Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №9

Технології розроблення програмного забезпечення

«Різні види взаємодії додатків: Client-Server, Peer-to-Peer, Service-Oriented Architecture»

Варіант 2

Виконала
студентка групи ІА–24:
Кійко А. О.

Перевірив
Мягкий М. Ю.

Київ 2024

ЗМІСТ

Тема: HTTP-сервер (state, builder, factory method, mediator, composite, p2p). Сервер повинен мати можливість розпізнавати вхідні запити і формувати коректні відповіді (згідно протоколу HTTP), надавати сторінки chtml (html сторінки з додаванням найпростіших C# конструкцій на розсуд студента), вести статистику вхідних запитів, обробку запитів у багатопотоковому/подієвому режимах.

Короткі теоретичні відомості

У архітектурі **Client-Server** система поділяється на дві основні складові: клієнти (Client) та сервер(и) (Server). Сервер виступає центральним вузлом, що надає певний набір послуг (наприклад, доступ до баз даних, веб-сторінок, обчислювальних ресурсів), тоді як клієнти звертаються до нього із запитами. Такий підхід характеризується чітким розподілом ролей: сервер управляє ресурсами та логікою, а клієнт ініціює запити та отримує результати. Перевагами є централізований контроль, зручність адміністрування та безпеки, а недоліком може стати залежність від доступності та продуктивності сервера.

На відміну від моделі клієнт-сервер, в **Peer-to-Peer** усі вузли мережі мають однаковий статус: кожен вузол може бути як клієнтом, так і сервером одночасно. Кожен "пір" може надавати та отримувати ресурси напряму від інших. Така децентралізована структура підвищує стійкість до відмов окремих вузлів, оскільки немає єдиного "центрального" сервера. Це сприяє масштабованості та зниженню навантаження на окремі точки, але ускладнює питання безпеки, пошуку ресурсів та керування версіями даних.

Service-Oriented Architecture (SOA) ґрунтується на ідеї створення незалежних програмних компонентів, які реалізують конкретні послуги (сервіси) та взаємодіють через чітко визначені інтерфейси. Сервіси можуть бути розгорнуті на різних платформах та технологіях, але завдяки стандартизованим протоколам (наприклад, веб-сервісам) їх можна легко

об’єднати для побудови складніших бізнес-процесів. SOA сприяє гнучкості та повторному використанню компонентів, знижує зв’язаність системи та дозволяє ефективно реагувати на зміни вимог шляхом додавання або заміни сервісів.

Хід роботи

Завдання:

1. Ознайомитися з короткими теоретичними відомостями.
2. Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.
3. Застосування одного з розглянутих шаблонів при реалізації програми.
4. Скласти звіт про виконану роботу.

Реалізація архітектури Peer-to-Peer

Peer-to-peer (P2P) — варіант архітектури системи, в основі якої стоїть мережа рівноправних вузлів. Реалізація архітектурної моделі “Peer-to-peer” у моєму проєкті полягає у забезпеченні можливості перенаправлення (redirect) запиту деякого ресурсу (Web-сторінки) від клієнта на інший (резервний) Http-сервер у випадку, якщо заданий ресурс відсутній на основному сервері. Налаштування доступу до резервного серверу здійснюється у конфігураційному файлу застосунку. Приклад налаштування наведений на рис. 1

```
10      <add key="remote_host" value="localhost"/>
11      <add key="remote_port" value="7777"/>
```

Рис.1 Приклад налаштування доступу до резервного серверу

У свою чергу, віддалений сервер також може взаємодіяти з іншим віддаленим сервером через власні налаштування і т.д. Єдиною умовою визначення налаштувань є забезпечення відсутності циклічних посилань. Таким чином, кожен екземпляр застосунку може виступати:

- як сервер, коли до нього безпосередньо звертається клієнт із запитом;

- як клієнт, коли він звертається до іншого резервного серверу при обробці запиту Web-сторінки.

Для реалізації даного функціоналу мною був створений окремий клас **P2pRedirect**, що містить публічний метод **HandleRedirect,** в який передається пакет запиту (масив байт) та сокет запиту. Відповідно у методі створюється новий сокет для забезпечення доступу до резервного серверу, який відправляє дані запиту на резервний сервер та отриману відповідь одразу (порціями у випадку великого розміру) передає клієнту, після чого створений сокет закривається.

