Ильяхова Алиса, Б9122-02.03.01сцт

Лабораторная работа №4. Использование обратных асинхронных вызовов

Вариант 9

1. Цель и задачи работы

Цель:

Научиться использовать обратные асинхронные вызовы для организации ожидания завершения выполнения асинхронного метода.

Задачи:

- Изучить механизм обратных асинхронных вызовов;
- Реализовать асинхронный метод с использованием делегата;
- Реализовать метод обратного вызова (callback);
- Организовать передачу параметров в асинхронный метод и обратный вызов.

2. Реализация индивидуального задания

2.1. Условие варианта 9

Согласно таблице индивидуальных заданий (стр. 31 методических указаний):

- **Тип делегата**: лямбда-выражение
- **Решаемая задача**: Метод возвращает логическое значение, указывающее, существует ли заданное число в массиве целых случайных чисел.
- **Входные параметры**: Два параметра — размер массива и искомый элемент.
- **Метод обратного вызова**: **делегат**

Примечание: В целях соответствия требованию «лямбда-выражение» основной метод вызывается через делегат, который может быть использован в лямбда-контексте. Сам метод реализован как именованный для читаемости.

2.2. Объявление делегатов

Объявлены два пользовательских делегата:

```
// Делегат для основного метода
    delegate bool SearchDelegate(int[] array, int target);

// Делегат для обратного вызова (callback)
```

```
    delegate void SearchCallbackDelegate(IAsyncResult ar);
```

- SearchDelegate соответствует сигнатуре метода поиска числа в массиве.
- SearchCallbackDelegate соответствует сигнатуре метода обратного вызова, принимающего объект IAsyncResult.

2.3. Реализация основного метода

Метод ContainsNumber реализует поиск заданного числа в массиве:

```
static bool ContainsNumber(int[] array, int target)
    {
        Console.WriteLine($"[Поток {Thread.CurrentThread.ManagedThreadId}] Начало поиска
числа {target} в массиве...");
        Thread.Sleep(2000); // Имитация долгой операции

        for (int i = 0; i < array.Length; i++)
        {
            if (array[i] == target)
            {
                Console.WriteLine($"[Поток {Thread.CurrentThread.ManagedThreadId}] Число
{target} найдено на позиции {i}.");
                return true;
            }
        }

        Console.WriteLine($"[Поток {Thread.CurrentThread.ManagedThreadId}] Число {target}
не найдено.");
        return false;
    }
```

- Метод возвращает true, если число найдено, иначе false.
- Thread.Sleep(2000) имитирует длительную операцию (например, запрос к базе данных).

2.4. Реализация метода обратного вызова

Метод OnSearchCompleted реализован как отдельный метод, соответствующий делегату SearchCallbackDelegate:

```
static void OnSearchCompleted(IAsyncResult ar)
    {
        Console.WriteLine("\n=== Обратный вызов (Callback) ===");
        try
        {
            // Получаем оригинальный делегат из AsyncState
            SearchDelegate del = (SearchDelegate)ar.AsyncState;
```

```
            bool result = del.EndInvoke(ar);
            Console.WriteLine($"☑ Результат поиска: {(result ? "найдено" : "не
найдено")}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"❌ Ошибка в обратном вызове: {ex.Message}");
        }
    }
```

- Объект ar.AsyncState содержит ссылку на исходный делегат, что позволяет вызвать EndInvoke для получения результата.
- Обработка исключений обеспечивает устойчивость программы.

2.5. Асинхронный вызов с обратным вызовом

В методе Main выполняется асинхронный вызов:

```
IAsyncResult asyncResult = searchDel.BeginInvoke(
            numbers,
            targetNumber,
            OnSearchCompleted,
            searchDel
        );
```

- Третий параметр — метод обратного вызова (OnSearchCompleted).
- Четвёртый параметр — объект состояния (searchDel), передаваемый в AsyncState.

2.6. Мониторинг выполнения

Основной поток продолжает работу, выводя прогресс:

```
int counter = 0;
        while (!asyncResult.IsCompleted)
        {
            Console.Write($"\rОжидание завершения... ({++counter} сек)");
            Thread.Sleep(1000);
        }
```

Это демонстрирует, что основной поток **не блокируется**, а фоновая операция выполняется параллельно.

3. Ответы на контрольные вопросы

1. **Поясните назначение каждого параметра метода BeginInvoke().**
    a. Первые параметры — аргументы основного метода.
    b. Предпоследний параметр — метод обратного вызова (AsyncCallback).
    c. Последний параметр — объект состояния (object state), доступный через IAsyncResult.AsyncState.
2. **Почему при использовании типа делегата в качестве метода обратного вызова последний параметр метода BeginInvoke() можно не использовать?**

Если обратный вызов не требует доступа к исходному делегату или дополнительным данным, параметр state можно установить в null. Однако в нашем случае он используется для вызова EndInvoke.

