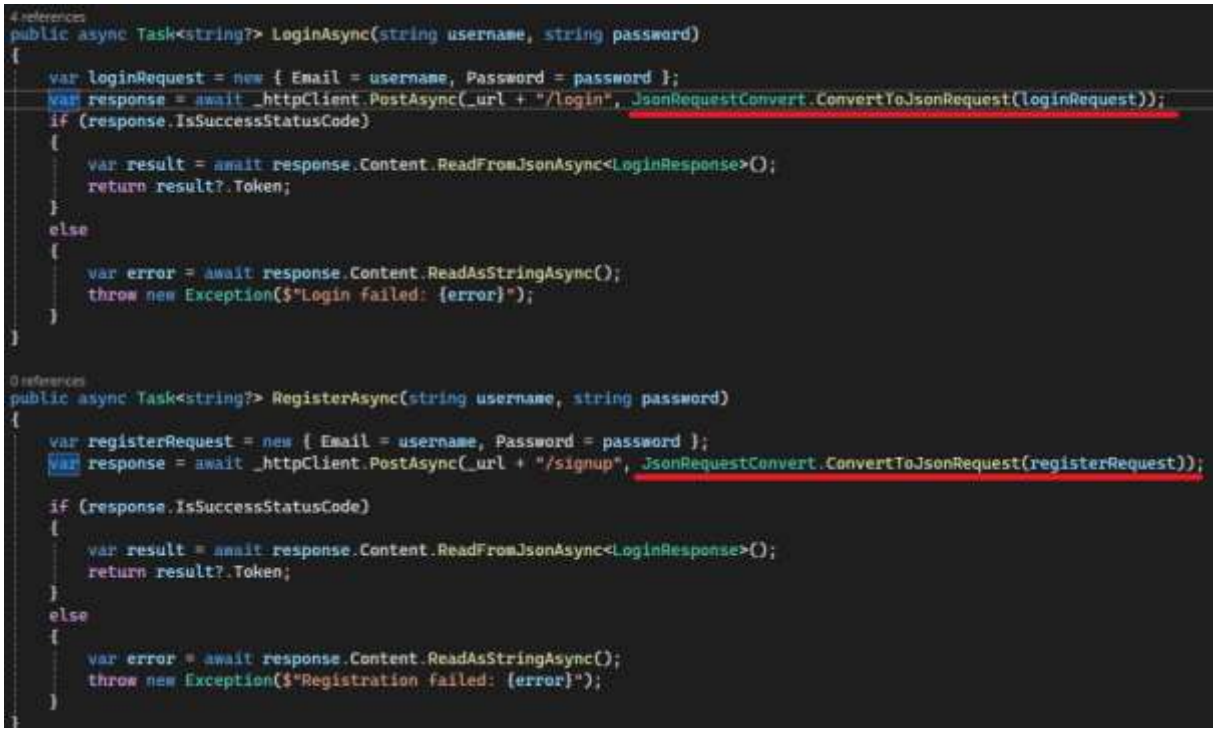

Рис. 2 – Використання JsonRequestConvert у API

LoginAsync() і RegisterAsync() відповідають за авторизацію та реєстрацію користувачів. У кожному з них створюється об’єкт із відповідними даними (ім'я користувача та пароль). Цей об'єкт перетворюється на JSON за допомогою ConvertToJsonRequest, після чого використовується в запиті PostAsync.

Якщо відповідь від сервера має успішний статус, дані розпарсюються й повертаються користувачеві (наприклад, токен авторизації). У разі помилки формується виключення із відповідним повідомленням.

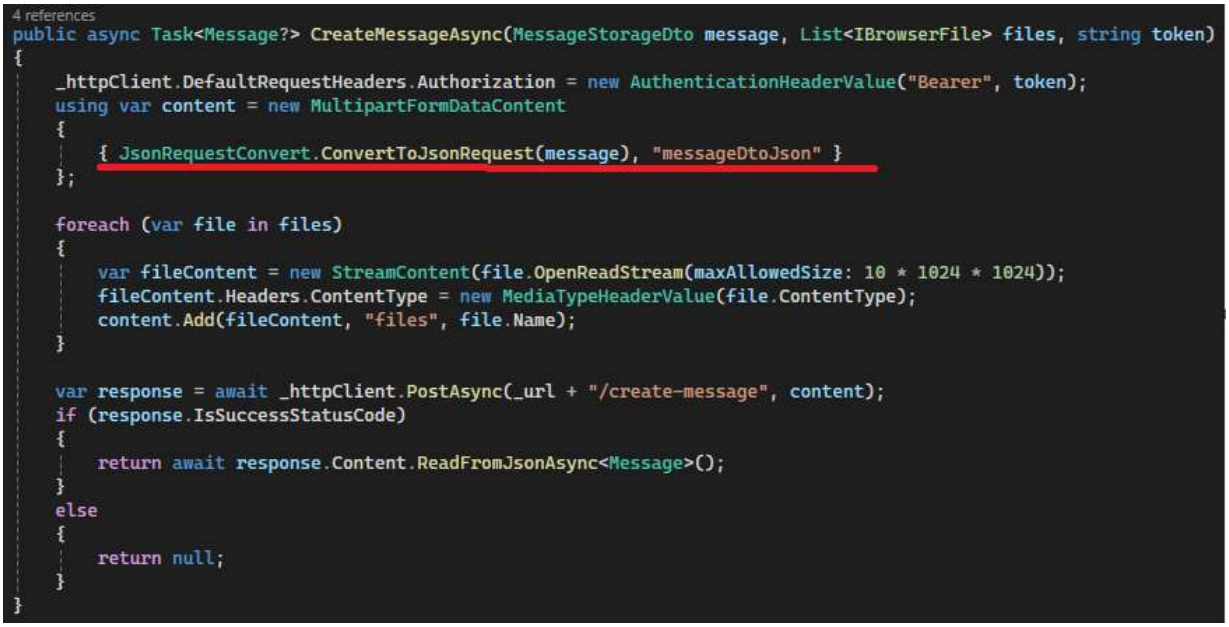

Рис. 3 – Використання JsonRequestConvert у API

Метод CreateMessageAsync() використовується для створення повідомлення із можливістю завантаження файлів. У цьому методі передається об'єкт message і список файлів. Спочатку об'єкт message перетворюється на JSON, який додається до форми MultipartFormDataContent. Далі кожен файл із переданого списку додається до форми, встановлюється його ім’я та тип. Після формування запиту він надсилається на сервер через PostAsync.

Якщо відповідь від сервера успішна, повідомлення повертається користувачеві. У разі помилки метод повертає null.

Висновок: я створив клас JsonRequestConvert з метод ConvertToJsonRequest(), який спрощує підготовку даних для передачі через HTTP-запити у форматі JSON. Його використання в API, як у методах авторизації, реєстрації та створення повідомлень, забезпечує зручність і зменшує дублювання коду, підвищуючи читабельність і підтримуваність програмного забезпечення.