Реалізація класу P2pRedirect наведена на рис. 2.

```
using HttpServApp.Models;
using System.Net.Sockets;

namespace HttpServApp.p2p
{
  /// <summary>
  /// Клас перенаправлення запиту на інший резервний сервер
  /// </summary>
  internal class P2pRedirect
  {
    private string host = Configuration.RemoteHost ?? "127.0.0.1";
    private int port = Configuration.RemotePort ?? 80;
    public P2pRedirect() { }

    public P2pRedirect(string host, int port)
    {
      this.host = host;
      this.port = port;
    }

    /// <summary>
    /// Метод перенаправлення даних запиту на резервний сервер
    /// та відправки відповіді від нього клієнту
    /// </summary>
    /// <param name="sendMsg"></param>
    /// <param name="requestSocket"></param>
    /// <returns></returns>
    public int HandleRedirect(byte[] sendMsg, Socket requestSocket)
    {
      // Створюємо сокет для перенаправлення даних запиту на резервний сервер
      Socket redirectSocket = new Socket(SocketType.Stream, ProtocolType.Tcp);
      int sendCount = 0;

      try
```

```
      {
        // Встановлюємо з'єднання
        redirectSocket.Connect(host, port);
        if (redirectSocket.Connected)
        {
          // Відправляємо на резервний сервер масив байтів запиту
          int sentCount = redirectSocket.Send(sendMsg);
          redirectSocket.ReceiveTimeout = 60 * 1000;

          // Повертаємо зміст відповіді від віддаленого серверу
          // Встановлюємо розмір блоку даних
          byte[] bufferBytes = new byte[1024 * 16];
          // Цикл, поки не досягли закінчення масиву
          do
          {
            // Читаємо порцію даних
            int readByteCount = redirectSocket.Receive(bufferBytes,
bufferBytes.Length, SocketFlags.None);
            // Одразу віддаємо цю порцію клієнту (в requestSocket)
            int sendByteCount = requestSocket.Send(bufferBytes, 0, readByteCount,
SocketFlags.None);
            // Підбиваємо підсумки переданих байтів
            sendCount += sendByteCount;
            if (readByteCount != sendByteCount)
            {
              throw new Exception("HandleRedirect: кількість зчитаних байтів не
співпадає з кількістю переданих!");
            }
            // Трошки почекаємо: може, "залишок" не встиг прийти
            if (redirectSocket.Available <= 0)
                Thread.Sleep(500);
          }
          while (redirectSocket.Available > 0);
        }
        else
        {
          throw new Exception("HandleRedirect: клієнтський сокет втратв з'єднання");
        }
      }
      catch (Exception ex)
      {
        Console.WriteLine($"==== HandleRedirect exception
({host}:{port}):\n{ex.Message}");
      }
      finally
      {
        redirectSocket.Shutdown(SocketShutdown.Both);
        redirectSocket.Close();
        redirectSocket.Dispose();
      }
      return sendCount;
```

```
    }
  }
}
```

Рис. 2 - Клас P2pRedirect

Використання моделі взаємодії p2p здійснюється наступним чином. У класі **BuilderPage** (Builder-клас побудови відповіді на запит Web-сторінки) у методі **BuildStatus** (формування статусу обробки запиту) при відсутності на основному сервері фалу сторінки, що запитується, та визначення в налаштуваннях застосунку резервного серверу, статус запиту набуває значення **StatusEnum.REDIRECT**. Фрагмент коду наведений на рис. 3

```
      // Якщо статус обробки запиту сторінки StatusEnum.NOT_FOUND
      // і заданий резервний віддалений сервер, передаємо запит серверу (модель
взаємодії peer-to-peer)
      if (httpRequestPage.Status == StatusEnum.NOT_FOUND &&
        !string.IsNullOrEmpty(Configuration.RemoteHost))
      {
        httpRequestPage.Status = StatusEnum.REDIRECT;
        httpRequestPage.Message += $"<br/>Запит перенаправляється на резервний
сервер: {Configuration.RemoteHost}:{Configuration.RemotePort}";
      }
```

Рис. 3 - Фрагмент присвоєння статусу обробки запиту StatusEnum.REDIRECT

Відповідно, у класі **ValidatePageState** (що визначає стан запиту після валідації Web-сторінки), після формування відповіді клієнту аналізується статус обробки запиту і у разі, якщо він дорівнює StatusEnum.REDIRECT, створюється об’єкт класу **P2pRedirect** та викликається метод **HandleRedirect** (рис. 4).

```
      // Якщо статус обробки запиту сторінки StatusEnum.REDIRECT
      // і заданий резервний віддалений сервер, передаємо запит цьому серверу
(модель взаємодії peer-to-peer)
      if (httpRequest.Status == StatusEnum.REDIRECT)
      {
        P2pRedirect p2PRedirect = new P2pRedirect(Configuration.RemoteHost,
Configuration.RemotePort ?? 80);

p2PRedirect.HandleRedirect(Encoding.UTF8.GetBytes(httpRequest.StringRequest),
socket);
      }
```

```
else
{
  // Відсилаємо відповідь клієнту
  httpRequest.SendResponseByte(socket, sendBytes);
}
```

Рис. 4 - Створення об’єкту класу P2pRedirect та виклик методу HandleRedirect

У результаті такої послідовної обробки ланцюжка запитів на сервері бази даних буде зберігатись **N запитів** відповідно до кількості серверів у ланцюжку переходів між ними (перші N-1 запит зі статусом обробки **308 REDIRECT**; N-й (кінцевий) запит буде мати один з можливих статусів:

- **200 OK**: сторінка знайдена;
- **405 NOT_ALLOWED**: знайдено декілька сторінок із заданим ім’ям (якщо запит виконувався з використанням маски пошуку);
- **404 NOT_FOUND** - сторінку не знайдено.