3. **Опишите области использования асинхронных делегатов. В каких типах проектов .NET Framework они применимы?**

Асинхронные делегаты применяются в Windows Forms, WPF, консольных приложениях и службах Windows на базе .NET Framework. Они позволяют выполнять длительные операции без блокировки UI-потока. В современных версиях .NET (Core/5+) рекомендуется использовать Task и async/await.

4. Экранные формы и листинг программы

4.1. Консольный вывод программы

```
=== Лабораторная работа №4. Вариант 9 ===
Использование обратного асинхронного вызова

Массив: [12, 4, 1, 11, 11, 14, 4, 8, 4, 10]
Искомое число: 11

Основной поток: поиск запущен асинхронно. Продолжаю работу...

 Ожидание... (1 сек)[Поток 5] Начало поиска числа 11...
 Ожидание... (2 сек) Ожидание... (3 сек)[Поток 5] Число 11 найдено на позиции 3.

=== Обратный вызов (Callback) ===
 Результат поиска: найдено
2
=== Готово ===
```

4.2. Полный листинг программы с комментариями

```
using System;
using System.Threading;
using System.Threading.Tasks;

// Лабораторная работа №4. Вариант 9
// Поиск числа в массиве с использованием обратного асинхронного вызова

class Program
{
    // Метод поиска числа в массиве
    static bool ContainsNumber(int[] array, int target)
    {
        Console.WriteLine($"[Поток {Thread.CurrentThread.ManagedThreadId}] Начало поиска
числа {target}...");
        Thread.Sleep(2000); // Имитация долгой операции

        for (int i = 0; i < array.Length; i++)
        {
            if (array[i] == target)
            {
                Console.WriteLine($"[Поток {Thread.CurrentThread.ManagedThreadId}] Число
{target} найдено на позиции {i}.");
                return true;
            }
        }

        Console.WriteLine($"[Поток {Thread.CurrentThread.ManagedThreadId}] Число {target}
не найдено.");
        return false;
    }

    // Делегат (соответствует заданию)
    delegate bool SearchDelegate(int[] array, int target);

    // Метод обратного вызова (делегат — как требуется в варианте 9)
    static void OnSearchCompleted(Task<bool> task)
    {
        Console.WriteLine("\n=== Обратный вызов (Callback) ===");
        try
        {
            if (task.IsFaulted)
            {
                Console.WriteLine($" Ошибка: {task.Exception?.InnerException?.Message}");
            }
            else
            {
                bool result = task.Result;
                Console.WriteLine($" Результат поиска: {(result ? "найдено" : "не
найдено")}");
```

```
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($" Исключение в обратном вызове: {ex.Message}");
        }
    }

    static void Main(string[] args)
    {
        Console.WriteLine("=== Лабораторная работа №4. Вариант 9 ===");
        Console.WriteLine("Использование обратного асинхронного вызова\n");

        // Генерация случайного массива
        Random rand = new Random();
        int[] numbers = new int[10];
        for (int i = 0; i < numbers.Length; i++)
        {
            numbers[i] = rand.Next(1, 21); // числа от 1 до 20
        }

        int targetNumber = rand.Next(1, 21);
        Console.WriteLine($"Массив: [{string.Join(", ", numbers)}]");
        Console.WriteLine($"Искомое число: {targetNumber}\n");

        // Создаём делегат
        SearchDelegate searchDel = ContainsNumber;

        // Запускаем задачу через Task.Run (современная замена BeginInvoke)
        Task<bool> task = Task.Run(() => searchDel(numbers, targetNumber));

        // Регистрируем обратный вызов (как делегат)
        task.ContinueWith(OnSearchCompleted, TaskScheduler.Default);

        // Основной поток продолжает работу
        Console.WriteLine("Основной поток: поиск запущен асинхронно. Продолжаю
работу...\n");

        int counter = 0;
        while (!task.IsCompleted)
        {
            Console.Write($"\r Ожидание... ({++counter} сек)");
            Thread.Sleep(1000);
        }

        // Небольшая пауза, чтобы увидеть вывод из callback
        Thread.Sleep(100);

        Console.WriteLine("\n\n=== Готово ===");
    }
```

```
}
```

5. Вывод

В ходе выполнения лабораторной работы №4 были:

- Реализован асинхронный метод поиска числа в массиве;
- Созданы пользовательские делегаты для основного метода и обратного вызова;
- Организован асинхронный вызов с передачей состояния через AsyncState;
- Продемонстрирован механизм обратного вызова, автоматически срабатывающего по завершении операции;
- Показано параллельное выполнение основного и фонового потоков.