  Приклад відображення статистики вибору даних запитів за період наведено на рис. 5.

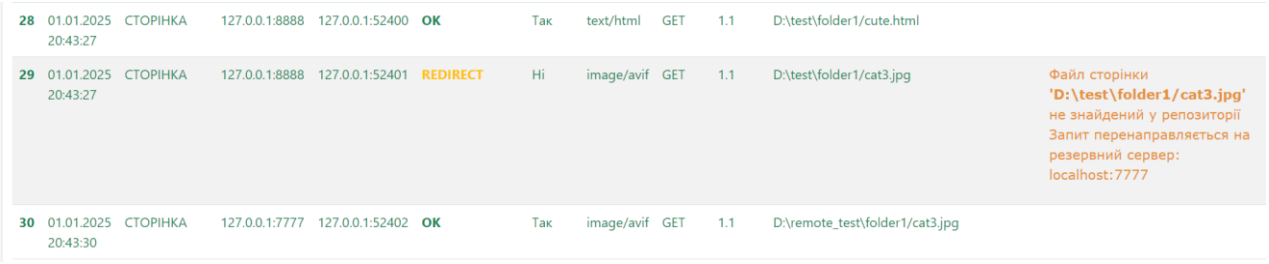

Рис. 5 - Результати запиту статистики при функціонуванні 2-х екземплярів серверів

Загальний вигляд структури класів та їх взаємозв’язків наведений на діаграмі класів (рис. 6).

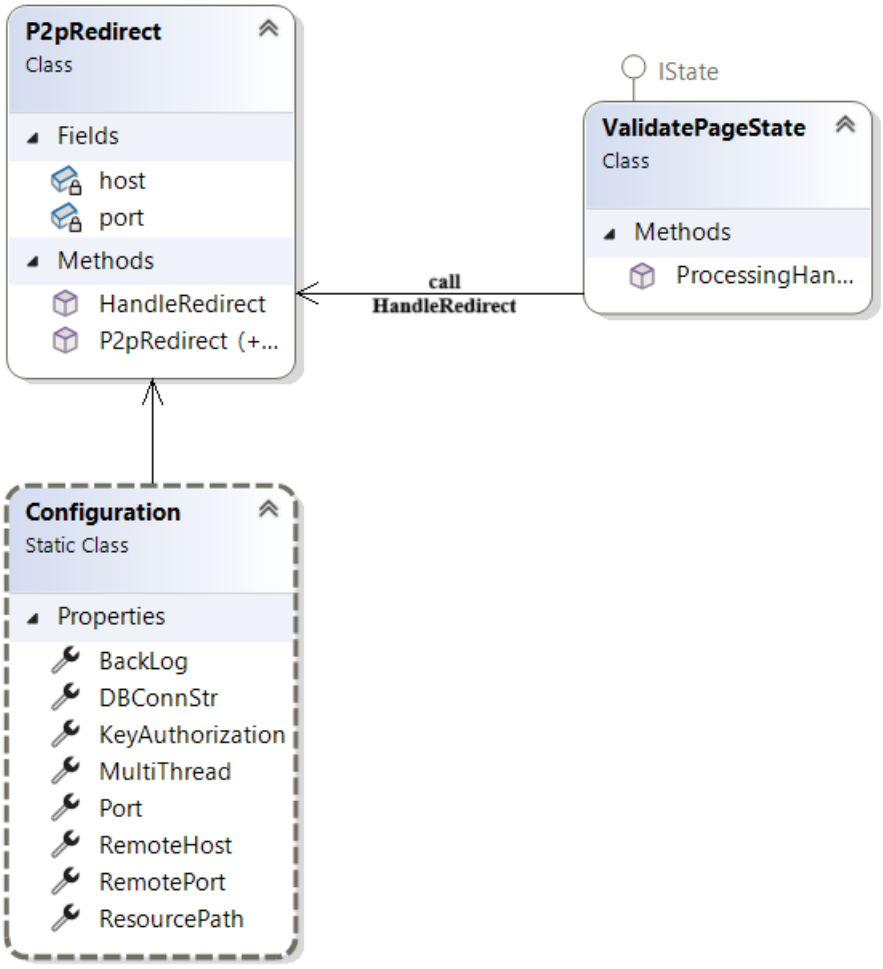

Рис. 6 - Діаграма класів

Висновок: у ході виконання даної лабораторної роботи я реалізувала взаємодію окремих екземплядів додатка за архітектурною моделлю P2P (peer-to-peer), яка дозволяє перенаправляти виконання запитів між різними Http-серверами. Відповідно, кожен додаток може виступати одночасно як сервер та як клієнт. Це дозволяє забезпечити ефективність розподілу ресурсів сховища сторінок (файлів зображень, стилів тощо) та підвищує масштабованість системи.

Посилання на репозиторій: AnnaKiikoIA24/TRPZ_labs_Kiiko_AO_IA24 (github.com)